

OS, Operating System, Revision 2, for 600XL / 800XL

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:	OS.ASM found in SYSTEMS\OS\REV2
Title of document:	OS, Operating System, Revision 2, for 600XL / 800XL
Author(s):	Multiple authors. Last update by R. K. Nordin (NORDIN, Richard K. (Hud))
Original file date:	1983-11-01 (found in description)
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-1983-11-01-SOFT-0012A-F which should be quoted in any communication.
Tags:	#Atari #8bit #6502 #600XL #800XL #OS

Additional details from "Information" file:

OS Rev. 2 is the current version of the Rev. 2 OS for the 600XL and 800XL. It is based on a CRASS version of Rev. 2 and incorporates source changes only (no changes to the object generated) to improve the documentation.

Comments:

This is the printout of the original source code, fully commented.

Typos were corrected in comments. The source code was left unaltered.

According to the introduction, this OS Revision 2 is dated 1983-11-01. But the date 1984 appears in the © copyright Atari.

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

Formatting:

- Comments are in grey, so that the black text (code source) stands out.
- Conditional compilation IF ELSE ENDIF sections have a pale orange background.
- Presentation / Description of source code in blue.
- Presentation / Description of tables & data in green.
- Important 6502 instructions are highlighted in fluo.

JMP JSR RTS BEQ BNE BMI BPL BCS BCC BVS BVC

This page intentionally left blank

OS, Operating System, Revision 2, for 600XL / 800XL

Copyright © 1984 Atari.

Unauthorized reproduction, adaptation, distribution, performance or display of this computer program or the associated audiovisual work is strictly prohibited.

OS - Operating System

Notes

This represents an attempt to bring the OS source into conformance with the Atari Internal Software Standards as defined in the Software Development Committee Report on Procedures and Standards (1981-10-27).

Due to time constraints, the entire source could not be brought up to the standard, particularly in the area of subroutine header documentation (ENTRY, EXIT, CHANGES and CALLS).

More complete and consistent conformance to the standard is planned for the next revision of the Operating System (Revision 3).

Modifications

Revision A (400/800)

D. Crane/A. Miller/L. Kaplan/R. Whitehead

Revision B (400/800)

Fix several problems.

M. Mahar/R. S. Scheiman

Revision 10 (1200XL)

Support 1200XL, add new features.

H. Stewart/L. Winner/R. S. Scheiman/Y. M. Chen/M. W. Colburn – 1982-10-26

Revision 11 (1200XL)

Fix several problems.

R. S. Scheiman – 1982-12-23

Revision 1 (600XL/800XL)

Support PBI and on-board BASIC.

R. S. Scheiman/R. K. Nordin/Y. M. Chen – 1983-03-11

Revision 2 (600XL/800XL)

Fix several problems.

R. S. Scheiman – 1983-05-10

Bring closer to Coding Standard (object unchanged).

R. K. Nordin – 1983-11-01

Program Structure

The sections of the OS appear in the following order with corresponding subtitles:

Equates and Definitions

- System Symbol Equates
- System Address Equates
- Miscellaneous Address Equates
- Macro Definitions

Code and Data

- First 8K ROM Identification and Checksum

- Interrupt Handler
- Initialization
- Disk Input/Output
- Relocating Loader
- Self-test, Part 1
- Parallel Input/Output
- Peripheral Handler Loading Facility, Part 1
- Self-test, Part 2
- Peripheral Handler Loading Facility, Part 2

- International Character Set

- Self-test, Part 3
- Floating Point Package

- Domestic Character Set

- Device Handler Vector Tables
- Jump Vectors
- Generic Parallel Device Handler Vector Table

- \$E4C0 Patch
- Central Input/Output
- Peripheral Handler Loading Facility, Part 3
- \$E912 Patch
- Peripheral Handler Loading Facility, Part 4
- \$E959 Patch
- Serial Input/Output
- Keyboard, Editor and Screen Handler, Part 1
- Peripheral Handler Loading Facility, Part 5
- \$EF6B Patch
- Keyboard, Editor and Screen Handler, Part 2
- \$F223 Patch
- Keyboard, Editor and Screen Handler, Part 3

\$FCD8 Patch
Cassette Handler
Printer Handler
Self-test, Part 4

Second 8K ROM Identification and Checksum
6502 Machine Vectors

System Symbol Equates

** Assembly Option Equates

```
FALSE EQU 0
TRUE EQU not FALSE

VGC SET TRUE ;virtual game controllers
RAMSYS SET FALSE ;not RAM based system
LNBUG SET FALSE ;no LNBUG interface
ACMI SET FALSE ;no asynchronous communications module interface
```

** Identification Equates

```
IDREV EQU $02 ;identification revision number
IDDAY EQU $10 ;identification day
IDMON EQU $05 ;identification month
IDYEAR EQU $83 ;identification year
IDCPU EQU $02 ;identification CPU series
IDPN1 EQU 'B' ;identification part number field 1
IDPN2 EQU 'B' ;identification part number field 2
IDPN3 EQU $00 ;identification part number field 3
IDPN4 EQU $00 ;identification part number field 4
IDPN5 EQU $01 ;identification part number field 5
```

** Configuration Equates

```
*
* NOTES
* Problem: last byte of HATABS (as defined by MAXDEV)
* overlaps first power-up validation byte.
```

```
MAXDEV EQU 33 ;offset to last possible entry of HATABS
IOCBSZ EQU 16 ;length of IOCB

SEIOCB EQU 0*IOCBSZ ;screen editor IOCB index
MAXIOC EQU 8*IOCBSZ ;first invalid IOCB index

DSCTSZ EQU 128 ;disk sector size

LEDGE EQU 2 ;left edge
REDGE EQU 39 ;right edge

INIML EQU $0700 ;initial MEMLO

ICSORG EQU $CC00 ;international character set origin
DCSORG EQU $E000 ;domestic character set origin
```

** IOCB Command Code Equates

```
OPEN EQU $03 ;open
GETREC EQU $05 ;get record
GETCHR EQU $07 ;get character(s)
PUTREC EQU $09 ;put record
PUTCHR EQU $0B ;put character(s)
CLOSE EQU $0C ;close
STATIS EQU $0D ;status
SPECIL EQU $0E ;special
```

**** Special Entry Command Equates**

; Screen Commands

DRAWLN EQU \$11 ;draw line
FILLIN EQU \$12 ;draw line with right fill

**** ICAX1 Auxiliary Byte 1 Equates**

APPEND EQU \$01 ;open write append (D:) or screen read (E:)
DIRECT EQU \$02 ;open for directory access (D:)
OPNIN EQU \$04 ;open for input (all devices)
OPNOT EQU \$08 ;open for output (all devices)
MXDMOD EQU \$10 ;open for mixed mode (E:, S:)
INSCLR EQU \$20 ;open for input without clearing screen (E:, S:)

**** Device Code Equates**

CASSET EQU 'C' ;cassette
DISK EQU 'D' ;disk
SCREDT EQU 'E' ;screen editor
KBD EQU 'K' ;keyboard
PRINTR EQU 'P' ;printer
DISPLAY EQU 'S' ;screen display

**** Character and Key Code Equates**

CLS EQU \$7D ;clear screen
EOL EQU \$9B ;end of line (RETURN)

HELP EQU \$11 ;key code for HELP
CNTLF1 EQU \$83 ;key code for CTRL-F1
CNTLF2 EQU \$84 ;key code for CTRL-F2
CNTLF3 EQU \$93 ;key code for CTRL-F3
CNTLF4 EQU \$94 ;key code for CTRL-F4
CNTL1 EQU \$9F ;key code for CTRL-1

**** Status Code Equates**

SUCCESS EQU 1 ;successful operation

BRKABT EQU 128 ;BREAK key abort
PRVOPN EQU 129 ;IOCB already open error
NONDEV EQU 130 ;nonexistent device error
WRONLY EQU 131 ;IOCB opened for write only error
NVALID EQU 132 ;invalid command error
NOTOPN EQU 133 ;device/file not open error
BADIOC EQU 134 ;invalid IOCB index error
RONLY EQU 135 ;IOCB opened for read only error
EOFERR EQU 136 ;end of file error
TRNRCD EQU 137 ;truncated record error
TIMEOUT EQU 138 ;peripheral device timeout error
DNACK EQU 139 ;device does not acknowledge command error
FRMERR EQU 140 ;serial bus framing error
CURSOR EQU 141 ;cursor overrange error
OVRUN EQU 142 ;serial bus data overrun error
CHKERR EQU 143 ;serial bus checksum error
DERRR EQU 144 ;device done (operation incomplete) error
BADMOD EQU 145 ;bad screen mode number error
FNCNOT EQU 146 ;function not implemented in handler error

SCRMEM EQU 147 ;insufficient memory for screen mode error

** DCB Device Bus ID Equates

DISKID EQU \$31 ;disk bus ID
PDEVN EQU \$40 ;printer bus ID
CASET EQU \$60 ;cassette bus ID

** Bus Command Equates

FOMAT EQU '!' ;format command
PUTSEC EQU 'P' ;put sector command
READ EQU 'R' ;read command
STATC EQU 'S' ;status command
WRITE EQU 'W' ;write command

** Command Auxiliary Byte Equates

DOUBLE EQU 'D' ;print 20 characters double width
NORMAL EQU 'N' ;print 40 characters normally
PLOT EQU 'P' ;plot
SIDWAY EQU 'S' ;print 16 characters sideways

** Bus Response Equates

ACK EQU 'A' ;device acknowledged
COMPLT EQU 'C' ;device successfully completed operation
ERROR EQU 'E' ;device incurred error in attempted operation
NACK EQU 'N' ;device did not understand

** Floating Point Package Miscellaneous Equates

FPREC EQU 6 ;precision
FMPREC EQU FPREC-1 ;length of mantissa

** Cassette Record Type Equates

HDR EQU \$FB ;header
DTA EQU \$FC ;data record
DT1 EQU \$FA ;last data record
EOT EQU \$FE ;end of tape (file)

TONE1 EQU 2 ;record
TONE2 EQU 1 ;playback

** Cassette Timing Equates

WLEADN EQU 1152 ;NTSC 19.2 second WRITE file leader
RLEADN EQU 576 ;NTSC 9.6 second READ file leader
WIRGLN EQU 180 ;NTSC 3.0 second WRITE IRG
RIRGLN EQU 120 ;NTSC 2.0 second READ IRG
WSIRGN EQU 15 ;NTSC 0.25 second WRITE short IRG
RSIRGN EQU 10 ;NTSC 0.16 second READ short IRG
BEEPNN EQU 30 ;NTSC 0.5 second beep duration
BEEPFN EQU 10 ;NTSC 0.16 second beep separation

WLEADP EQU 960 ;PAL 19.2 second WRITE file leader

RLEADP EQU	480	;PAL 9.6 second READ file leader
WIRGLP EQU	150	;PAL 3.0 second WRITE IRG
RIRGLP EQU	100	;PAL 2.0 second READ IRG
WSIRGP EQU	13	;PAL 0.25 second WRITE short IRG
RSIRGP EQU	8	;PAL 0.16 second READ short IRG
BEEPNP EQU	25	;PAL 0.5 second beep duration
BEEFPF EQU	8	;PAL 0.16 second beep separation
WIRGHI EQU	0	;high WRITE IRG
RIRGHI EQU	0	;high READ IRG

** Power-up Validation Byte Value Equates

PUPVL1 EQU	\$5C	;power-up validation value 1
PUPVL2 EQU	\$93	;power-up validation value 2
PUPVL3 EQU	\$25	;power-up validation value 3

** Relocating Loader Miscellaneous Equates

DATAER EQU	156	;end of record appears before END record error
MEMERR EQU	157	;memory insufficient for load error

** Miscellaneous Equates

IOCFRE EQU	\$FF	;IOCB free indicator
B19200 EQU	\$0028	;19200 baud POKEY counter value
B00600 EQU	\$05CC	;600 baud POKEY counter value
HITONE EQU	\$05	;FSK high freq. POKEY counter value (5326 Hz)
LOTONE EQU	\$07	;FSK low freq. POKEY counter value (3995 Hz)
NCOMLO EQU	\$34	;PIA lower NOT COMMAND line command
NCOMHI EQU	\$3C	;PIA raise NOT COMMAND line command
MOTRGO EQU	\$34	;PIA cassette motor ON command
MOTRST EQU	\$3C	;PIA cassette motor OFF command
NODAT EQU	\$00	;SIO immediate operation
GETDAT EQU	\$40	;SIO read data frame
PUTDAT EQU	\$80	;SIO write data frame
CRETRI EQU	13	;number of command frame retries
DRETRI EQU	1	;number of device retries
CTIM EQU	2	;command frame ACK timeout
NBUFSZ EQU	40	;print normal buffer size
DBUFSZ EQU	20	;print double buffer size
SBUFSZ EQU	29	;print sideways buffer size

System Address Equates

** Page Zero Address Equates

```
LNFLG EQU    $0000 ;1-byte LNBUG flag (0 = not LNBUG)
NGFLAG EQU    $0001 ;1-byte memory status (0 = failure)

;    Not Cleared

CASINI EQU    $0002 ;2-byte cassette program initialization address
RAMLO EQU     $0004 ;2-byte RAM address for memory test
TRAMSZ EQU    $0006 ;1-byte RAM size temporary
CMCMD EQU     $0007 ;1-byte command communications

;    Cleared upon Coldstart Only

WARMST EQU    $0008 ;1-byte warmstart flag (0 = coldstart)
BOOT? EQU     $0009 ;1-byte successful boot flags
DOSVEC EQU    $000A ;2-byte disk program start vector
DOSINI EQU    $000C ;2-byte disk program initialization address
APPMHI EQU    $000E ;2-byte applications memory high limit

;    Cleared upon Coldstart or Warmstart

INTZBS EQU    $0010 ;first page zero location to clear

POKMSK EQU    $0010 ;1-byte IRQEN shadow
BRKKEY EQU    $0011 ;1-byte BREAK key flag (0 = no BREAK)
RTCLOK EQU    $0012 ;3-byte real time clock (16 millisecond units)
BUFADR EQU    $0015 ;2-byte disk interface buffer address
ICCOMT EQU    $0017 ;1-byte CIO command table index
DSKFMS EQU    $0018 ;2-byte DOS File Management System pointer
DSKUTL EQU    $001A ;2-byte DOS utility pointer
ABUFPT EQU    $001C ;4-byte ACMI buffer pointer area

ZIOCB EQU     $0020 ;address of page zero IOCB
IOCBAS EQU    $0020 ;16-byte page zero IOCB
ICHIDZ EQU    $0020 ;1-byte handler ID ($FF = IOCB free)
ICDNOZ EQU    $0021 ;1-byte device number
ICCOMZ EQU    $0022 ;1-byte command code
ICSTAZ EQU    $0023 ;1-byte status of last action
ICBALZ EQU    $0024 ;1-byte low buffer address
ICBAHZ EQU    $0025 ;1-byte high buffer address
ICPTLZ EQU    $0026 ;1-byte low PUT-BYTE routine address-1
ICPTHZ EQU    $0027 ;1-byte high PUT-BYTE routine address-1
ICBLLZ EQU    $0028 ;1-byte low buffer length
ICBLHZ EQU    $0029 ;1-byte high buffer length
ICAX1Z EQU    $002A ;1-byte first auxiliary information
ICAX2Z EQU    $002B ;1-byte second auxiliary information
ICSPRZ EQU    $002C ;4-byte spares

ENTVEC EQU    $002C ;2-byte (not used)
ICIDNO EQU    $002E ;1-byte IOCB index (IOCB number times IOCBSZ)
CIOCHR EQU    $002F ;1-byte character for current CIO operation

STATUS EQU    $0030 ;1-byte SIO operation status
CHKSUM EQU    $0031 ;1-byte checksum (single byte sum with carry)
BUFRLO EQU    $0032 ;1-byte low data buffer address
BUFRHI EQU    $0033 ;1-byte high data buffer address
BFENLO EQU    $0034 ;1-byte low data buffer end address
BFENHI EQU    $0035 ;1-byte high data buffer end address
LTEMP EQU     $0036 ;2-byte relocating loader temporary
BUFRFL EQU    $0038 ;1-byte data buffer full flag (0 = not full)
RECVDN EQU    $0039 ;1-byte receive-frame done flag (0 = not done)
XMTDON EQU    $003A ;1-byte transmit-frame done flag (0 = not done)
```

CHKSNT EQU	\$003B	;1-byte checksum sent flag (0 = not sent)
NOCKSM EQU	\$003C	;1-byte no checksum follows data flag (0 = does)
BPTR EQU	\$003D	;1-byte cassette buffer pointer
FTYPE EQU	\$003E	;1-byte cassette IRG type (neg. = continuous)
FE0F EQU	\$003F	;1-byte cassette EOF flag (0 = no EOF)
FREQ EQU	\$0040	;1-byte cassette beep counter
SOUNDR EQU	\$0041	;1-byte noisy I/O flag (0 = quiet)
CRITIC EQU	\$0042	;1-byte critical section flag (0 = not critical)
FMSZPG EQU	\$0043	;7-byte reserved for DOS File Management System
ZCHAIN EQU	\$004A	;2-byte handler linkage chain pointer
DSTAT EQU	\$004C	;1-byte display status
ATTRACT EQU	\$004D	;1-byte attract-mode timer and flag
DRKMSK EQU	\$004E	;1-byte attract-mode dark (luminance) mask
COLRSH EQU	\$004F	;1-byte attract-mode color shift
TMPCHR EQU	\$0050	;1-byte temporary character
HOLD1 EQU	\$0051	;1-byte temporary
LMARGN EQU	\$0052	;1-byte text column left margin
RMARGN EQU	\$0053	;1-byte text column right margin
ROWCRS EQU	\$0054	;1-byte cursor row
COLCRS EQU	\$0055	;2-byte cursor column
DINDEX EQU	\$0057	;1-byte display mode
SAVMSC EQU	\$0058	;2-byte saved memory scan counter
OLDROW EQU	\$005A	;1-byte prior row
OLDCOL EQU	\$005B	;2-byte prior column
OLDCHR EQU	\$005D	;1-byte saved character under cursor
OLDADR EQU	\$005E	;2-byte saved cursor memory address
FKDEF EQU	\$0060	;2-byte function key definition table address
PALNTS EQU	\$0062	;1-byte PAL/NTSC indicator (0 = NTSC, PAL = 1)
LOGCOL EQU	\$0063	;1-byte logical line cursor column
ADRESS EQU	\$0064	;2-byte temporary address
MLTTMP EQU	\$0066	;1-byte temporary
OPNTMP EQU	\$0066	;1-byte open temporary
TOADR EQU	\$0066	;2-byte destination address
SAVADR EQU	\$0068	;2-byte saved address
FRMADR EQU	\$0068	;2-byte source address
RAMTOP EQU	\$006A	;1-byte RAM size
BUFCNT EQU	\$006B	;1-byte buffer count (logical line size)
BUFSTR EQU	\$006C	;2-byte buffer start pointer
BITMSK EQU	\$006E	;1-byte bit mask for bit map operation
SHFAMT EQU	\$006F	;1-byte shift amount for pixel justification
ROWAC EQU	\$0070	;2-byte draw working row
COLAC EQU	\$0072	;2-byte draw working column
ENDPT EQU	\$0074	;2-byte end point
DELTAR EQU	\$0076	;1-byte row difference
DELTAC EQU	\$0077	;2-byte column difference
KEYDEF EQU	\$0079	;2-byte key definition table address
SWPFLG EQU	\$007B	;1-byte split screen swap flag (0 = not swapped)
HOLDCH EQU	\$007C	;1-byte temporary character
INSDAT EQU	\$007D	;1-byte temporary
COUNTR EQU	\$007E	;2-byte draw iteration count
;	Reserved for Application and Floating Point Package	
;	EQU	\$0080 ;128 bytes reserved for application and FPP

**** Floating Point Package Page Zero Address Equates**

FR0	EQU	\$00D4	;6-byte register 0
FR0M	EQU	\$00D5	;5-byte register 0 mantissa
QTEMP	EQU	\$00D9	;1-byte temporary
FRE	EQU	\$00DA	;6-byte (internal) register E
FR1	EQU	\$00E0	;6-byte register 1
FR1M	EQU	\$00E1	;5-byte register 1 mantissa
FR2	EQU	\$00E6	;6-byte (internal) register 2
FRX	EQU	\$00EC	;1-byte temporary
EEXP	EQU	\$00ED	;1-byte value of exponent
FRSIGN	EQU	\$00EE	;1-byte floating point sign
NSIGN	EQU	\$00EE	;1-byte sign of number
PLYCNT	EQU	\$00EF	;1-byte polynomial degree
ESIGN	EQU	\$00EF	;1-byte sign of exponent
SGNFLG	EQU	\$00F0	;1-byte sign flag
FCHFLG	EQU	\$00F0	;1-byte first character flag
XFMFLG	EQU	\$00F1	;1-byte transform flag
DIGRT	EQU	\$00F1	;1-byte number of digits after decimal point
CIX	EQU	\$00F2	;1-byte current input index
INBUFF	EQU	\$00F3	;2-byte line input buffer
ZTEMP1	EQU	\$00F5	;2-byte temporary
ZTEMP4	EQU	\$00F7	;2-byte temporary
ZTEMP3	EQU	\$00F9	;2-byte temporary
FLPTR	EQU	\$00FC	;2-byte floating point number pointer
FPTR2	EQU	\$00FE	;2-byte floating point number pointer

**** Page One (Stack) Address Equates**

; EQU \$0100 ;256-byte stack

**** Page Two Address Equates**

INTABS	EQU	\$0200	;42-byte interrupt handler table
VDSLST	EQU	\$0200	;2-byte display list NMI vector
VPRCED	EQU	\$0202	;2-byte serial I/O proceed line IRQ vector
VINTER	EQU	\$0204	;2-byte serial I/O interrupt line IRQ vector
VBREAK	EQU	\$0206	;2-byte BRK instruction IRQ vector
VKEYBD	EQU	\$0208	;2-byte keyboard IRQ vector
VSERIN	EQU	\$020A	;2-byte serial input ready IRQ vector
VSEROR	EQU	\$020C	;2-byte serial output ready IRQ vector
VSEROC	EQU	\$020E	;2-byte serial output complete IRQ vector
VTIMR1	EQU	\$0210	;2-byte POKEY timer 1 IRQ vector
VTIMR2	EQU	\$0212	;2-byte POKEY timer 2 IRQ vector
VTIMR4	EQU	\$0214	;2-byte POKEY timer 4 IRQ vector
VIMIRQ	EQU	\$0216	;2-byte immediate IRQ vector
CDTMV1	EQU	\$0218	;2-byte countdown timer 1 value
CDTMV2	EQU	\$021A	;2-byte countdown timer 2 value
CDTMV3	EQU	\$021C	;2-byte countdown timer 3 value
CDTMV4	EQU	\$021E	;2-byte countdown timer 4 value

CDTMV5 EQU	\$0220	;2-byte countdown timer 5 value
VVBLKI EQU	\$0222	;2-byte immediate VBLANK NMI vector
VVBLKD EQU	\$0224	;2-byte deferred VBLANK NMI vector
CDTMA1 EQU	\$0226	;2-byte countdown timer 1 vector
CDTMA2 EQU	\$0228	;2-byte countdown timer 2 vector
CDTMF3 EQU	\$022A	;1-byte countdown timer 3 flag (0 = expired)
SRTIMR EQU	\$022B	;1-byte software key repeat timer
CDTMF4 EQU	\$022C	;1-byte countdown timer 4 flag (0 = expired)
INTMP EQU	\$022D	;1-byte temporary
CDTMF5 EQU	\$022E	;1-byte countdown timer 5 flag (0 = expired)
SDMCTL EQU	\$022F	;1-byte DMACTL shadow
SDLSTL EQU	\$0230	;1-byte DLISTL shadow
SDLSTH EQU	\$0231	;1-byte DLISTH shadow
SSKCTL EQU	\$0232	;1-byte SKCTL shadow
LCOUNT EQU	\$0233	;1-byte relocating loader record length counter
LPENH EQU	\$0234	;1-byte light pen horizontal value
LPENV EQU	\$0235	;1-byte light pen vertical value
BRKKY EQU	\$0236	;2-byte BREAK key vector
VPIRQ EQU	\$0238	;2-byte parallel device IRQ vector
CDEVIC EQU	\$023A	;1-byte command frame device ID
CCOMND EQU	\$023B	;1-byte command frame command
CAUX1 EQU	\$023C	;1-byte command auxiliary 1
CAUX2 EQU	\$023D	;1-byte command auxiliary 2
TEMP EQU	\$023E	;1-byte temporary
ASSERT low TEMP<>\$FF		;may not be the last word on a page
ERRFLG EQU	\$023F	;1-byte I/O error flag (0 = no error)
ASSERT low ERRFLG<>\$FF		;may not be the last word on a page
DFLAGS EQU	\$0240	;1-byte disk flags from sector 1
DBSECT EQU	\$0241	;1-byte disk boot sector count
BOOTAD EQU	\$0242	;2-byte disk boot memory address
COLDST EQU	\$0244	;1-byte coldstart flag (0 = complete)
RECLEN EQU	\$0245	;1-byte relocating loader record length
DSKTIM EQU	\$0246	;1-byte disk format timeout
PDVMSK EQU	\$0247	;1-byte parallel device selection mask
SHPDVS EQU	\$0248	;1-byte PDVS (parallel device selection) shadow
PDIMSK EQU	\$0249	;1-byte parallel device IRQ selection mask
RELADR EQU	\$024A	;2-byte relocating loader relative address
PPTMPA EQU	\$024C	;1-byte parallel device handler temporary
PPTMPX EQU	\$024D	;1-byte parallel device handler temporary
; EQU	\$024E	;6 bytes reserved for Atari
; EQU	\$0254	;23 bytes reserved for Atari
CHSALT EQU	\$026B	;1-byte character set alternate
VSFLAG EQU	\$026C	;1-byte fine vertical scroll count
KEYDIS EQU	\$026D	;1-byte keyboard disable
FINE EQU	\$026E	;1-byte fine scrolling mode
GPRIOR EQU	\$026F	;1-byte PRIOR shadow
PADDL0 EQU	\$0270	;1-byte potentiometer 0
PADDL1 EQU	\$0271	;1-byte potentiometer 1
PADDL2 EQU	\$0272	;1-byte potentiometer 2
PADDL3 EQU	\$0273	;1-byte potentiometer 3
PADDL4 EQU	\$0274	;1-byte potentiometer 4
PADDL5 EQU	\$0275	;1-byte potentiometer 5
PADDL6 EQU	\$0276	;1-byte potentiometer 6
PADDL7 EQU	\$0277	;1-byte potentiometer 7
STICK0 EQU	\$0278	;1-byte joystick 0
STICK1 EQU	\$0279	;1-byte joystick 1

STICK2 EQU	\$027A	;1-byte joystick 2
STICK3 EQU	\$027B	;1-byte joystick 3
PTRIG0 EQU	\$027C	;1-byte paddle trigger 0
PTRIG1 EQU	\$027D	;1-byte paddle trigger 1
PTRIG2 EQU	\$027E	;1-byte paddle trigger 2
PTRIG3 EQU	\$027F	;1-byte paddle trigger 3
PTRIG4 EQU	\$0280	;1-byte paddle trigger 4
PTRIG5 EQU	\$0281	;1-byte paddle trigger 5
PTRIG6 EQU	\$0282	;1-byte paddle trigger 6
PTRIG7 EQU	\$0283	;1-byte paddle trigger 7
STRIG0 EQU	\$0284	;1-byte joystick trigger 0
STRIG1 EQU	\$0285	;1-byte joystick trigger 1
STRIG2 EQU	\$0286	;1-byte joystick trigger 2
STRIG3 EQU	\$0287	;1-byte joystick trigger 3
HIBYTE EQU	\$0288	;1-byte relocating loader high byte value
WMODE EQU	\$0289	;1-byte cassette WRITE mode (\$80 = writing)
BLIM EQU	\$028A	;1-byte cassette buffer limit
IMASK EQU	\$028B	;1-byte (not used)
JVECK EQU	\$028C	;2-byte jump vector or temporary
NEWADR EQU	\$028E	;2-byte relocating address
TXTR0W EQU	\$0290	;1-byte split screen text cursor row
TXTCOL EQU	\$0291	;2-byte split screen text cursor column
TINDEX EQU	\$0293	;1-byte split scree text mode
TXTMSC EQU	\$0294	;2-byte split screen memory scan counter
TXOLD EQU	\$0296	;6-byte OLDROW, OLDCOL, OLDCHR, OLDADR for text
CRETRY EQU	\$029C	;1-byte number of command frame retries
HOLD3 EQU	\$029D	;1-byte temporary
SUBTMP EQU	\$029E	;1-byte temporary
HOLD2 EQU	\$029F	;1-byte (not used)
DMASK EQU	\$02A0	;1-byte display (pixel location) mask
TMLBT EQU	\$02A1	;1-byte (not used)
ESCFLG EQU	\$02A2	;1-byte escape flag (\$80 = ESC detected)
TABMAP EQU	\$02A3	;15-byte (120-bit) tab stop bit map
LOGMAP EQU	\$02B2	;8-byte (32-bit) logical line bit map
INVFLG EQU	\$02B6	;1-byte inverse video flag (\$80 = inverse)
FILFLG EQU	\$02B7	;1-byte right fill flag (0 = no fill)
TMPROW EQU	\$02B8	;1-byte temporary row
TMPCOL EQU	\$02B9	;2-byte temporary column
SCRFLG EQU	\$02BB	;1-byte scroll occurrence flag (0 = not occurred)
HOLD4 EQU	\$02BC	;1-byte temporary
DRETRY EQU	\$02BD	;1-byte number of device retries
SHFLOK EQU	\$02BE	;1-byte shift/control lock flags
BOTSCR EQU	\$02BF	;1-byte screen bottom (24 = normal, 4 = split)
PCOLR0 EQU	\$02C0	;1-byte player-missile 0 color/luminance
PCOLR1 EQU	\$02C1	;1-byte player-missile 1 color/luminance
PCOLR2 EQU	\$02C2	;1-byte player-missile 2 color/luminance
PCOLR3 EQU	\$02C3	;1-byte player-missile 3 color/luminance
COLOR0 EQU	\$02C4	;1-byte playfield 0 color/luminance
COLOR1 EQU	\$02C5	;1-byte playfield 1 color/luminance
COLOR2 EQU	\$02C6	;1-byte playfield 2 color/luminance
COLOR3 EQU	\$02C7	;1-byte playfield 3 color/luminance
COLOR4 EQU	\$02C8	;1-byte background color/luminance
PARMBL EQU	\$02C9	;6-byte relocating loader parameter block
RUNADR EQU	\$02C9	;2-byte run address
HIUSED EQU	\$02CB	;2-byte highest non-zero page address
ZHIUSE EQU	\$02CD	;2-byte highest zero page address
OLDPAR EQU	\$02CF	;6-byte relocating loader parameter block
GBYTEA EQU	\$02CF	;2-byte GET-BYTE routine address
LOADAD EQU	\$02D1	;2-byte non-zero page load address

ZLOADA EQU	\$02D3	;2-byte zero page load address
DSCTLN EQU	\$02D5	;2-byte disk sector length
ACMISR EQU	\$02D7	;2-byte ACMI interrupt service routine address
KRPDEL EQU	\$02D9	;1-byte auto-repeat delay
KEYREP EQU	\$02DA	;1-byte auto-repeat rate
NOCLIK EQU	\$02DB	;1-byte key click disable
HELPGF EQU	\$02DC	;1-byte HELP key flag (0 = no HELP)
DMASAV EQU	\$02DD	;1-byte SDMCTL save/restore
PBPNT EQU	\$02DE	;1-byte printer buffer pointer
PBUFSZ EQU	\$02DF	;1-byte printer buffer size
;		
EQU	\$02E0	;4 bytes reserved for DOS
RAMSIZ EQU	\$02E4	;1-byte high RAM size
MEMTOP EQU	\$02E5	;2-byte top of available user memory
MEMLO EQU	\$02E7	;2-byte bottom of available user memory
HNDLOD EQU	\$02E9	;1-byte user load flag (0 = no handler load)
DVSTAT EQU	\$02EA	;4-byte device status buffer
CBAUDL EQU	\$02EE	;1-byte low cassette baud rate
CBAUDH EQU	\$02EF	;1-byte high cassette baud rate
CRSINH EQU	\$02F0	;1-byte cursor inhibit (0 = cursor on)
KEYDEL EQU	\$02F1	;1-byte key debounce delay timer
CH1 EQU	\$02F2	;1-byte prior keyboard character
CHACT EQU	\$02F3	;1-byte CHACTL shadow
CHBAS EQU	\$02F4	;1-byte CHBASE shadow
NEWROW EQU	\$02F5	;1-byte draw destination row
NEWCOL EQU	\$02F6	;2-byte draw destination column
ROWINC EQU	\$02F8	;1-byte draw row increment
COLINC EQU	\$02F9	;1-byte draw column increment
CHAR EQU	\$02FA	;1-byte internal character
ATACHR EQU	\$02FB	;1-byte ATASCII character or plot point
CH EQU	\$02FC	;1-byte keyboard code (buffer)
FILDAT EQU	\$02FD	;1-byte right fill data
DSPFLG EQU	\$02FE	;1-byte control character display flag (0 = no)
SSFLAG EQU	\$02FF	;1-byte start/stop flag (0 = not stopped)

** Page Three Address Equates

DCB EQU	\$0300	;12-byte device control block
DDEVIC EQU	\$0300	;1-byte unit 1 bus ID
DUNIT EQU	\$0301	;1-byte unit number
DCOMND EQU	\$0302	;1-byte bus command
DSTATS EQU	\$0303	;1-byte command type/status return
DBUFLO EQU	\$0304	;1-byte low data buffer address
DBUFHI EQU	\$0305	;1-byte high data buffer address
DTIMLO EQU	\$0306	;1-byte timeout (seconds)
DUNUSE EQU	\$0307	;1-byte (not used)
DBYTLO EQU	\$0308	;1-byte low number of bytes to transfer
DBYTHI EQU	\$0309	;1-byte high number of bytes to transfer
DAUX1 EQU	\$030A	;1-byte first command auxiliary
DAUX2 EQU	\$030B	;1-byte second command auxiliary
TIMER1 EQU	\$030C	;2-byte initial baud rate timer value
JMPERS EQU	\$030E	;1-byte jumper options
CASFLG EQU	\$030F	;1-byte cassette I/O flag (0 = not cassette I/O)
TIMER2 EQU	\$0310	;2-byte final baud rate timer value
TEMP1 EQU	\$0312	;2-byte temporary
TEMP2 EQU	\$0313	;1-byte temporary
PTIMOT EQU	\$0314	;1-byte printer timeout
TEMP3 EQU	\$0315	;1-byte temporary
SAVIO EQU	\$0316	;1-byte saved serial data input indicator
TIMFLG EQU	\$0317	;1-byte timeout flag (0 = timeout)

```

STACKP EQU    $0318 ;1-byte SI0 saved stack pointer
TSTAT  EQU    $0319 ;1-byte temporary status

HATABS EQU    $031A ;35-byte handler address table

PUPBT1 EQU    $033D ;1-byte power-up validation byte 1
PUPBT2 EQU    $033E ;1-byte power-up validation byte 2
PUPBT3 EQU    $033F ;1-byte power-up validation byte 3

IOCB   EQU    $0340 ;128-byte I/O control blocks area
ICHID  EQU    $0340 ;1-byte handler ID ($FF = free)
ICDNO  EQU    $0341 ;1-byte device number
ICCOM  EQU    $0342 ;1-byte command code
ICSTA  EQU    $0343 ;1-byte status of last action
ICBAL  EQU    $0344 ;1-byte low buffer address
ICBAH  EQU    $0345 ;1-byte high buffer address
ICPTL  EQU    $0346 ;1-byte low PUT-BYTE routine address-1
ICPTH  EQU    $0347 ;1-byte high PUT-BYTE routine address-1
ICBL  EQU    $0348 ;1-byte low buffer length
ICBLH  EQU    $0349 ;1-byte high buffer length
ICAX1  EQU    $034A ;1-byte first auxiliary information
ICAX2  EQU    $034B ;1-byte second auxiliary information
ICSPR  EQU    $034C ;4-byte work area

PRNBUF EQU    $03C0 ;40-byte printer buffer
SUPERF EQU    $03E8 ;1-byte editor super function flag (0 = not)
CKEY   EQU    $03E9 ;1-byte cassette boot request flag (0 = not)
CASSBT EQU    $03EA ;1-byte cassette boot flag (0 = not)
CARTCK EQU    $03EB ;1-byte cartridge equivalence checksum
DERRF  EQU    $03EC ;1-byte screen OPEN error flag (0 = not)

;      Remainder of Page Three Not Cleared upon Reset

ACMVAR EQU    $03ED ;11 bytes reserved for ACMI
BASICF EQU    $03F8 ;1-byte BASIC switch flag (0 = BASIC enabled)
MINTLK EQU    $03F9 ;1-byte ACMI module interlock
GINTLK EQU    $03FA ;1-byte cartridge interlock
CHLINK EQU    $03FB ;2-byte loaded handler chain link
CASBUF EQU    $03FD ;3-byte first 3 bytes of cassette buffer

```

** Page Four Address Equates

```

;      EQU    $0400 ;128-byte remainder of cassette buffer

;      Reserved for Application

USAREA EQU    $0480 ;128 bytes reserved for application

```

** Page Five Address Equates

```

;      Reserved for Application and Floating Point Package

;      EQU    $0500 ;256 bytes reserved for application and FPP

```

** Floating Point Package Address Equates

```

LBPR1 EQU    $057E ;1-byte LBUFF preamble
LBPR2 EQU    $057F ;1-byte LBUFF preamble
LBUFF EQU    $0580 ;128-byte line buffer

PLYARG EQU    $05E0 ;6-byte floating point polynomial argument
FPSCR  EQU    $05E6 ;6-byte floating point temporary
FPSCR1 EQU    $05EC ;6-byte floating point temporary

```


** Page Six Address Equates

```
;      Reserved for Application  
;  
;      EQU      $0600 ;256 bytes reserved for application
```

** LNBUG Address Equates

```
LNBUG IF      LNBUG  
LNORG EQU     $6000 ;LNBUG origin  
LNIRQ EQU     $6033 ;LNBUG IRQ entry  
LNNMI EQU     $8351 ;LNBUG NMI vector  
LNBUG ENDIF
```

** Cartridge Address Equates

```
CARTCS EQU     $BFFA ;2-byte cartridge coldstart address  
CART  EQU     $BFFC ;1-byte cartridge present indicator  
CARTFG EQU     $BFFD ;1-byte cartridge flags  
CARTAD EQU     $BFFE ;2-byte cartridge start vector
```

** CTIA/GTIA Address Equates

```
CTIA  EQU     $D000 ;CTIA/GTIA area  
  
;      Read/Write Addresses  
  
CONSOL EQU     $D01F ;console switches and speaker control  
  
;      Read Addresses  
  
M0PF  EQU     $D000 ;missile 0 and playfield collision  
M1PF  EQU     $D001 ;missile 1 and playfield collision  
M2PF  EQU     $D002 ;missile 2 and playfield collision  
M3PF  EQU     $D003 ;missile 3 and playfield collision  
  
P0PF  EQU     $D004 ;player 0 and playfield collision  
P1PF  EQU     $D005 ;player 1 and playfield collision  
P2PF  EQU     $D006 ;player 2 and playfield collision  
P3PF  EQU     $D007 ;player 3 and playfield collision  
  
M0PL  EQU     $D008 ;missile 0 and player collision  
M1PL  EQU     $D009 ;missile 1 and player collision  
M2PL  EQU     $D00A ;missile 2 and player collision  
M3PL  EQU     $D00B ;missile 3 and player collision  
  
P0PL  EQU     $D00C ;player 0 and player collision  
P1PL  EQU     $D00D ;player 1 and player collision  
P2PL  EQU     $D00E ;player 2 and player collision  
P3PL  EQU     $D00F ;player 3 and player collision  
  
TRIG0 EQU     $D010 ;joystick trigger 0  
TRIG1 EQU     $D011 ;joystick trigger 1  
  
TRIG2 EQU     $D012 ;cartridge interlock  
TRIG3 EQU     $D013 ;ACMI module interlock  
  
PAL   EQU     $D014 ;PAL/NTSC indicator  
  
;      Write Addresses
```

```

HPOSP0 EQU    $D000 ;player 0 horizontal position
HPOSP1 EQU    $D001 ;player 1 horizontal position
HPOSP2 EQU    $D002 ;player 2 horizontal position
HPOSP3 EQU    $D003 ;player 3 horizontal position

HPOSM0 EQU    $D004 ;missile 0 horizontal position
HPOSM1 EQU    $D005 ;missile 1 horizontal position
HPOSM2 EQU    $D006 ;missile 2 horizontal position
HPOSM3 EQU    $D007 ;missile 3 horizontal position

SIZEP0 EQU    $D008 ;player 0 size
SIZEP1 EQU    $D009 ;player 1 size
SIZEP2 EQU    $D00A ;player 2 size
SIZEP3 EQU    $D00B ;player 3 size

SIZEM EQU     $D00C ;missile sizes

GRAFP0 EQU    $D00D ;player 0 graphics
GRAFP1 EQU    $D00E ;player 1 graphics
GRAFP2 EQU    $D00F ;player 2 graphics
GRAFP3 EQU    $D010 ;player 3 graphics

GRAFM EQU     $D011 ;missile graphics

COLPM0 EQU    $D012 ;player-missile 0 color/luminance
COLPM1 EQU    $D013 ;player-missile 1 color/luminance
COLPM2 EQU    $D014 ;player-missile 2 color/luminance
COLPM3 EQU    $D015 ;player-missile 3 color/luminance

COLPF0 EQU    $D016 ;playfield 0 color/luminance
COLPF1 EQU    $D017 ;playfield 1 color/luminance
COLPF2 EQU    $D018 ;playfield 2 color/luminance
COLPF3 EQU    $D019 ;playfield 3 color/luminance

COLBK EQU     $D01A ;background color/luminance

PRIOR EQU     $D01B ;priority select
VDELAY EQU    $D01C ;vertical delay
GRCTL EQU     $D01D ;graphic control
HITCLR EQU    $D01E ;collision clear

```

** PBI Address Equates

```

PBI EQU       $D100 ;parallel bus interface area

; Read Addresses

PDVI EQU      $D1FF ;parallel device IRQ status

; Write Addresses

PDVS EQU      $D1FF ;parallel device select

```

** POKEY Address Equates

```

POKEY EQU     $D200 ;POKEY area

; Read Addresses

POT0 EQU      $D200 ;potentiometer 0
POT1 EQU      $D201 ;potentiometer 1
POT2 EQU      $D202 ;potentiometer 2
POT3 EQU      $D203 ;potentiometer 3
POT4 EQU      $D204 ;potentiometer 4

```

```

POT5 EQU $D205 ;potentiometer 5
POT6 EQU $D206 ;potentiometer 6
POT7 EQU $D207 ;potentiometer 7

ALLPOT EQU $D208 ;potentiometer port state
KBCODE EQU $D209 ;keyboard code
RANDOM EQU $D20A ;random number generator
SERIN EQU $D20D ;serial port input
IRQST EQU $D20E ;IRQ interrupt status
SKSTAT EQU $D20F ;serial port and keyboard status

; Write Addresses

AUDF1 EQU $D200 ;channel 1 audio frequency
AUDC1 EQU $D201 ;channel 1 audio control

AUDF2 EQU $D202 ;channel 2 audio frequency
AUDC2 EQU $D203 ;channel 2 audio control

AUDF3 EQU $D204 ;channel 3 audio frequency
AUDC3 EQU $D205 ;channel 3 audio control

AUDF4 EQU $D206 ;channel 4 audio frequency
AUDC4 EQU $D207 ;channel 4 audio control

AUDCTL EQU $D208 ;audio control
STIMER EQU $D209 ;start timers
SKRES EQU $D20A ;reset SKSTAT status
POTGO EQU $D20B ;start potentiometer scan sequence
SEROUT EQU $D20D ;serial port output
IRQEN EQU $D20E ;IRQ interrupt enable
SKCTL EQU $D20F ;serial port and keyboard control

```

** PIA Address Equates

```

PIA EQU $D300 ;PIA area

; Read/Write Addresses

PORTA EQU $D300 ;port A direction register or jacks 0 and 1
PORTB EQU $D301 ;port B direction register or memory control

PACTL EQU $D302 ;port A control
PBCTL EQU $D303 ;port B control

```

** ANTIC Address Equates

```

ANTIC EQU $D400 ;ANTIC area

; Read Addresses

VCOUNT EQU $D40B ;vertical line counter
PENH EQU $D40C ;light pen horizontal position
PENV EQU $D40D ;light pen vertical position
NMIST EQU $D40F ;NMI interrupt status

; Write Addresses

DMACTL EQU $D400 ;DMA control
CHACTL EQU $D401 ;character control
DLISTL EQU $D402 ;low display list address
DLISTH EQU $D403 ;high display list address
HSCROL EQU $D404 ;horizontal scroll
VSCROL EQU $D405 ;vertical scroll

```

```

PMBASE EQU    $D407 ;player-missile base address
CHBASE EQU    $D409 ;character base address
WSYNC EQU     $D40A ;wait for HBLANK synchronization
NMIIEN EQU    $D40E ;NMI enable
NMIIRES EQU   $D40F ;NMI interrupt status reset

```

** PBI RAM Address Equates

```

PBIRAM EQU    $D600 ;parallel bus interface RAM area

```

** ACMI Address Equates

```

ACMI  IF      ACMI
AMVTAD EQU    $D7EA ;vector table
AMINIT EQU    $D7F6 ;initialization address
AMNAME EQU    $D7F9 ;ACMI device code
AMISRA EQU    $D7FA ;interrupt service routine address
AMFLAG EQU    $D7FF ;ACMI flags
ACMI  ENDIF

```

** Floating Point Package Address Equates

```

AFP      EQU    $D800 ;convert ASCII to floating point
FASC     EQU    $D8E6 ;convert floating point to ASCII
IFP      EQU    $D9AA ;convert integer to floating point
FPI      EQU    $D9D2 ;convert floating point to integer
ZFR0     EQU    $DA44 ;zero FR0
ZF1      EQU    $DA46 ;zero floating point number
FSUB     EQU    $DA60 ;subtract floating point numbers
FADD     EQU    $DA66 ;add floating point numbers
FMUL     EQU    $DADB ;multiply floating point numbers
FDIV     EQU    $DB28 ;divide floating point numbers
PLYEVL   EQU    $DD40 ;evaluate floating point polynomial
FLD0R    EQU    $DD89 ;load floating point number
FLD0P    EQU    $DD8D ;load floating point number
FLD1R    EQU    $DD98 ;load floating point number
FLD1P    EQU    $DD9C ;load floating point number
FST0R    EQU    $DDA7 ;store floating point number
FST0P    EQU    $DDAB ;store floating point number
FMOVE    EQU    $ddb6 ;move floating point number
LOG       EQU    $DECD ;calculate floating point logarithm
LOG10    EQU    $DED1 ;calculate floating point base 10 logarithm
EXP       EQU    $DDC0 ;calculate floating point exponentiation
EXP10    EQU    $DDCC ;calculate floating point base 10 exponentiation

```

** Parallel Device Address Equates

```

PDID1 EQU    $D803 ;parallel device ID 1
PDIOV EQU    $D805 ;parallel device I/O vector
PDIRQV EQU    $D808 ;parallel device IRQ vector
PDID2 EQU    $D80B ;parallel device ID 2
PDVV  EQU    $D80D ;parallel device vector table

```

** Device Handler Vector Table Address Equates

```

EDITRV EQU    $E400 ;editor handler vector table
SCRENV EQU    $E410 ;screen handler vector table
KEYBDV EQU    $E420 ;keyboard handler vector table
PRINTV EQU    $E430 ;printer handler vector table
CASETV EQU    $E440 ;cassette handler vector table

```

**** Jump Vector Address Equates**

DINITV EQU	\$E450	;vector to initialize DIO
DSKINV EQU	\$E453	;vector to DIO
CIOV EQU	\$E456	;vector to CIO
SIOV EQU	\$E459	;vector to SIO
SETVBV EQU	\$E45C	;vector to set VBLANK parameters
SYSVBV EQU	\$E45F	;vector to process immediate VBLANK NMI
XITVBV EQU	\$E462	;vector to process deferred VBLANK NMI
SIOINV EQU	\$E465	;vector to initialize SIO
SENDEV EQU	\$E468	;vector to enable SEND
INTINV EQU	\$E46B	;vector to initialize interrupt handler
CIOINV EQU	\$E46E	;vector to initialize CIO
BLKBDV EQU	\$E471	;vector to power-up display (formerly memo pad)
WARMSV EQU	\$E474	;vector to warmstart
COLDSV EQU	\$E477	;vector to coldstart
RBLOKV EQU	\$E47A	;vector to read cassette block
CSOPIV EQU	\$E47D	;vector to open cassette for input
PUPDIV EQU	\$E480	;vector to power-up display
SLFTSV EQU	\$E483	;vector to self-test
PHENTV EQU	\$E486	;vector to enter peripheral handler
PHUNLV EQU	\$E489	;vector to unlink peripheral handler
PHINIV EQU	\$E48C	;vector to initialize peripheral handler

**** Generic Parallel Device Handler Vector Table Address Equates**

GPDVV EQU	\$E48F	;generic parallel device handler vector table
-----------	--------	---

Miscellaneous Address Equates

** Self-test Page Zero Address Equates

STTIME EQU	\$0080	;2-byte main screen timeout timer
STAUT EQU	\$0082	;1-byte auto-mode flag
STJMP EQU	\$0083	;3-byte ANTIC jump instruction
STSEL EQU	\$0086	;1-byte selection
STPASS EQU	\$0087	;1-byte pass
STSPP EQU	\$0088	;1-byte SELECT previously pressed flag
;	\$0089	;1-byte (not used)
STKST EQU	\$008A	;1-byte keyboard self-test flag (0 = not)
STCHK EQU	\$008B	;2-byte checksum
STSMM EQU	\$008D	;1-byte screen memory mask
STSMP EQU	\$008E	;1-byte screen memory pointer
ST1K EQU	\$008F	;1-byte current 1K of memory to test
STPAG EQU	\$0090	;2-byte current page to test
STPC EQU	\$0092	;1-byte page count
STMVAL EQU	\$0093	;1-byte correct value for memory test
STSKP EQU	\$0094	;1-byte simulated keypress index
STTMP1 EQU	\$0095	;2-byte temporary
STVOC EQU	\$0097	;1-byte current voice indicator
STNOT EQU	\$0098	;1-byte current note counter
STCDI EQU	\$0099	;1-byte cleft display pointer
STCDA EQU	\$009A	;1-byte cleft data pointer
STTMP2 EQU	\$009B	;2-byte temporary
STTMP3 EQU	\$009D	;1-byte temporary
STADR1 EQU	\$009E	;2-byte temporary address
STADR2 EQU	\$00A0	;2-byte temporary address
STBL EQU	\$00A2	;1-byte blink counter
STTMP4 EQU	\$00A3	;1-byte temporary
STLM EQU	\$00A4	;1-byte LED mask
STTMP5 EQU	\$00A5	;1-byte temporary

** Self-test Address Equates

ST3000 EQU	\$3000	;screen memory
ST3002 EQU	\$3002	;cleft display
ST3004 EQU	\$3004	;"VOICE #" text display
ST300B EQU	\$300B	;voice number display
ST301C EQU	\$301C	;START key display
ST301E EQU	\$301E	;SELECT key display
ST3020 EQU	\$3020	;OPTION key display, first 8K ROM display
ST3021 EQU	\$3021	;keyboard character display
ST3022 EQU	\$3022	;keyboard text display
ST3024 EQU	\$3024	;second 8K ROM display
ST3028 EQU	\$3028	;"RAM" text display
ST3038 EQU	\$3038	;RAM display
ST303C EQU	\$303C	;fifth note display
ST304C EQU	\$304C	;"B S" text display
ST3052 EQU	\$3052	;tab key display
ST3062 EQU	\$3062	;cleft display
ST306D EQU	\$306D	;return key display
ST3072 EQU	\$3072	;control key display
ST3092 EQU	\$3092	;"SH" text display
ST309E EQU	\$309E	;sixth note display
ST30AB EQU	\$30AB	;"SH" text display
ST30B7 EQU	\$30B7	;"S P A C E B A R" text display
ST30C1 EQU	\$30C1	;cleft display
ST30C2 EQU	\$30C2	;cleft display
ST30C7 EQU	\$30C7	;third note display
ST30CA EQU	\$30CA	;fourth note display
ST30F8 EQU	\$30F8	;third note display

ST3100 EQU	\$3100	;screen memory
ST3121 EQU	\$3121	;cleft display
ST3122 EQU	\$3122	;cleft display
ST313C EQU	\$313C	;fifth note display
ST3150 EQU	\$3150	;first line of staff display
ST3154 EQU	\$3154	;first note display
ST3181 EQU	\$3181	;cleft display
ST3182 EQU	\$3182	;cleft display
ST3186 EQU	\$3186	;second note display
ST318C EQU	\$318C	;fifth note display
ST31B0 EQU	\$31B0	;second line of staff display
ST31C2 EQU	\$31C2	;cleft display
ST31CA EQU	\$31CA	;fourth note display
ST31EE EQU	\$31EE	;sixth note display
ST31F1 EQU	\$31F1	;cleft display
ST3210 EQU	\$3210	;third line of staff display
ST321A EQU	\$321A	;fourth note display
ST3248 EQU	\$3248	;third note display
ST3270 EQU	\$3270	;fourth line of staff display
ST32D0 EQU	\$32D0	;fifth line of staff display

Macro Definitions

```
**      FIX - Fix Address
*
*      FIX sets the origin counter to the value specified as an
*      argument.  If the current origin counter is less than the
*      argument, FIX fills the intervening bytes with zero and
*      issues a message to document the location and number of
*      bytes that are zero filled.
*
*      ENTRY  FIX      address
*
*
*      EXIT
*          Origin counter set to specified address.
*          Message issued if zero fill required.
*
*      CHANGES
*          -none-
*
*      CALLS
*          -none-
*
*      NOTES
*          Due to ECHO limitation of 255 iterations, FIX is
*          recursive.
*          If the current origin counter value is beyond the
*          argument, FIX generates an error.
*
*      MODS
*          R. K. Nordin 1983-11-01
```

```
FIX      MACRO  address
IF      %1 <> *0
IF      %1 > *0
IF      %1 - *0 < 256
MSG     '$',%1 - *0,' free bytes from $',*0,' to $',%1 - 1
ECHO    %1 - *0
DB      0
ENDM
ELSE
FIX     *0 + 255
FIX     %1
ENDIF
ELSE
ERR      ;%1 precedes current origin counter value
ENDIF
ENDIF
ORG      *0
ENDM
```

First 8K ROM Identification and Checksum

ORG \$C000

** First 8K ROM Identification and Checksum
--

DW	\$0000		;reserved for checksum
DB	IDDAY, IDMON, IDYEAR		;date (day, month, year)
DB	\$00		;not used
DB	IDPN1, IDPN2, IDPN3, IDPN4, IDPN5		;part number
DB	IDREV		;revision number

Interrupt Handler

```
**      IIH - Initialize Interrupt Handler
*
*      ENTRY JSR      IIH
*              TRIG3 = ACMI module interlock
*              TRIG2 = cartridge interlock
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01
```

```
IIH      =      *      ;entry

      LDA      #$40
      STA      NMIIEN ;disable DLI and enable VBLANK NMI

      LDA      TRIG3 ;cartridge interlock
      STA      GINTLK ;cartridge interlock status
```

```
ACMI     IF      ACMI
      LDA      TRIG2 ;ACMI module interlock
      STA      MINTLK ;ACMI module interlock status
ACMI     ENDIF
```

```
      RTS      ;return
```

```
**      NMI - Process NMI
*
*      ENTRY JMP      NMI
*
*      EXIT
*              Exits via appropriate vector to process NMI
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01
```

```
NMI      =      *      ;entry

      ASSERT $C0=high NMI ;for compatibility with LNBUG

;      Check for display list NMI.

      BIT      NMIST
      BPL      NMI1      ;if not display list NMI

      JMP      (VDSLST) ;process display list NMI, return

;      Initialize.

NMI1     CLD

;      Save registers.

      PHA      ;save A
      TXA
      PHA      ;save X
      TYA
      PHA      ;save Y
```

```
;      Reset NMI status.

      STA      NMIRES      ;reset NMI status

;      Process NMI.
```

```
LNBUG IF      LNBUG
      LDA      LNFLG      ;LNBUG flag
      BNE      NMI2        ;if LNBUG

      JMP      (VVBKLI)    ;process immediate VBLANK NMI, return

NMI2  JMP      (LNNMI)     ;invoke LNBUG NMI routine, return
LNBUG ELSE
      JMP      (VVBKLI)    ;process immediate VBLANK NMI, return
LNBUG ENDIF
```

```
**      IRQ - Process IRQ
*
*      ENTRY  JMP      IRQ
*
*      EXIT
*          Exits via VIMIRQ vector
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
IRQ    =      *          ;entry

;      Initialize.

      CLD

;      Process IRQ.
```

```
LNBUG IF      LNBUG
      BIT      LNFLG
      BMI      IRQ1        ;if LNBUG on

      JMP      (VIMIRQ)    ;process immediate IRQ, return

IRQ1  JMP      LNIRQ       ;invoke LNBUG IRQ routine, return
LNBUG ELSE
      JMP      (VIMIRQ)    ;process immediate IRQ, return
LNBUG ENDIF
```

```
**      IIR - Process Immediate IRQ
*
*      ENTRY  JMP      IIR
*
*      EXIT
*          Exits via appropriate vector to process IRQ
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
IIR    =      *          ;entry

;      Initialize.
```

```

        PHA                ;save A
;    Check for serial input ready IRQ.

        LDA    IRQST        ;IRQ status
        AND    #$20         ;serial input ready
        BNE    IIR1         ;if not serial input ready
;    Process serial input IRQ.

        LDA    #not $20     ;all other interrupts
        STA    IRQEN        ;enable all other interrupts
        LDA    POKMSK
        STA    IRQEN
        JMP    (VSERIN)     ;process serial input ready IRQ, return
;    Process possible ACMI IRQ.

IIR1
ACMI    IF    ACMI
        JSR    AIR          ;process possible ACMI IRQ
        BCS    BIR1        ;if interrupt processed, return
ACMI    ENDIF
;    Initialize further.

        TXA
        PHA                ;save X
;    Check for parallel device IRQ.

        LDA    PDVI         ;parallel device IRQ statuses
        AND    PDIMSK       ;select desired IRQ statuses
        BEQ    IIR2         ;if no desired IRQ
;    Process parallel device IRQ.

        JMP    (VPIRQ)      ;process parallel device IRQ, return
;    Check other types of IRQ.

IIR2    LDX    #TIRQL-1-1   ;offset to next to last entry
IIR3    LDA    TIRQ,X       ;IRQ type
        CPX    #5           ;offset to serial out complete
        BNE    IIR4         ;if not serial out complete

        AND    POKMSK       ;and with POKEY IRQ enable
        BEQ    IIR5         ;if serial out complete not enabled
IIR4    BIT    IRQST        ;IRQ interrupt status
        BEQ    IIR6         ;if interrupt found
IIR5    DEX
        BPL    IIR3         ;if not done
;    Continue IRQ processing.

        JMP    CIR          ;continue IRQ processing, return
;    Enable other interrupts.

IIR6    EOR    #$FF         ;complement mask
        STA    IRQEN        ;enable all others
        LDA    POKMSK       ;POKEY IRQ mask
        STA    IRQEN        ;enable indicated IRQ's

```

```

;      Check for BREAK key IRQ.

      CPX      #0
      BNE      IIR7          ;if not BREAK key IRQ

;      Check for keyboard disabled.

      LDA      KEYDIS
      BNE      CIR          ;if keyboard disabled, continue, return

;      Process IRQ.

IIR7  LDA      TOIH,X        ;offset to interrupt handler
      TAX
      LDA      INTABS,X      ;interrupt handler address
      STA      JVECK
      LDA      INTABS+1,X
      STA      JVECK+1
      PLA
      TAX                ;restore X
      JMP      (JVECK)       ;process interrupt, return

```

```

**      BIR - Process BREAK Key IRQ
*
*      ENTRY  JMP      BIR
*
*      EXIT
*          Exits via RTI
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

BIR    =      *          ;entry

;      Process BREAK.

      LDA      #0
      STA      BRKKEY ;clear BREAK key flag
      STA      SSFLAG ;clear start/stop flag
      STA      CRSINH ;enable cursor
      STA      ATTRACT ;turn off attract-mode

;      Exit.

BIR1   PLA          ;restore A
      RTI           ;return

```

```

**      CIR - Continue IRQ Processing
*
*      ENTRY  JMP      CIR
*
*      EXIT
*          Exits via appropriate vector to process IRQ or to XIR
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CIR    =      *          ;entry

;      Initialize.

```

```

    PLA
    TAX

;    Check for port A interrupt.

    BIT    PACTL        ;port A control
    BPL    CIR1          ;if not port A interrupt

;    Process proceed line IRQ.

    LDA    PORTA        ;clear interrupt status bit
    JMP    (VPRCED)     ;process proceed line IRQ, return

;    Check for port B interrupt.

CIR1  BIT    PBCTL        ;port B control
      BPL    CIR2          ;if not port B interrupt

;    Process interrupt line IRQ.

    LDA    PORTB        ;clear interrupt status bit
    JMP    (VINTER)     ;process interrupt line IRQ, return

;    Check for BRK instruction IRQ.

CIR2  PLA
      STA    JVECK

      PLA                ;saved P
      PHA                ;resave P
      AND    #$10        ;B bit of P register
      BEQ    CIR3          ;if not BRK instruction IRQ

;    Process BRK instruction IRQ.

    LDA    JVECK
    PHA
    JMP    (VBREAK)      ;process BRK instruction IRQ, return

;    Exit IRQ processing.

CIR3  LDA    JVECK
      PHA
      JMP    XIR          ;exit IRQ processing, return

```

```

**    XIR - Exit IRQ Processing
*
*    ENTRY  JMP    XIR
*
*    EXIT
*          Exits to RIR
*
*    MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

XIR   =    *    ;entry
      PLA      ;restore A
;      JMP    RIR    ;return from interrupt

```

```

**      RIR - Return from Interrupt
*
*      ENTRY JMP      RIR
*
*      EXIT
*          Exits via RTI
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

RIR      =      *      ;entry
          RTI      ;return

```

```

**      AIR - Process ACMI IRQ
*
*      ENTRY JSR      AIR
*
*      EXIT  Exits via ACMISR vector
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

ACMI     IF      ACMI
AIR      =      *      ;entry
          JMP      (ACMISR) ;process ACMI interrupt, return
ACMI     ENDIF

```

```

**      TIRQ - Table of IRQ Types
*
*      Entry n is the interrupt indicator of priority n (0 is lowest).
*
*      NOTES
*          Problem: entry 7 (serial input ready) not used.

```

```

TIRQ     DB      $80      ;0 - BREAK key IRQ
          DB      $40      ;1 - keyboard IRQ
          DB      $04      ;2 - timer 4 IRQ
          DB      $02      ;3 - timer 2 IRQ
          DB      $01      ;4 - timer 1 IRQ
          DB      $08      ;5 - serial output complete IRQ
          DB      $10      ;6 - serial output ready IRQ
          DB      $20      ;7 - serial input ready IRQ

```

```

TIRQL    =      *-TIRQ ;length

```

```

**      TOIH - Table of Offsets to Interrupt Handlers
*
*      Entry n is the offset to the interrupt handler vector
*      corresponding to entry n of TIRQ.
*
*      NOTES
*          Problem: entry 7 (serial input ready) not used.

```

```

TOIH     DB      BRKKY-INTABS ;0 - BREAK key IRQ
          DB      VKEYBD-INTABS;1 - keyboard IRQ
          DB      VTIMR4-INTABS;2 - timer 4 IRQ
          DB      VTIMR2-INTABS;3 - timer 2 IRQ
          DB      VTIMR1-INTABS;4 - timer 1 IRQ

```

```

DB    VSEROC-INTABS;5 - serial output complete IRQ
DB    VSEROR-INTABS;6 - serial output ready IRQ
DB    VSERIN-INTABS;7 - serial input ready IRQ

```

```

**    WFR - Wait for RESET
*
*    WFR loops forever.
*
*    ENTRY  JMP    WFR
*
*    EXIT
*          Does not exit
*
*    MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

WFR    =    *    ;entry
;      Loop forever, waiting for RESET.
WFR1   JMP    WFR1 ;loop

```

```

**    IVNM - Process Immediate VBLANK NMI
*
*    ENTRY  JMP    IVNM
*
*    EXIT
*          Exits to DVNM or via VVBLKD vector
*
*    MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

IVNM    =    *    ;entry
;      Increment frame counter and attract-mode counter.

      INC    RTCLOK+2    ;increment low frame counter
      BNE    IVN1        ;if low counter not zero

      INC    ATTRACT      ;increment attract-mode counter/flag
      INC    RTCLOK+1    ;increment middle frame counter
      BNE    IVN1        ;if middle counter not zero

      INC    RTCLOK      ;increment high frame counter

;      Set attract-mode effects.

IVN1    LDA    #$FE      ;select no luminance change
      LDX    #0          ;select no color shift
      LDY    ATTRACT      ;attract-mode timer/flag
      BPL    IVN2        ;if not attract-mode

      STA    ATTRACT      ;ensure continued attract-mode
      LDX    RTCLOK+1    ;select color shift
      LDA    #$F6        ;select lower luminance

IVN2    STA    DRKMSK      ;attract-mode luminance
      STX    COLRSH      ;attract-mode color shift

;      Update COLPF1 (in case fine scrolling and critical section).

```



```

    LDA    COLOR1        ;playfield 1 color
    EOR    COLRSH        ;modify color with attract-mode color shift
    AND    DRKMSK        ;modify with attract-mode luminance
    STA    COLPF1        ;set playfield 1 color/luminance

;    Process countdown timer 1.

    LDX    #0            ;indicate countdown timer 1
    JSR    DCT            ;decrement countdown timer
    BNE    IVN3          ;if timer not expired

    JSR    PTO            ;process countdown timer 1 expiration

;    Check for critical section.

IVN3    LDA    CRITIC
    BNE    IVN4          ;if critical section

;    Check for IRQ enabled.

    TSX     ;stack pointer
    LDA    $0104,X      ;stacked P
    AND    #$04          ;I (IRQ disable) bit
    BEQ    IVN5          ;if IRQ enabled

;    Exit.

IVN4    JMP    DVNM      ;process deferred VBLANK NMI, return

;    Process IRQ enabled non-critical section.

IVN5

;    Check for ACMI module change.

ACMI    IF    ACMI
    LDA    TRIG2        ;ACMI module interlock
    CMP    MINTLK       ;previous ACMI module interlock status
    BNE    WFR          ;if ACMI module change, wait for RESET
ACMI    ENDIF

;    Check for cartridge change.

    LDA    TRIG3        ;cartridge interlock
    CMP    GINTLK       ;previous cartridge interlock status
    BNE    WFR          ;if cartridge change, wait for RESET

;    Set hardware registers from shadows.

    LDA    PENV
    STA    LPENV        ;light pen vertical position
    LDA    PENH
    STA    LPENH        ;light pen vertical position
    LDA    SDLSTH
    STA    DLISTH       ;high display list address
    LDA    SDLSTL
    STA    DLISTL       ;low display list address
    LDA    SDMCTL
    STA    DMACTL       ;DMA control
    LDA    GPRIOR
    STA    PRIOR        ;priority select

;    Check for vertical scroll enabled.

    LDA    VSFLAG        ;vertical scroll count
    BEQ    IVN6          ;if vertical scroll not enabled

```

```

; Scroll one line.
    DEC    VSFLAG    ;decrement vertical scroll count
    LDA    #8        ;scroll one line
    SEC
    SBC    VSFLAG    ;subtract vertical scroll count
    AND    #07
    STA    VSCROL    ;set vertical scroll

; Turn off speaker.
IVN6  LDX    #$08    ;speaker off
      STX    CONSOL  ;set speaker control

; Set color registers from shadows.
; LDX    #8        ;offset to background color
IVN7  CLI
      LDA    PCOLR0,X ;color register shadow
      EOR    COLRSH   ;modify with attract-mode color shift
      AND    DRKMSK   ;modify with attract-mode luminance
      STA    COLPM0,X ;set color register
      DEX
      BPL    IVN7     ;if not done

; Set character set control.
      LDA    CHBAS
      STA    CHBASE
      LDA    CHACT
      STA    CHACTL

; Process countdown timer 2.
      LDX    #2        ;indicate countdown timer 2
      JSR    DCT        ;decrement countdown timer
      BNE    IVN8       ;if timer not expired

      JSR    PTT        ;process countdown timer 2 expiration

; Process timers 3, 4 and 5.
IVN8  LDX    #2        ;preset offset to timer 2
IVN9  INX
      INX                ;offset to countdown timer
      LDA    CDTMV3-4,X  ;countdown timer
      ORA    CDTMV3+1-4,X
      BEQ    IVN10       ;if countdown timer already expired

      JSR    DCT        ;decrement countdown timer
      STA    CDTMF3-4,X  ;indicate timer expiration status
IVN10 CPX    #8        ;offset to timer 5
      BNE    IVN9       ;if all timers not done

; Check debounce counter.
      LDA    SKSTAT     ;keyboard status
      AND    #$04       ;key down indicator
      BEQ    IVN11      ;if key down

; Process key up.
      LDA    KEYDEL     ;key delay counter

```

```

    BEQ     IVN11          ;if counted down already
    DEC     KEYDEL         ;decrement key delay counter
;    Check software key repeat timer.
IVN11 LDA     SRTIMR       ;key repeat timer
    BEQ     IVN13         ;if key repeat timer expired

    LDA     SKSTAT        ;keyboard status
    AND     #$04          ;key down indicator
    BNE     IVN12         ;if key no longer down

    DEC     SRTIMR        ;decrement key repeat timer
    BNE     IVN13         ;if key repeat timer not expired
;    Process key repeat timer expiration.

    LDA     KEYDIS        ;keyboard disable flag
    BNE     IVN13         ;if keyboard disabled, no repeat

    LDA     KEYREP        ;initial timer value
    STA     SRTIMR        ;reset key repeat timer
    LDA     KBCODE        ;key code
;    Check for hidden codes.

    CMP     #CNTL1
    BEQ     IVN13         ;if CTRL-1

    CMP     #CNTLF1
    BEQ     IVN13         ;if CTRL-F1

    CMP     #CNTLF2
    BEQ     IVN13         ;if CTRL-F2

    CMP     #CNTLF4
    BEQ     IVN13         ;if CTRL-F4

    AND     #$3F
    CMP     #HELP
    BEQ     IVN13         ;if HELP
;    Set key code.

    LDA     KBCODE        ;key code
    STA     CH            ;set key code
    JMP     IVN13         ;continue
;    Zero key repeat timer.
IVN12 LDA     #0
    STA     SRTIMR        ;zero key repeat timer
;    Read joysticks.
IVN13 LDA     PORTA       ;joystick readings
    LSR     A
    LSR     A
    LSR     A
    LSR     A             ;joystick 1 reading
    STA     STICK1        ;set joystick 1 reading
VGC IF     VGC
VGC STA     STICK3        ;simulate joystick 3 reading
VGC ENDF
    LDA     PORTA       ;joystick readings
    AND     #$0F        ;joystick 0 reading

```

```

        STA  STICK0      ;set joystick 0 reading
VGC    IF    VGC
        STA  STICK2      ;simulate joystick 2 reading
VGC    ENDIF

;      Read joystick triggers.

        LDA  TRIG0      ;trigger 0 indicator
        STA  STRIG0     ;set trigger 0 indicator
VGC    IF    VGC
        STA  STRIG2     ;simulate trigger 2 indicator
VGC    ENDIF
        LDA  TRIG1      ;trigger 1 indicator
        STA  STRIG1     ;set trigger 1 indicator
VGC    IF    VGC
        STA  STRIG3     ;simulate trigger 3 indicator
VGC    ENDIF

;      Read potentiometers.

        LDX  #3          ;offset to last potentiometer
IVN14  LDA  POT0,X       ;potentiometer reading
        STA  PADDL0,X    ;set potentiometer reading
VGC    IF    VGC
        STA  PADDL4,X    ;simulate potentiometer reading
VGC    ENDIF
        DEX
        BPL  IVN14       ;if not done

;      Start potentiometers for next time.

        STA  POTG0       ;start potentiometers

;      Read paddle triggers.

        LDX  #2          ;offset to paddle trigger reading
        LDY  #1          ;offset to joystick reading
IVN15  LDA  STICK0,Y     ;joystick reading
        LSR  A
        LSR  A
        LSR  A           ;paddle trigger reading
        STA  PTRIG1,X    ;set paddle trigger reading
VGC    IF    VGC
        STA  PTRIG5,X    ;simulate paddle trigger reading
VGC    ENDIF

        LDA  #0
        ROL  A           ;paddle trigger reading
        STA  PTRIG0,X    ;set paddle trigger reading
VGC    IF    VGC
        STA  PTRIG4,X    ;simulate paddle trigger reading
VGC    ENDIF
        DEX
        DEX
        DEY
        BPL  IVN15       ;if not done

;      Exit.

        JMP  (VBLKD)     ;process deferred VBLANK NMI, return

```

```

**      PTO - Process Countdown Timer One Expiration
*
*      ENTRY JSR    PTO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PTO      =      *      ;entry
          JMP      (CDTMA1) ;process countdown timer 1 expiration

```

```

**      PTT - Process Countdown Timer Two Expiration
*
*      ENTRY JSR    PTT
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PTT      =      *      ;entry
          JMP      (CDTMA2) ;process countdown timer 2 expiration

```

```

**      DCT - Decrement Countdown Timer
*
*      ENTRY JSR    DCT
*          X = offset to timer value
*
*      EXIT
*          A = 0, if timer expired
*          = $FF, if timer did not expire
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DCT      =      *      ;entry
          LDY      CDTMV1,X ;low timer value
          BNE      DCT1      ;if low timer value not zero

          LDY      CDTMV1+1,X ;high timer value
          BEQ      DCT2      ;if timer value zero, exit

          DEC      CDTMV1+1,X ;decrement high timer value

DCT1     DEC      CDTMV1,X ;decrement low timer value
          BNE      DCT2      ;if low timer value not zero

          LDY      CDTMV1+1,X ;high timer value
          BNE      DCT2      ;if high timer value not zero

          LDA      #0          ;indicate timer expired
          RTS

DCT2     LDA      #$FF        ;indicate timer did not expire
          RTS

```

```

**      SVP - Set Vertical Blank Parameters
*
*      SVP sets countdown timers and VBLANK vectors.
*
*      ENTRY JSR      SVP
*              X = high initial timer value or high vector address
*              Y = low initial timer value or low vector address
*              A = 1, if timer 1 value
*                  2, if timer 2 value
*                  3, if timer 3 value
*                  4, if timer 4 value
*                  5, if timer 5 value
*                  6, if immediate VBLANK vector
*                  7, if deferred VBLANK vector
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

SVP      =      *      ;entry
;      Initialize.

      ASL      A      ;compute offset+2 to value or vector
      STA      INTEMP ;offset+2 to value or vector
      TXA      ;high timer value or high vector address

;      Ensure no VBLANK in progress by delaying after HBLANK.

      LDX      #5      ;20 CPU cycles
      STA      WSYNC   ;wait for HBLANK synchronization

SVP1     DEX
      BNE      SVP1    ;if not done delaying

;      Set timer value or vector address.

      LDX      INTEMP  ;offset+2 to value or vector
      STA      CDTMV1-2+1,X ;high timer value or high vector address
      TYA
      STA      CDTMV1-2,X ;low timer value or low vector address
      RTS

```

```

**      DVNM - Process Deferred VBLANK NMI
*
*      ENTRY JMP      DVNM
*
*      EXIT
*              Exits via RTI
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

DVNM     =      *      ;entry
      PLA
      TAY      ;restore Y
      PLA
      TAX      ;restore X
      PLA      ;restore A
      RTI      ;return

```

Initialization

```
**      PWS - Perform Warmstart
*
*      ENTRY JMP      PWS
*
*      EXIT
*          Exits to PCS or PRS
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
PWS      =      *      ;entry
;      Initialize.
      SEI
;      Check for cartridge change.
      LDA      TRIG3 ;cartridge interlock
      CMP      GINTLK ;previous cartridge interlock status
      BNE      PCS      ;if cartridge changed, perform coldstart
;      Check for cartridge.
      ROR      A
      BCC      PWS1      ;if no cartridge
;      Verify no change in cartridge.
      JSR      CCE      ;check cartridge equivalence
      BNE      PCS      ;if different cartridge, coldstart
;      Check coldstart status.
PWS1     LDA      COLDST ;coldstart status
      BNE      PCS      ;if coldstart was in progress, perform coldstart
;      Perform warmstart.
      LDA      #$FF      ;indicate warmstart
      BNE      PRS      ;preset memory, return
```

```
**      RES - Process RESET
*
*      ENTRY JMP      RES
*
*      EXIT
*          Exits to PCS, if coldstart, or PWS, if warmstart
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
RES      =      *      ;entry
;      Initialize.
      SEI
```

```

;      Delay 0.1 second for RESET bounce.

      LDX    #140    ;0.1 second delay
RES1  DEY
      BNE    RES1    ;if inner loop not done

      DEX
      BNE    RES1    ;if outer loop not done

;      Check power-up validation bytes.

      LDA    PUPBT1
      CMP    #PUPVL1
      BNE    PCS      ;if validation byte 1 differs, coldstart

      LDA    PUPBT2
      CMP    #PUPVL2
      BNE    PCS      ;if validation byte 2 differs, coldstart

      LDA    PUPBT3
      CMP    #PUPVL3
      BEQ    PWS      ;if all bytes validated, perform warmstart

;      JMP    PCS      ;perform coldstart, return

```

```

**      PCS - Perform Coldstart
*
*      ENTRY  JMP    PCS
*
*      EXIT
*           Exits to PRS
*
*      MODS
*           Original Author Unknown
*           1. Bring closer to Coding Standard (object unchanged).
*           R. K. Nordin      1983-11-01

```

```

PCS    =      *      ;entry
      LDA    #0      ;indicate coldstart
;      JMP    PRS     ;preset memory, return

```

```

**      PRS - Preset Memory
*
*      ENTRY  JMP    PRS
*
*      EXIT
*           Exits via CARTCS vector or DOSVEC vector
*
*      NOTES
*           Problem: in the CRASS65 version, APPMHI was not declared
*           zero-page.
*
*      MODS
*           Original Author Unknown
*           1. Bring closer to Coding Standard (object unchanged).
*           R. K. Nordin      1983-11-01

```

```

PRS    =      *      ;entry

;      Update warmstart flag.

      STA    WARMST    ;update warmstart flag

```



```

;      Set initial conditions.

      SEI
      CLD
      LDX    #$FF
      TXS                      ;set stack pointer

;      Initialize LNBUG flag, if necessary.

LNBUG IF      LNBUG
      LDY    #0                ;assume no LNBUG
      LDA    LNORG             ;first byte of LNBUG
      CMP    #$4C              ;$4C instruction
      BNE    PRS2              ;if $4C not present, indicate no LNBUG

      INC    LNORG             ;try to increment test byte
      CMP    LNORG             ;original contents of test byte
      BEQ    PRS1              ;if no change, LNBUG present

      DEC    LNORG             ;restore test byte
      BNE    PRS2              ;indicate no LNBUG

PRS1   DEY                      ;LNBUG present indicator

PRS2   STY    LNFLAG           ;LNBUG flag
LNBUG ENDF

;      Perform miscellaneous initialization.

      JSR    PMI                ;perform miscellaneous initialization

;      Initialize memory status.

      LDA    #1                ;no failure indicator
      STA    NGFLAG            ;memory status flag

;      Check type.

      LDA    WARMST            ;warmstart flag
      BNE    PRS8              ;if warmstart

;      Zero all RAM (except beginning of page zero).

      LDA    #0
      LDY    #WARMST           ;initial offset into page zero
      STA    RAMLO
      STA    RAMLO+1           ;initialize RAM pointer

PRS3   LDA    #$FF
      STA    (RAMLO),Y         ;attempt to store $FF
      CMP    (RAMLO),Y
      BEQ    PRS4              ;if $FF stored successfully

      LSR    NGFLAG            ;indicate memory failure

PRS4   LDA    #$00
      STA    (RAMLO),Y         ;attempt to store $00
      CMP    (RAMLO),Y
      BEQ    PRS5              ;if $00 stored successfully

      LSR    NGFLAG            ;indicate memory failure

PRS5   INY
      BNE    PRS3              ;if not end of page

;      Advance to next page and check for completion.

```

```

    INC    RAMLO+1      ;advance RAM pointer to next page
    LDX    RAMLO+1
    CPX    TRAMSZ       ;RAM size
    BNE    PRS3         ;if not at end of RAM

;    Initialize DOSVEC.

    LDA    #low PPD     ;power-up display routine address
    STA    DOSVEC       ;initialize DOS vector
    LDA    #high PPD
    STA    DOSVEC+1

;    Verify ROM checksums.

    LDA    PORTB
    AND    #$7F         ;select self-test ROM
    STA    PORTB        ;port B memory control

    JSR    VFR          ;verify first 8K ROM
    BCS    PRS6         ;if first 8K ROM bad

    JSR    VSR          ;verify second 8K ROM
    BCC    PRS7         ;if second 8K ROM good

PRS6    LSR    NGFLAG    ;indicate memory bad

PRS7    LDA    PORTB
    ORA    #$80         ;disable self-test ROM
    STA    PORTB        ;update port B memory control

;    Indicate coldstart in progress.

    LDA    #$FF
    STA    COLDST       ;indicate coldstart in progress
    BNE    PRS12        ;continue with coldstart procedures

;    Perform warmstart procedures.

PRS8    LDX    #0

    LDA    DERRF        ;screen OPEN error flag
    BEQ    PRS9         ;if in screen OPEN

;    Clean up APPMHI.

;    STX    APPMHI
;    VFD    8\,$8E,8\low APPMHI,8\high APPMHI
;    STX    APPMHI+1
;    VFD    8\,$8E,8\low [APPMHI+1],8\high [APPMHI+1]

    TXA

;    Clear page 2 and part of page 3.

PRS9    STA    $0200,X   ;clear byte of page 2

    CPX    #low ACMVAR  ;start of page 3 locations not to clear
    BCS    PRS10        ;if not to clear this page 3 location

    STA    $0300,X      ;clear byte of page 3

PRS10   DEX
    BNE    PRS9         ;if not done

;    Clear part of page 0.

```

```

        LDX    #INTZBS        ;offset to first page 0 byte to clear
PRS11  STA    $0000,X        ;clear byte of page 0
        INX
        BPL    PRS11         ;if not done
;      Record BASIC status.

PRS12  LDX    #0             ;initially assume BASIC enabled
        LDA    PORTB         ;port B memory control
        AND    #$02         ;BASIC enabled indicator
        BEQ    PRS13         ;if BASIC enabled

        INX                 ;indicate BASIC disabled
PRS13  STX    BASICF         ;BASIC flag
;      Establish power-up validation bytes.

        LDA    #PUPVL1
        STA    PUPBT1        ;validation byte 1
        LDA    #PUPVL2
        STA    PUPBT2        ;validation byte 2
        LDA    #PUPVL3
        STA    PUPBT3        ;validation byte 3
;      Establish screen margins.

        LDA    #LEDGE
        STA    LMARGN        ;left margin
        LDA    #REDGE
        STA    RMARGN        ;right margin
;      Establish parameters for NTSC or PAL.

        LDA    PAL           ;GTIA flag bits
        AND    #$0E         ;PAL/NTSC indicator
        BNE    PRS14         ;if NTSC

        LDA    #5           ;PAL key repeat delay
        LDX    #1           ;PAL indicator
        LDY    #40          ;PAL key repeat initial delay
        BNE    PRS15         ;set parameters

PRS14  LDA    #6           ;NTSC key repeat delay
        LDX    #0           ;NTSC indicator
        LDY    #48          ;NTSC key repeat initial delay

PRS15  STA    KEYREP        ;set key repeat rate
        STX    PALNTS       ;set PAL/NTSC status
        STY    KRPDEL       ;set key repeat initial delay
;      Initialize missing controller ports, if not simulated.

VGC    IF    not VGC
        LDA    #$0F         ;joystick centered
        STA    STICK2
        STA    STICK3
        LDA    #$01         ;trigger not pressed
        STA    STRIG2
        STA    STRIG3

        LDX    #3           ;offset to last controller
PRS16  LDA    #$E4         ;paddle fully counter-clockwise
        STA    PADDL4,X
        LDA    #$01         ;trigger not pressed

```

```

VGC    STA    PTRIG4,X
      DEX
      BPL     PRS16      ;if not done
      ENDIF

;      Copy interrupt vector table from ROM to RAM.

      LDX     #TIHVL-1    ;offset to last byte of table
PRS17  LDA     TIHV,X      ;byte of table of interrupt vectors
      STA     INTABS,X    ;byte of RAM table
      DEX
      BPL     PRS17      ;if not done

;      Copy handler vector table from ROM to RAM.

      LDX     #THAVL-1    ;offset to last byte of table
PRS18  LDA     THAV,X      ;byte of handler vector table
      STA     HATABS,X    ;byte of RAM table
      DEX
      BPL     PRS18      ;if not done

;      Initialize software.

      JSR     ISW          ;initialize software

;      Initialize ACMI module, if present.

ACMI   IF      ACMI
      LDA     TRIG2
      ROR     A
      BCS     PRS19      ;if ACMI module not inserted

      LDA     AMFLAG
      BMI     PRS19      ;if not to install handler

;      Install ACMI handler.

      LDA     AMNAME      ;device code
      STA     HATABS+THAVL
      LDA     #low AMVTAD ;vector table address
      STA     HATABS+THAVL+1
      LDA     #high AMVTAD
      STA     HATABS+THAVL+2

;      Set ACMI interrupt service routine.

      LDA     AMISRA      ;interrupt service routine address
      STA     ACMISR
      LDA     AMISRA+1
      STA     ACMISR+1

;      Initialize ACMI.

      JSR     AMINIT      ;initialize ACMI
      JMP     PRS20      ;continue

;      Set ACMI interrupt routine.

PRS19  LDA     #low PAI    ;ACMI interrupt routine address
      STA     ACMISR
      LDA     #high PAI
      STA     ACMISR+1

;      Continue.

```

```
PRS20
ACMI  ENDIF
```

```
;    Enable IRQ interrupts.

    CLI

;    Check for memory problems.

    LDA    NGFLAG        ;memory status
    BNE    PRS21          ;if memory good

;    Perform memory self-test on bad memory.

    LDA    PORTB
    AND    #$7F           ;enable self-test ROM
    STA    PORTB          ;update port B memory control
    LDA    #2
    STA    CHACT          ;CHACTL (character control) shadow
    LDA    #high DCSORG   ;high domestic character set origin
    STA    CHBAS          ;CHBASE (character base) shadow
    JMP    EMS            ;execute memory self-test

;    Check for cartridge.

PRS21 LDX    #0
    STX    TRAMSZ         ;clear cartridge flag

    LDX    RAMSIZ         ;RAM size
    CPX    #high $B000    ;start of cartridge area
    BCS    PRS22          ;if RAM in cartridge area

    LDX    CART
    BNE    PRS22          ;if no cartridge

    INC    TRAMSZ         ;set cartridge flag
    JSR    CCE            ;check cartridge equivalence
    JSR    ICS            ;initialize cartridge software

;    Open screen editor.

PRS22 LDA    #OPEN
    LDX    #SEIOCB        ;screen editor IOCB index
    STA    ICCOM,X        ;command
    LDA    #low SEDS      ;screen editor device specification
    STA    ICBAL,X        ;buffer address
    LDA    #high SEDS
    STA    ICBAH,X
    LDA    #OPNIN+OPNOT   ;open for input/output
    STA    ICAX1,X        ;auxiliary information 1
    JSR    CIOV           ;vector to CIO
    BPL    PRS23          ;if no error

;    Process error (which should never happen).

    JMP    RES            ;retry power-up

;    Delay, ensuring VBLANK.

PRS23 INX
    BNE    PRS23          ;if inner loop not done

    INY
    BPL    PRS23          ;if outer loop not done

;    Attempt cassette boot.
```

```

    JSR    ACB            ;attempt cassette boot
;    Check cartridge for disk boot.

    LDA    TRAMSZ
    BEQ    PRS24          ;if no cartridge

    LDA    CARTFG        ;cartridge mode flags
    ROR    A
    BCC    PRS25          ;if disk boot not desired
;    Attempt disk boot.
PRS24 JSR    ADB            ;attempt disk boot
;    Initialize peripheral handler loading facility.

    JSR    PHR            ;poll, load, relocate, initialize
;    Indicate coldstart complete.
PRS25 LDA    #0
    STA    COLDST        ;indicate coldstart complete
;    Check cartridge for execution.

    LDA    TRAMSZ
    BEQ    PRS26          ;if no cartridge

    LDA    CARTFG        ;cartridge mode flags
    AND    #$04
    BEQ    PRS26          ;if execution not desired
;    Execute cartridge.

    JMP    (CARTCS)       ;execute cartridge
;    Exit to power-up display or booted program.
PRS26 JMP    (DOSVEC)     ;vector to booted program

```

```

**    ICS - Initialize Cartridge Software
*
*    ENTRY JSR    ICS
*
*    MODS
*        Original Author Unknown
*        1. Bring closer to Coding Standard (object unchanged).
*        R. K. Nordin    1983-11-01

```

```

ICS    =    *            ;entry
    JMP    (CARTAD)      ;initialize cartridge software

```

```

**      PAI - Process ACMI Interrupt
*
*      PAI does nothing.
*
*      ENTRY JSR    PAI
*
*      NOTES
*          Problem: this code is unneeded unless ACMI assembly
*          option is selected.
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PAI      =      *      ;entry
          CLC
          RTS      ;return

```

```

**      THAV - Table of Handler Vectors
*
*      NOTES
*          THAV is moved to RAM table HATABS.

```

```

THAV      DB      PRINTR      ;printer device code
          DW      PRINTV      ;printer handler vector table

          DB      CASSET      ;cassette device code
          DW      CASETV      ;cassette handler vector table

          DB      SCREDIT      ;editor device code
          DW      EDITRV      ;editor handler vector table

          DB      DISPLAY      ;screen device code
          DW      SCRENV      ;screen handler vector table

          DB      KBD          ;keyboard device code
          DW      KEYBDV      ;keyboard handler vector table

THAVL     =      *-THAV      ;length

```

```

**      BMSG - Boot Error Message

```

```

BMSG      DB      'BOOT ERROR',EOL

```

```

**      Screen Editor Device Specification

```

```

SEDS      DB      'E:',EOL

```

```

**      TIHV - Table of Interrupt Handler Vectors
*
*      NOTES
*          TIHV is moved to RAM table INTABS.

```

```

TIHV      DW      RIR      ;VDSLST - display list NMI vector
          DW      XIR      ;VPRCED - proceed line IRQ vector
          DW      XIR      ;VINTER - interrupt line IRQ vector
          DW      XIR      ;VBREAK - BRK instruction IRQ vector
          DW      KIR      ;VKEYBD - keyboard IRQ vector
          DW      IRIR      ;VSERIN - serial input ready IRQ vector

```

```

DW    ORIR    ;VSEROR - serial output ready IRQ vector
DW    OCIR    ;VSEROC - serial output complete IRQ vector
DW    XIR     ;VTIMR1 - POKEY timer 1 IRQ vector
DW    XIR     ;VTIMR2 - POKEY timer 2 IRQ vector
DW    XIR     ;VTIMR4 - POKEY timer 4 IRQ vector
DW    IIR     ;VIMIRQ - immediate IRQ vector
DW    0       ;CDTMV1 - countdown timer 1 vector
DW    0       ;CDTMV2 - countdown timer 2 vector
DW    0       ;CDTMV3 - countdown timer 3 vector
DW    0       ;CDTMV4 - countdown timer 4 vector
DW    0       ;CDTMV5 - countdown timer 5 vector
DW    IVNM    ;VVBLKI - immediate VBLANK NMI vector
DW    DVNM    ;VVBLKD - deferred VBLANK NMI vector

```

```
TIHVL =    *-TIHV ;length
```

```

**    PMI - Perform Miscellaneous Initialization
*
*    ENTRY JSR    PMI
*
*    NOTES
*        Problem: initial address for sizing RAM should be
*        $4000 (16K) instead of $2800.
*
*    MODS
*        Original Author Unknown
*        1. Bring closer to Coding Standard (object unchanged).
*        R. K. Nordin      1983-11-01

```

```

PMI    =    *    ;entry

;    Check for cartridge special execution case.

LDA    TRIG3
ROR    A
BCC    PMI1    ;if cartridge not inserted

LDA    CART
BNE    PMI1    ;if not cartridge

LDA    CARTFG ;cartridge flags
BPL    PMI1    ;if special execution not desired

;    Execute cartridge.

JMP    (CARTAD)    ;execute cartridge

;    Initialize hardware.

PMI1   JSR    IHW    ;initialize hardware

;    Disable BASIC.

LDA    PORTB
ORA    #$02    ;disable BASIC
STA    PORTB    ;update port B memory control

;    If warmstart, check previous BASIC status.

LDA    WARMST
BEQ    PMI2    ;if coldstart

LDA    BASICF ;BASIC flag
BNE    PMI4    ;if BASIC not previously enabled

```



```

    BEQ    PMI3    ;enable BASIC

;    Check OPTION key.

PMI2    LDA    CONSOL ;console switches
    AND    #$04    ;OPTION key indicator
    BEQ    PMI4    ;if OPTION key pressed, do not enable BASIC

;    Enable BASIC.

PMI3    LDA    PORTB
    AND    #$FD    ;disable BASIC
    STA    PORTB    ;update port B memory control

;    Determine size of RAM.

RAMSYS IF    RAMSYS
PMI4    LDA    #high $4000    ;16K
    STA    TRAMSZ            ;set RAM size
    RTS                    ;return
RAMSYS ELSE

PMI4    LDA    #low $2800    ;initial low address
    TAY                    ;offset to first byte of page
    STA    TRAMSZ-1        ;set initial low address

    LDA    #high $2800    ;initial RAM size
    STA    TRAMSZ            ;set initial RAM size (high address)

PMI5    LDA    (TRAMSZ-1),Y    ;first byte of page
    EOR    #$FF            ;complement
    STA    (TRAMSZ-1),Y    ;attempt to store complement
    CMP    (TRAMSZ-1),Y
    BNE    PMI6            ;if complement not stored

    EOR    #$FF            ;original value
    STA    (TRAMSZ-1),Y    ;attempt to store original value
    CMP    (TRAMSZ-1),Y
    BNE    PMI6            ;if original value not stored

    INC    TRAMSZ            ;increment high address
    BNE    PMI5            ;continue

;    Exit.

PMI6    RTS                    ;return
RAMSYS ENDIF

```

```

**    CCE - Check Cartridge Equivalence
*
*    ENTRY JSR    CCE
*
*    NOTES
*        Problem: this code checksums $BFF0 - $C0EF; it should
*        checksum $BF00 - $BFFF.
*
*    MODS
*        Original Author Unknown
*        1. Bring closer to Coding Standard (object unchanged).
*        R. K. Nordin    1983-11-01

```

```

CCE    =    *    ;entry

;    Initialize.

```

```

        LDA    #0      ;initial sum
        TAX          ;offset to first byte
        CLC

;       Checksum 256 bytes of cartridge area.

CCE1    ADC     $BFF0,X    ;add in byte
        INX
        BNE     CCE1      ;if not done

;       Exit.

        CMP     CARTCK ;previous checksum
        STA     CARTCK ;new checksum
        RTS      ;return

```

```

**      IHW - Initialize Hardware
*
*      ENTRY JSR    IHW
*
*      MODS
*
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

IHW      =      *      ;entry

;       Initialize CTIA, ANTIC and POKEY areas.

        LDA     #0      ;initialization value
        TAX          ;initial offset
        STA     PBCTL    ;set for direction register first

IHW1     STA     CTIA,X    ;initialize CTIA/GTIA area register
        STA     ANTIC,X   ;initialize ANTIC area register
        STA     POKEY,X   ;initialize POKEY area register
        CPX     #low PORTB
        BEQ     IHW2      ;if port B, don't initialize

        STA     PIA,X     ;initialize PIA area register

IHW2     INX
        BNE     IHW1      ;if not done

;       Initialize PIA.

        LDA     #$3C
        STA     PBCTL    ;precondition port B outputs
        LDA     #$FF
        STA     PORTB    ;all high
        LDA     #$38
        STA     PACTL    ;select data direction register
        STA     PBCTL    ;select data direction register
        LDA     #$00
        STA     PORTA    ;all inputs
        LDA     #$FF
        STA     PORTB    ;all outputs
        LDA     #$3C
        STA     PACTL    ;back to port
        STA     PBCTL    ;back to port
        LDA     PORTB    ;clear interrupts
        LDA     PORTA    ;clear interrupts

;       Initialize POKEY.

```

```

LDA    #$22    ;get POKEY out of initialize mode and set ch. 4
STA    SKCTL    ;set serial port control

LDA    #$A0    ;pure tone, no volume
STA    AUDC3    ;turn off channel 3
STA    AUDC4    ;turn off channel 4

LDA    #$28    ;clock ch. 3 with 1.79 MHz, ch. 4 with ch. 3
STA    AUDCTL    ;set audio control

LDA    #$FF
STA    SEROUT    ;start bit only

RTS                ;return

```

```

**      ISW - Initialize Software
*
*      ENTRY JSR    ISW
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

ISW      =      *                ;entry

;      Initialize BREAK key handling.

DEC      BRKKEY                ;turn off BREAK key flag

LDA      #low BIR
STA      BRKKY                ;set BREAK key IRQ routine address
LDA      #high BIR
STA      BRKKY+1

;      Initialize RAMSIZ and MEMTOP.

LDA      TRAMSZ                ;determined size of RAM
STA      RAMSIZ                ;size of RAM
STA      MEMTOP+1              ;high top of memory
LDA      #$00
STA      MEMTOP                ;low top of memory

;      Initialize MEMLO.

LDA      #low INIML            ;initial MEMLO address
STA      MEMLO
LDA      #high INIML
STA      MEMLO+1

;      Initialize device handlers.

JSR      EDITRV+12              ;initialize editor handler
JSR      SCRENV+12              ;initialize screen handler
JSR      KEYBDV+12              ;initialize keyboard handler
JSR      PRINTV+12              ;initialize printer handler
JSR      CASETV+12              ;initialize cassette handler

;      Initialize various routines.

JSR      CIOINV                ;initialize CIO
JSR      SIOINV                ;initialize SIO
JSR      INTINV                ;initialize interrupt handler
JSR      DINITV                ;initialize DIO

```

```

; Initialize generic parallel device handler.

LDA    #low PIR
STA    VPIRQ        ;parallel device IRQ routine address
LDA    #high PIR
STA    VPIRQ+1

JSR    GPDVV+12      ;initialize parallel device handler

; Set status of START key.

LDA    CONSOL        ;console switches
AND    #$01          ;START key indicator
EOR    #$01          ;START key status
STA    CKEY          ;cassette boot request flag

RTS                    ;return

```

```

**      ADB - Attempt Disk Boot
*
*      ENTRY JSR    ADB
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

ADB    =    *        ;entry

; Check type of reset.

LDA    WARMST
BEQ    ADB1    ;if not warmstart

; Process warmstart.

LDA    BOOT?    ;successful boot flags
AND    #$01    ;successful disk boot indicator
BEQ    BAI2    ;if disk boot not successful, return

; Initialize disk booted software.

JMP    IBS      ;initialize booted software

; Process coldstart.

ADB1   LDA    #1
STA    DUNIT    ;disk unit number
LDA    #STATC    ;status
STA    DCOMND    ;command
JSR    DSKINV    ;issue command
BMI    BAI2    ;if error, return

; Boot.

; JMP    ABI      ;attempt boot and initialize

```

```

**      ABI - Attempt Boot and Initialize
*
*      ENTRY JSR      ABI
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

ABI      =      *          ;entry

          LDA      #high 1
          STA      DAUX2
          LDA      #low 1      ;sector number
          STA      DAUX1

          LDA      #low [CASBUF+3]      ;buffer address
          STA      DBUFLO
          LDA      #high [CASBUF+3]
          STA      DBUFHI

;      JNF      BAI          ;boot and initialize

```

```

**      BAI - Boot and Initialize
*
*      ENTRY JSR      BAI
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

BAI      =      *          ;entry

;      Read first sector.

      JSR      GNS      ;get next sector
      BPL      CBI      ;if no error, complete boot and initialize

;      Process error.

BAI1     JSR      DBE      ;display boot error message

          LDA      CASSBT
          BEQ     ABI      ;if not cassette boot, try again

;      Exit.

BAI2     RTS              ;return

```

```

**      CBI - Complete Boot and Initialize
*
*      ENTRY JSR      CBI
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CBI      =      *          ;entry

;      Transfer flags.

```

```

        LDX    #3

CBI1   LDA    CASBUF+3,X    ;byte from buffer
        STA    DFLAGS,X    ;flag byte
        DEX
        BPL    CBI1        ;if not done

;       Transfer sector.

        LDA    BOOTAD
        STA    RAMLO        ;set boot address
        LDA    BOOTAD+1
        STA    RAMLO+1

        LDA    CASBUF+7
        STA    DOSINI        ;establish initialization address
        LDA    CASBUF+8
        STA    DOSINI+1

CBI2   LDY    #127          ;offset to last byte of sector

CBI3   LDA    CASBUF+3,Y    ;byte of sector buffer
        STA    (RAMLO),Y    ;byte of boot program
        DEY
        BPL    CBI3        ;if not done

;       Increment loader buffer pointer.

        CLC
        LDA    RAMLO
        ADC    #$80
        STA    RAMLO
        LDA    RAMLO+1
        ADC    #0
        STA    RAMLO+1    ;increment boot loader buffer pointer

;       Decrement and check number of sectors.

        DEC    DBSECT        ;decrement number of sectors
        BEQ    CBI5        ;if no more sectors

;       Get next sector.

        INC    DAUX1        ;increment sector number

CBI4   JSR    GNS            ;get next sector
        BPL    CBI2        ;if status OK

;       Process error.

        JSR    DBE            ;display boot error message
        LDA    CASSBT
        BNE    BAI1        ;if cassette, start over

        BEQ    CBI4        ;try sector again

;       Clean up.

CBI5   LDA    CASSBT
        BEQ    CBI6        ;if not cassette boot

        JSR    GNS            ;get EOF record (but do not use it)

;       Execute boot loader.

CBI6   JSR    EBL            ;execute boot loader
        BCS    BAI1        ;if bad boot, try again

```

```
;      Initialize booted software.

JSR    IBS          ;initialize booted software
INC     BOOT?       ;indicate boot success
RTS                     ;return
```

```
**      EBL - Execute Boot Loader
*
*      ENTRY JSR     EBL
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
EBL     =      *          ;entry

;      Move boot loader start address to RAMLO.

CLC
LDA     BOOTAD
ADC     #6
STA     RAMLO          ;boot loader start address
LDA     BOOTAD+1
ADC     #0
STA     RAMLO+1

;      Execute boot loader.

JMP     (RAMLO)        ;execute boot loader
```

```
**      IBS - Initialize Booted Software
*
*      ENTRY JSR     IBS
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
IBS     =      *          ;entry
JMP     (DOSINI)        ;initialize booted software
```

```
**      DBE - Display Boot Error Message
*
*      ENTRY JSR     DBE
*
*      NOTES
*          Problem: bytes wasted by LDX/TXA and LDY/TYA
*          combinations.
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
DBE     =      *          ;entry

;      Set up IOCB.

LDX     #low BMSG      ;boot error message
LDY     #high BMSG
```

```

TXA
LDX #SEIOCB ;screen editor IOCB index
STA ICBAL,X ;low buffer address
TYA
STA ICBAL,X ;high buffer address
LDA #PUTREC
STA ICCOM,X ;command
LDA #$FF
STA ICBLL,X ;buffer length

; Perform CIO.

JMP CIOV ;vector to CIO, return

```

```

** GNS - Get Next Sector
*
* ENTRY JSR GNS
*
* MODS
* Original Author Unknown
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01

```

```

GNS = * ;entry

; Check type of boot.

LDA CASSBT
BEQ GNS1 ;if not cassette boot

; Read block from cassette.

JMP RBLQKV ;vector to read cassette block routine, return

; Read sector from disk.

GNS1 LDA #READ
STA DCOMND ;command
LDA #1 ;drive number 1
STA DUNIT ;set drive number
JMP DSKINV ;vector to DIO, return

```

```

** ACB - Attempt Cassette Boot
*
* ENTRY JSR ACB
*
* MODS
* Original Author Unknown
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01

```

```

ACB = * ;entry

; Check type.

LDA WARMST ;warmstart flag
BEQ ACB1 ;if coldstart

; Perform warmstart procedures.

LDA B00T? ;successful boot flags
AND #$02 ;successful cassette boot indicator
BEQ ACB2 ;if cassette boot not successful

```



```

        JMP      ACB3          ;initialize cassette
;      Perform coldstart procedures.
ACB1    LDA      CKEY          ;cassette boot request flag
        BEQ      ACB2          ;if cassette boot not requested, return
;      Boot cassette.
        LDA      #$80
        STA      FTYPE        ;set long IRG type
        INC      CASSBT       ;set cassette boot flag
        JSR      CSOPIV       ;open cassette for input
        JSR      BAI          ;boot and initialize
        LDA      #0
        STA      CASSBT       ;clear cassette boot flag
        STA      CKEY         ;clear cassette boot request flag
        ASL      BOOT?        ;indicate successful cassette boot
        LDA      DOSINI
        STA      CASINI        ;cassette initialization address
        LDA      DOSINI+1
        STA      CASINI+1
;      Exit.
ACB2    RTS                   ;return
;      Initialize cassette booted program.
ACB3    JMP      (CASINI)      ;initialize cassette booted program

```

Disk Input/Output

```
**      IDIO - Initialize DIO
*
*      ENTRY JSR      IDIO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
IDIO  =      *          ;entry
      LDA      #160      ;160 second timeout
      STA      DSKTIM     ;set initial disk timeout
      LDA      #low DSCTS ;disk sector size
      STA      DSCTLN
      LDA      #high DSCTS
      STA      DSCTLN+1
      RTS          ;return
```

```
**      DIO - Disk I/O
*
*      ENTRY JSR      DIO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
DIO    =      *          ;entry
;      Initialize.

      LDA      #DISKID    ;disk bus ID
      STA      DDEVIC     ;device bus ID
      LDA      DSKTIM     ;timeout
      LDX      DCOMND     ;command
      CPX      #FORMAT
      BEQ      DI01       ;if FORMAT command

      LDA      #7         ;set timeout to 7 seconds
DI01   STA      DTIMLO     ;timeout
;      Set SIO command.

      LDX      #GETDAT    ;assume GET DATA

      LDA      DCOMND     ;command
      CMP      #PUTSEC
      BEQ      DI02       ;if PUT SECTOR command

      CMP      #WRITE
      BNE      DI03       ;if not WRITE command
DI02   LDX      #PUTDAT    ;select PUT DATA
;      Check command.

DI03   CMP      #STATC
      BNE      DI04       ;if not STATUS command
;      Set up STATUS command.
```

```

        LDA    #low DVSTAT
        STA    DBUFLO      ;buffer address
        LDA    #high DVSTAT
        STA    DBUFHI
        LDY    #low 4      ;low byte count
        LDA    #high 4     ;high byte count
        BEQ    DI05        ;perform SIO

;      Set up other commands.

DI04    LDY    DSCTLN      ;low byte count
        LDA    DSCTLN+1    ;high byte count

;      Perform SIO.

DI05    STX    DSTATS      ;SIO command
        STY    DBYTLO      ;low byte count
        STA    DBYTHI      ;high byte count
        JSR    SIOV        ;vector to SIO
        BPL    DI06        ;if no error

;      Process error.

        RTS                ;return

;      Process successful operation.

DI06    LDA    DCOMND      ;command
        CMP    #STATC
        BNE    DI07        ;if not STATUS command

        JSR    SBA         ;set buffer address
        LDY    #2
        LDA    (BUFADR),Y  ;timeout status
        STA    DSKTIM      ;disk timeout

;      Set byte count.

DI07    LDA    DCOMND
        CMP    #FOMAT
        BNE    DI010       ;if not FORMAT command

        JSR    SBA         ;set buffer address
        LDY    #$FE        ;initial buffer pointer

DI08    INY
        INY                ;increment buffer pointer
        INY                ;increment buffer pointer

DI09    LDA    (BUFADR),Y  ;low bad sector data
        CMP    #$FF
        BNE    DI08        ;if low not $FF

        INY
        LDA    (BUFADR),Y  ;high bad sector data
        INY
        CMP    #$FF
        BNE    DI09        ;if high not $FF

        DEY
        DEY
        STY    DBYTLO      ;low bad sector byte count
        LDA    #0
        STA    DBYTHI      ;high bad sector byte count

;      Exit.

```

```
DI010 LDY   DSTATS      ;status
      RTS                ;return
```

```
**      SBA - Set Buffer Address
*
*      ENTRY JSR   SBA
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
SBA   =      *          ;entry
      LDA     DBUFLO
      STA     BUFADR      ;buffer address
      LDA     DBUFHI
      STA     BUFADR+1
      RTS                ;return
```

Relocating Loader

```
**      RLR - Relocate Routine
*
*      RLR relocates a relocatable routine which is assembled at
*      origin 0.
*
*      ENTRY  JSR    RLR
*              GBYTEA - GBYTEA+1 = address of get-byte routine
*
*      MODS
*              Y. M. Chen    1982-04-01
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin  1983-11-01
```

```
RLR      =      *              ;entry

;      Clear parameter block.

          LDX      #5              ;offset to last parameter

RLR1     LDA      #0
          STA      PARMBL,X        ;clear byte of parameter block
          DEX
          BPL      RLR1            ;if not done

;      Get a new record type and set the subroutine vectors.

RLR2     LDA      #0
          STA      LCOUNT          ;process 0th byte of a record
          JSR      GBY              ;get type ID
          LDY      #DATAER
          BCS      RLR4            ;if EOF before END record

          STA      HIBYTE           ;save type ID
          JSR      GBY              ;get record length
          LDY      #DATAER
          BCS      RLR4            ;if EOF before END record

          STA      RECLN
          LDA      HIBYTE           ;get type ID
          CMP      #$0B            ;END record
          BEQ      END              ;if END record

          ROL      A                ;set subroutine vectors
          TAX
          LDA      TRPR,X
          STA      RUNADR
          LDA      TRPR+1,X
          STA      RUNADR+1

RLR3     LDA      RECLN
          CMP      LCOUNT
          BEQ      RLR2            ;if LCOUNT=RECLN, get new record

          JSR      GBY              ;get next byte
          LDY      #DATAER
          BCS      RLR4            ;if EOF before END record

          JSR      CAL              ;call record subroutine
          INC      LCOUNT
          BNE      RLR3            ;continue

RLR4     RTS                        ;return
```

```

**      END - Handle END Record
*
*      END handles record type of
*      1.End Record
*
*      Record format:
*      Byte 0      Type ID
*      Byte 1      Self-start flag
*      Bytes 2-3   Run address
*
*      Process formula
*
*      RUNADR+LOADAD ==> Start Execution Address n Loader-Caller
*      parameter block.
*
*      End record calculates the start execution address by adding
*      RUNADR with LOADAD, and returns to the Caller with parameter
*      block and a status byte in the Y register. Y=1 means operation
*      successful, else is a data structure error.
*
*      ENTRY JSR    END
*
*      MODS
*      Y. M. Chen    1982-04-01
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin  1983-11-01

```

```

END      =      *      ;entry
          JSR      GBY      ;get low byte of the RUNADR
          LDY      #DATAER
          BCS      END3      ;if EOF before END record

          STA      RUNADR
          JSR      GBY      ;get high byte of the RUNADR
          LDY      #DATAER
          BCS      END3      ;if EOF before END record

          STA      RUNADR+1
          LDA      RECLEN ;RECLEN here is self-start flag
          CMP      #1
          BEQ      END2      ;if 1, an absolute RUNADR, no fixup

          BCC      END4      ;if 0, this is not a self-start program

;      Process relative start.

          CLC
          LDA      RUNADR      ;execution address, needs fixup
          ADC      LOADAD
          TAY
          LDA      RUNADR+1
          ADC      LOADAD+1      ;A= high byte, Y=low byte

END1     STY      RUNADR      ;set up Loader-Caller parameter block
          STA      RUNADR+1

END2     LDY      #SUCCES      ;Y=1 successful operation

END3     RTS

END4     LDY      #0          ;fill self-start parameter with 0
          LDA      #0          ;for non-self start program
          BEQ      END1        ;continue

```

```

**      GBY - Get Byte
*
*      ENTRY JSR      GBY
*
*      MODS
*          Y. M. Chen    1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

GBY      =      *          ;entry
          JMP      (GBYTEA) ;get byte, return

```

```

**      CAL - Execute at Run Address
*
*      ENTRY JSR      CAL
*
*      MODS
*          Y. M. Chen    1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

CAL      =      *          ;entry
          JMP      (RUNADR) ;process record, return

```

```

**      TEX - Handle Text Record
*
*      TEX handles record types of
*
*      1.Non-zero page relocatable text
*      2.Zero page relocatable text
*      3.Absolute text
*
*      Record format
*
*      |Type |Length          |Relative addr.    |text |
*      |ID   |(RECLen)      |(RELADR)         |     |
*
*      Process formula
*      A register ==> (NEWADR+LCOUNT)
*
*      Relocate object text into fixed address of NEWADR+LCOUNT.
*
*      ENTRY JSR      TEX
*
*      NOTES
*
*      1.The relocating address (NEWADR) for absolute text is the
*      relative address (RELADR), relocating address fixup is not
*      needed.
*      2.There is no need to compare MEMTOP for processing zero page
*      text.
*      3.X register is used as an indexing to zero page variables
*      or non-zero page variables.      X=0 means pointing to non-zero
*      page variable, whereas X=2 means pointing to zero page
*      variables.
*      4.Each byte of the object text comes in A register.
*
*      MODS
*          Y. M. Chen    1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

TEX      =      *          ;entry

```

```

LDY    LCOUNT ;A register=data coming in
CPY    #$01
BEQ    TEX1    ;if 1, process highest used address

BCS    FTX     ;if 2 or greater, relocate object text

STA    RELADR
STA    NEWADR ;for absolute text NEWADR=RELADR
BCC    TEX8

;      Set highest used address.

TEX1    STA    RELADR+1    ;save high byte of RELADR
        STA    NEWADR+1    ;for absolute text NEWADR=RELADR
        LDX    #0         ;X=an index to non-zero or zero variable
        LDA    HIBYTE     ;HIBYTE=Type ID
        BEQ    TEX2       ;if 0, process non-zero page text record
        CMP    #$0A
        BEQ    TEX3       ;if $0A, needs no relative address fixup

        LDX    #2         ;X=2 for zero page text record

TEX2    CLC                ;fix relocating addr. for non-zero page
        LDA    RELADR      ;text & zero page text
        ADC    LOADAD,X    ;NEWADR=RELADR+LOADAD
        STA    NEWADR
        LDA    RELADR+1
        ADC    LOADAD+1,X
        STA    NEWADR+1    ;Loader start relocating

TEX3    CLC
        LDA    NEWADR ;NEWADR+RECLen is the last used memory
        ADC    RECLen ;for this particular record
        PHA
        LDA    #0         ;A=high byte, S=low byte
        ADC    NEWADR+1
        TAY                ;high byte
        PLA                ;low byte
        SEC
        SBC    #2         ;skip unwanted 2 bytes of relative address info
        BCS    TEX4

        DEY

TEX4    PHA
        TYA
        CMP    HIUSED+1,X  ;HIUSED stores the highest used memory
        PLA
        BCC    TEX6       ;if HIUSED>(NEWADR+RECLen), check memory

        BNE    TEX5       ;if HIUSED<=(NEWADR+RECLen)

        CMP    HIUSED,X
        BCC    TEX6

;      Update HIUSED.

TEX5    STA    HIUSED,X    ;update HIUSED
        PHA
        TYA
        STA    HIUSED+1,X
        PLA

TEX6    LDX    HIBYTE
        CPX    #$01
        BEQ    TEX8       ;if zero page text

```



```

;      Check MEMTOP.

      CPY      MEMTOP+1      ;MEMTOP>HIUSED, OK
      BCC      TEX8

      BNE      TEX7

      CMP      MEMTOP
      BCC      TEX8

TEX7   PLA                      ;MEMTOP<=HIUSED then error
      PLA                      ;do a force return to caller
      LDY      #MEMERR        ;set memory insufficient flag

TEX8   RTS                      ;return

```

```

**      FTX - Relocate Text into Memory
*
*      ENTRY JSR      FTX
*
*      NOTES
*          Problem: bytes wasted by JMP to RTS.
*
*      MODS
*          Y. M. Chen      1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

FTX   =      *      ;entry
      SEC
      PHA      ;A register has object text
      LDA      LCOUNT ;LCOUNT counts 2 bytes of relative address
      SBC      #2      ;-2 is the total bytes of object text
      CLC
      ADC      NEWADR
      STA      LTEMP   ;A ==>(NEWADR+LCOUNT-2)
      LDA      #0
      ADC      NEWADR+1
      STA      LTEMP+1
      PLA
      LDY      #0
      STA      (LTEMP),Y
      JMP      TEX8    ;return

```

```

**      WOR - Handle Word Reference Record Type
*
*      WOR handles record types of
*
*      1.Non-zero page word references to non-zero page.
*      2.Zero page word references to non-zero page.
*
*      Record format
*
*      |Type |Length          |Offset1|Offset2|Offsetn|
*      | ID  |(RECLEN)      |A Reg. |      |      |
*
*      Process formula
*
*      (A register +NEWADR)W +LOADAD ==> (NEWADR+ A register)
*
*      Count, the offset from the start relocating address, is the
*      low byte
*      of a word needing to be fixed. The fixup process is to get the
*      content of the word and add loading address, then restore the
*      fixed word.
*
*      Offset information comes in A register.
*
*      ENTRY JSR    WOR
*
*      MODS
*      Y. M. Chen    1982-04-01
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin  1983-11-01

```

```

WOR      =      *      ;entry
          CLC
          ADC     NEWADR      ;offset in A register
          STA     LTEMP
          LDA     #0
          ADC     NEWADR+1
          STA     LTEMP+1      ;offset +NEWADR= LTEMP
          LDY     #0
          LDA     (LTEMP),Y    ;get low byte content of what in LTEMP
          CLC
          ADC     LOADAD       ;fix low byte of a word
          STA     (LTEMP),Y
          INC     LTEMP        ;increment LTEMP pointer by one
          BNE     WOR1         ;if low not zero

          INC     LTEMP+1      ;increment high

WOR1     LDA     (LTEMP),Y    ;fix high byte of a word
          ADC     LOADAD+1
          STA     (LTEMP),Y    ;restore processed content
          RTS                 ;return

```

```

**      L00 - Handle Low Byte and One Byte Record Types
*
*      L00 handles record types of
*
*      1.Non-zero page low byte references to non-zero page.
*      2.Zero page low byte references to non-zero page.
*      3.Non-zero page one byte references to zero page.
*      4.Zero page one byte references to zero page.
*
*      Record format
*
*      |Type |Length          |Offset1|Offset2|Offsetn|
*      |ID   |(RECLen)      |A Reg. |A Reg.  |        |
*
*      The process formula for non-zero page low byte references to
*      non-zero page record and zero page low byte references to
*      non-zero page record is
*
*      (offset + NEWADR)+LOADAD ==> (offset +NEWADR)
*
*      The process formula for non-zero page one byte references to
*      zero
*      page record and zero page one byte references to zero page
*      record
*      is
*
*      (offset + NEWADR)+LOADADZ ==> (offset + NEWADR)
*
*      Count from the offset from the start relocating address is the
*      low byte or one byte need to be fixed. Get the content of the
*      low byte or one byte and add either LOADAD or LOADADZ (Zero
*      page loading address), then restore the value.
*
*      The offset comes in A register.
*
*      The X register for this routine points to either non-zero page
*      variables or zero page variables. Record type 2 & 3 needs
*      non-zero page variable, type 4 & 5 needs zero page variable.
*
*      X=2 points to zero page variable.
*
*      ENTRY JSR    L00
*
*      MODS
*
*      Y. M. Chen    1982-04-01
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin  1983-11-01

```

```

L00    =      *      ;entry
LDX    #0      ;X=0 points to non-zero page variable
LDY    HIBYTE  ;HIBYTE has Type ID
CPY    #4      ;type 4 & 5 needs zero page variable
BCC    L001    ;if type 2 or 3, need non-zero page variable

LDX    #2      ;point to zero page variable

L001   CLC      ;offset is in A register
ADC    NEWADR  ;offset+NEWADR=the byte needs fixup
STA    LTEMP
LDA    #0
ADC    NEWADR+1
STA    LTEMP+1
LDY    #0
LDA    (LTEMP),Y ;get the content of offset+NEWADR
CLC

```

```

ADC    LOADAD,X    ;do relocating fixup
STA    (LTEMP),Y   ;restore the being fixed value
RTS                                ;return

```

```

**    HIG - Handle High Byte Record Types
*
*    HIG handles record types of
*
*    1.Non-zero page high bytes references to non-zero page.
*    2.Zero page high bytes references to non-zero page.
*
*    Record format
*
*    |Type |Length           |Offset1|Low Byte|Offset2|Low...|
*    |ID   |(RECLEN)      |HIBYTE |A Reg.  |(HIBYTE)|      |
*
*    Process formula
*
*    (HIBYTE+NEWADR)+[[LOADAD+A]/256] ==> (HIBYTE+NEWADR)
*
*    Count the offset from the start relocating address is the high
*    byte needs to be fixed. Get the low byte information in the
*    A register, then add the low byte with LOADAD and set the C
*    flag depending on the calculation. Next do an addition of
*    high byte, NEWADR and the C flag. Restore the addition result
*    back to the high byte location in memory.
*
*    HIBYTE is not Type ID here. HIBYTE is used to store the high
*    byte value.
*
*    ENTRY JSR    HIG
*
*    NOTES
*        Problem: many instances of jumping to RTS instruction
*        wastes bytes.
*
*    MODS
*        Y. M. Chen    1982-04-01
*        1. Bring closer to Coding Standard (object unchanged).
*        R. K. Nordin    1983-11-01

```

```

HIG    =    *                ;entry
;
;    Initialize.
;
;    PHA                    ;save offset pointing to high byte
;
;    Check LCOUNT odd or even.
;
;    LDA    LCOUNT
;    ROR    A
;    PLA
;    BCS    HIG2            ;if even number, process low byte
;
;    Process high byte.
;
;    CLC
;    ADC    NEWADR
;    STA    LTEMP            ;get high byte value
;    LDA    #0
;    ADC    NEWADR+1
;    STA    LTEMP+1
;    LDY    #0
;    LDA    (LTEMP),Y
;    STA    HIBYTE            ;save high byte content

```

```

HIG1  RTS                ;return

;      Process low byte.

HIG2  CLC
      ADC  LOADAD        ;add low byte with LOADAD
      LDA  #0
      ADC  LOADAD+1
      ADC  HIBYTE        ;C flag+LOADAD(high byte)+HIBYTE
      LDY  #0
      STA  (LTEMP),Y     ;store being fixed high byte value
      BEQ  HIG1

```

** TRPR - Table of Record Processing Routines
--

```

TRPR  DW    TEX    ;0 - non-zero page relocatable text
      DW    TEX    ;1 - zero page relocatable text
      DW    L00    ;2 - non-zero page low byte to non-zero page
      DW    L00    ;3 - zero page low byte to non-zero page
      DW    L00    ;4 - non-zero page one byte to zero page
      DW    L00    ;5 - zero page one byte to zero page
      DW    WOR    ;6 - non-zero page word to non-zero page
      DW    WOR    ;7 - zero page word to non-zero page
      DW    HIG    ;8 - non-zero page high byte to non-zero page
      DW    HIG    ;9 - zero page high byte to non-zero page
      DW    TEX    ;10 - absolute text
      DW    END    ;11 - end record

```

Self-test, Part 1

```
**      SES - Select and Execute Self-test
*
*      SES selects the self-test ROM and executes the self-test.
*
*      ENTRY JSR    SES
*
*      NOTES
*      Problem: this could be contiguous with other OS ROM
*      self-test code (near TST0).
*
*      MODS
*      M. W. Colburn      1982-10-26
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01
```

```
SES      =      *      ;entry

      LDA      #$FF
      STA      COLDST ;force coldstart on RESET

      LDA      PORTB
      AND      #$7F   ;enable self-test ROM
      STA      PORTB  ;update port B memory control

      JMP      SLFTSV ;vector to self-test
```

Parallel Input/Output

```
**      GIN - Initialize Generic Parallel Device
*
*      ENTRY JSR      GIN
*
*      MODS
*          Y. M. Chen    1983-02-18
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01
```

```
GIN      =      *      ;entry
;      Initialize.

      LDA      #$01 ;initially select device 0
      STA      SHPDVS ;device select shadow

;      For each potential device, initialize if device present.

GIN1     LDA      SHPDVS ;device select shadow
      STA      PDVS ;device select

      LDA      PDID1 ;first ID
      CMP      #$80 ;required value
      BNE      GIN2 ;if first ID not verified

      LDA      PDID2 ;second ID
      CMP      #$91 ;required value
      BNE      GIN2 ;if second ID not verified

      JSR      PDVV+12 ;initialize parallel device handler

GIN2     ASL      SHPDVS ;advance to next device
      BNE      GIN1 ;if devices remain

;      Exit.

      LDA      #$00 ;select FPP (deselect device)
;      STA      SHPDVS ;device select shadow
      STA      PDVS ;device select
      RTS ;return
```

```
**      PIO - Parallel Input/Output
*
*      ENTRY JSR      PIO
*
*      NOTES
*          Problem: in the CRASS65 version, CRITIC was not declared
*          zero-page.
*
*      MODS
*          Y. M. Chen    1983-02-18
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01
```

```
PIO      =      *      ;entry
;      Initialize.

      LDA      #1
;      STA      CRITIC ;indicate critical section
      VFD      8\,$D,8\low CRITIC,8\high CRITIC
```

```

    LDA    DUNIT    ;device unit number
    PHA                    ;save device unit number
    LDA    PDVMSK   ;device selection mask
    BEQ    PI02      ;if no device to select

;    For each device, pass request to device I/O routine.

    LDX    #TPDSL   ;offset to first byte beyond table

PI01    JSR    SNP    ;select next parallel device
        BEQ    PI02      ;if no device selected

        TXA
        PHA                    ;save offset
        JSR    PDIOV   ;perform parallel device I/O
        PLA                    ;saved offset
        TAX                    ;restore offset
        BCC    PI01      ;if device did not field request

;    Restore Floating Point Package.

    LDA    #$00      ;select FPP (deselect device)
    STA    SHPDVS    ;device select shadow
    STA    PDVS       ;device select
    BEQ    PI03       ;exit

;    Perform SIO.

PI02    JSR    SIO    ;perform SIO

;    Exit.

PI03    PLA                    ;saved device unit number
        STA    DUNIT    ;restore device unit number
        LDA    #0
;    STA    CRITIC ;indicate non-critical section
        VFD    8\8D,8\low CRITIC,8\high CRITIC
        STY    DSTATS
        LDY    DSTATS ;status (re-establish N)
        RTS                    ;return

```

```

**      PIR - Handle Parallel Device IRQ
*
*      ENTRY  JMP    PIR
*
*      EXIT
*
*      Exits via RTI
*
*      MODS
*
*      Y. M. Chen    1983-02-18
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin    1983-11-01

```

```

PIR    =    *    ;entry

;    Determine which device made IRQ, in order of priority.

    LDX    #TPDSL   ;offset to first byte beyond table

PIR1    ROR    A
        BCS    PIR2    ;if IRQ of that device

        DEX
        BNE    PIR1    ;if devices remain

```



```

;      Select device and process IRQ.

PIR2  LDA    SHPDVS      ;current device selection
      PHA      ;save current device selection
      LDA    TPDS-1,X    ;device selection desired
      STA    SHPDVS      ;device select shadow
      STA    PDVS        ;device select
      JSR    PDIRQV      ;process IRQ

;      Exit.

      PLA      ;saved device selection
      STA    SHPDVS      ;restore device select shadow
      STA    PDVS        ;device select
      PLA      ;saved X
      TAX      ;restore X
      PLA      ;restore A
      RTI      ;return

```

```

**      GOP - Perform Generic Parallel Device OPEN
*
*      ENTRY JSR    GOP
*
*      MODS
*          Y. M. Chen    1983-02-18
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

GOP    =      *      ;entry
      LDY    #1      ;offset for OPEN
      JMP    EPC      ;execute parallel device handler command, return

```

```

**      GCL - Perform Generic Parallel Device CLOSE
*
*      ENTRY JSR    GCL
*
*      MODS
*          Y. M. Chen    1983-02-18
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

GCL    =      *      ;entry
      LDY    #3      ;offset for CLOSE
      JMP    EPC      ;execute parallel device handler command, return

```

```

**      GGB - Perform Generic Parallel Device GET-BYTE
*
*      ENTRY JSR    GGB
*
*      MODS
*          Y. M. Chen    1983-02-18
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

GGB    =      *      ;entry
      LDY    #5      ;offset for GET-BYTE
      JMP    EPC      ;execute parallel device handler command, return

```

```

**      GPB - Perform Generic Parallel Device PUT-BYTE
*
*      ENTRY JSR      GPB
*
*      MODS
*          Y. M. Chen    1983-02-18
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

GPB      =      *      ;entry
          LDY      #7      ;offset for PUT-BYTE
          JMP      EPC      ;execute parallel device handler command, return

```

```

**      GST - Perform Generic Parallel Device STATUS
*
*      ENTRY JSR      GST
*
*      MODS
*          Y. M. Chen    1983-02-18
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

GST      =      *      ;entry
          LDY      #9      ;offset for STATUS
          JMP      EPC      ;execute parallel device handler command, return

```

```

**      GSP - Perform Generic Parallel Device SPECIAL
*
*      ENTRY JSR      GSP
*
*      MODS
*          Y. M. Chen    1983-02-18
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

GSP      =      *      ;entry
          LDY      #11     ;offset for SPECIAL
          JMP      EPC      ;execute parallel device handler command, return

```

```

**      SNP - Select Next Parallel Device
*
*      ENTRY JSR      SNP
*
*      MODS
*          Y. M. Chen    1983-02-18
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

SNP      =      *      ;entry

;      Decrement and check offset.

SNP1     DEX          ;decrement offset
          BPL         SNP2      ;if devices remain

;      Exit.

          LDA         #$00      ;select FPP (deselect device)
          STA         SHPDVS     ;device select shadow
          STA         PDVS      ;device select
          RTS

```

```

;      Ensure device is indicated by selection mask.

SNP2  LDA    PDVMSK ;device selection mask
      AND    TPDS,X ;device select
      BEQ    SNP1   ;if device not indicated for selection

;      Select device.

      STA    SHPDVS ;device select shadow
      STA    PDVS   ;device select
      RTS

```

```

**      IPH - Invoke Parallel Device Handler
*
*      ENTRY JSR    IPH
*              Y = offset into parallel device vector table
*              PPTMPA = original A value
*              PPTMPX = original X value
*
*      NOTES
*              Problem: wasted byte for DEY.
*
*      MODS
*              Y. M. Chen    1983-02-18
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin  1983-11-01

```

```

IPH    =      *      ;entry
      LDA    PDVV,Y ;high routine address-1
      PHA                    ;place on stack
      DEY
      LDA    PDVV,Y ;low routine address-1
      PHA                    ;place on stack
      LDA    PPTMPA ;restore A for handler
      LDX    PPTMPX ;restore X for handler
      LDY    #FNCNOT      ;preset status
      RTS                ;invoke handler routine (address on stack)

```

```

**      EPC - Execute Parallel Device Handler Command
*
*      ENTRY JSR    EPC
*
*      NOTES
*              Problem: in the CRASS65 version, CRITIC was not declared
*              zero-page.
*
*      MODS
*              Y. M. Chen    1983-02-18
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin  1983-11-01

```

```

EPC    =      *      ;entry

;      Initialize.

      STA    PPTMPA ;save data byte
      STX    PPTMPX ;save X
;      LDA    CRITIC
;      VFD    8\AD,8\low CRITIC,8\high CRITIC
;      PHA                    ;save critical section status
;      LDA    #1
;      STA    CRITIC ;indicate critical section
;      VFD    8\8D,8\low CRITIC,8\high CRITIC

```

```

;      For each device, pass request to device handler.

      LDX      #TPDSL ;offset to first byte beyond table

EPC1  JSR      SNP      ;select next device
      BEQ      EPC2     ;if no device selected, return error

      TXA
      PHA      ;save offset
      TYA
      PHA      ;save Y
      JSR      IPH      ;invoke parallel device handler
      BCC      EPC4     ;if device did not field, try next device

;      Clean up.

      STA      PPTMPA ;save possible data byte
      PLA      ;clean stack
      PLA
      JMP      EPC3     ;exit

;      Return Nonexistent Device error.

EPC2  LDY      #NONDEV

;      Restore Floating Point Package.

EPC3  LDA      #$00     ;select FPP (deselect device)
      STA      SHPDVS ;device select shadow
      STA      PDVS    ;device select
      PLA      ;saved critical section status
;      STA      CRITIC ;restore critical section status
      VFD      8,$8D,8\low CRITIC,8\high CRITIC
      LDA      PPTMPA ;restore possible data byte
      STY      PPTMPX
      LDY      PPTMPX ;status (re-establish N)
      RTS      ;return

;      Prepare to try next device.

EPC4  PLA
      TAY      ;restore Y
      PLA
      TAX      ;restore X
      BCC      EPC1   ;try next device

```

```

**      TPDS - Table of Parallel Device Selects
*
*      NOTES
*      Problem: bytes wasted by replication of this table
*      elsewhere.

```

```

TPDS  DB      $80      ;0 - device 7 (lowest priority)
      DB      $40      ;1 - device 6
      DB      $20      ;2 - device 5
      DB      $10      ;3 - device 4
      DB      $08      ;4 - device 3
      DB      $04      ;5 - device 2
      DB      $02      ;6 - device 1
      DB      $01      ;7 - device 0 (highest priority)

TPDSL =      *-TPDS ;length

```

Peripheral Handler Loading Facility, Part 1

```
**      PHL - Load and Initialize Peripheral Handler
*
*      Subroutine to load, relocate, initialize and open a
*      "provisionally" opened IOCB. This routine is called
*      upon first I/O attempt following provisional open.
*      It does the final opening by simulating the first
*      part of a normal CIO OPEN and then finishing with
*      code which is in CIO.
*
*      Input parameters:
*      ICIDNO (specifies which IOCB);
*      various values in the provisionally-opened IOCB:
*          ICSPR (handler name)
*          ICSPR+1 (serial address for loading);
*      whatever the called subroutines require.
*
*      Output parameters:
*      None. (Error returns are all handled by called subroutines--
*          in fact, all returns are handled by called subroutines.)
*
*      Modified:
*      ICHID in both calling IOCB and ZIOCB (part of completing OPEN);
*      ICCOMT (a CIO variable);
*      Registers not saved.
*
*      Subroutines called:
*      LPH (does the loading);
*      PHC (initializes the loaded handler);
*      FDH (a CIO entry--finds handler table entry of
*          newly loaded/initialized handler);
*      IIO (a CIO entry--finishes full proper opening of IOCB,
*          including calling handler OPEN entry--IIO returns
*          to PHL's caller);
*      IND (a CIO entry--returns with error to PHL's caller).
*
*      ENTRY JSR    PHL
*
*      NOTES
*      Problem: in the CRASS65 version, ICIDNO was not declared
*      zero-page.
*
*      MODS
*      R. S. Scheiman      1982-04-01
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01
```

```
PHL    =      *          ;entry
;      Load peripheral handler.
;
;      LDX    ICIDNO      ;IOCB index
;      VFD    8\AE,8\low ICIDNO,8\high ICIDNO
;      LDA    ICSPR+1,X
;      JSR    LPH          ;load peripheral handler
;      BCS    PHL1         ;if error
;
;      Initialize peripheral handler.
;
;      CLC
;      JSR    PHC          ;indicate zero handler size
;      JSR    PHC          ;initialize peripheral handler
;      BCS    PHL1         ;if error
;
;      Find device handler.
```

```

;      LDX      ICIDNO      ;IOCB index
VFD      8\AE,8\low ICIDNO,8\high ICIDNO
LDA      ICSPR,X
JSR      FDH      ;find device handler
BCS      PHL1      ;if not found

;      Set handler ID.

;      LDX      ICIDNO      ;IOCB index
VFD      8\AE,8\low ICIDNO,8\high ICIDNO
STA      ICHID,X      ;handler ID
STA      ICHIDZ

;      Simulate initial CIO OPEN processing.

LDA      #OPEN ;OPEN command
STA      ICCOMT ;command
JMP      IIO      ;initialize IOCB for OPEN, return

;      Indicate nonexistent device error.
PHL1     JMP      IND      ;indicate nonexistent device error, return

```

Self-test, Part 2

** TST0 - Table of Self-test Text Offsets
--

TST0	DB	TXT0-TTXT	;0 - offset to "MEMORY TEST ROM" text
	DB	TXT1-TTXT	;1 - offset to "RAM" text
	DB	TXT2-TTXT	;2 - offset to "KEYBOARD TEST" text
	DB	TXT3-TTXT	;3 - offset to "S P A C E B A R" text
	DB	TXT4-TTXT	;4 - offset to "SH" text
	DB	TXT5-TTXT	;5 - offset to "SH" text
	DB	TXT6-TTXT	;6 - offset to "B S" text
	DB	TXT7-TTXT	;7 - offset to keyboard text
	DB	TXT8-TTXT	;8 - offset to control key text
	DB	TXT9-TTXT	;9 - offset to "VOICE #" text

** TTXT - Table of Text Sequences
--

TTXT = *

** TXT0 - "MEMORY TEST ROM" Text

TXT0	DB	\$00,\$00	
	DB	\$2D,\$25,\$2D,\$2F,\$32,\$39	; "MEMORY"
	DB	\$00	
	DB	\$34,\$25,\$33,\$34	; "TEST"
	DB	\$00,\$00,\$00	
	DB	\$32,\$2F,\$2D	; "ROM"

TXT0L = *-TXT0 ;length

** TXT1 - "RAM" Text

TXT1	DB	\$32,\$21,\$2D	; "RAM"
------	----	----------------	---------

TXT1L = *-TXT1 ;length

** TXT2 - "KEYBOARD TEST" Text

TXT2	DB	\$00,\$00	
	DB	\$2B,\$25,\$39,\$22,\$2F,\$21,\$32,\$24	; "KEYBOARD"
	DB	\$00	
	DB	\$34,\$25,\$33,\$34	; "TEST"
	DB	\$00,\$00,\$00	
	DB	\$B2	

TXT2L = *-TXT2 ;length

** TXT7 - Keyboard

TXT7 = *

; First Row (Function Keys)

DB	\$91	; "1"
DB	\$00	
DB	\$92	; "2"
DB	\$00	
DB	\$93	; "3"

```

DB      $00
DB      $94          ;"4"
DB      $00
DB      $A8          ;"H"
DB      $00
DB      $A1          ;"A"
DB      $00
DB      $A2          ;"B"
DB      $00,$00,$00

;      Second Row ("1 2 3 4 5 6 7 8 9 0 < >")

```

```

DB      $5B
DB      $00
DB      $11          ;"1"
DB      $00
DB      $12          ;"2"
DB      $00
DB      $13          ;"3"
DB      $00
DB      $14          ;"4"
DB      $00
DB      $15          ;"5"
DB      $00
DB      $16          ;"6"
DB      $00
DB      $17          ;"7"
DB      $00
DB      $18          ;"8"
DB      $00
DB      $19          ;"9"
DB      $00
DB      $10          ;"0"
DB      $00
DB      $1C          ;"<"
DB      $00
DB      $1E          ;">"
DB      $00
DB      $A2          ;"B"
DB      $80
DB      $B3          ;"S"
DB      $00,$00,$00

```

```

;      Third Row ("Q W E R T Y U I O P - =")

```

```

DB      $FF
DB      $FF
DB      $00
DB      $31          ;"Q"
DB      $00
DB      $37          ;"W"
DB      $00
DB      $25          ;"E"
DB      $00
DB      $32          ;"R"
DB      $00
DB      $34          ;"T"
DB      $00
DB      $39          ;"Y"
DB      $00
DB      $35          ;"U"
DB      $00
DB      $29          ;"I"
DB      $00
DB      $2F          ;"O"
DB      $00
DB      $30          ;"P"

```



```

DB      $00
DB      $0D          ; "-"
DB      $00
DB      $1D          ; "="
DB      $00
DB      $B2          ; "R"
DB      $B4          ; "T"
DB      $00,$00,$00

;      Fourth Row ("A S D F G H J K L ; + *")

```

```

DB      $80
DB      $DC
DB      $80
DB      $00
DB      $21          ; "A"
DB      $00
DB      $33          ; "S"
DB      $00
DB      $24          ; "D"
DB      $00
DB      $26          ; "F"
DB      $00
DB      $27          ; "G"
DB      $00
DB      $28          ; "H"
DB      $00
DB      $2A          ; "J"
DB      $00
DB      $2B          ; "K"
DB      $00
DB      $2C          ; "L"
DB      $00
DB      $1B          ; ";"
DB      $00
DB      $0B          ; "+"
DB      $00
DB      $0A          ; "*"
DB      $00
DB      $A3          ; "C"
DB      $00,$00,$00

```

```

;      Fifth Row ("Z X C V B N M , . /")

```

```

DB      $80
DB      $B3          ; "S"
DB      $A8          ; "H"
DB      $80
DB      $00
DB      $3A          ; "Z"
DB      $00
DB      $38          ; "X"
DB      $00
DB      $23          ; "C"
DB      $00
DB      $36          ; "V"
DB      $00
DB      $22          ; "B"
DB      $00
DB      $2E          ; "N"
DB      $00
DB      $2D          ; "M"
DB      $00
DB      $0C          ; ", "
DB      $00
DB      $0E          ; ". "
DB      $00

```

```

DB      $0F          ;"/"
DB      $00
DB      $80
DB      $B3          ;"S"
DB      $A8          ;"H"
DB      $80
DB      $00,$00,$00

```

; Sixth Row (Space Bar)

```

DB      $00,$00,$00,$00,$00
DB      $80
DB      $B3          ;"S"
DB      $80
DB      $B0          ;"P"
DB      $80
DB      $A1          ;"A"
DB      $80
DB      $A3          ;"C"
DB      $80
DB      $A5          ;"E"
DB      $80
DB      $80
DB      $80
DB      $A2          ;"B"
DB      $80
DB      $A1          ;"A"
DB      $80
DB      $B2          ;"R"
DB      $80

```

TXT7L = *-TXT7 ;length

** TXT3 - "S P A C E B A R" Text

```

TXT3 DB      $00
DB      $33          ;"S"
DB      $00
DB      $30          ;"P"
DB      $00
DB      $21          ;"A"
DB      $00
DB      $23          ;"C"
DB      $00
DB      $25          ;"E"
DB      $00
DB      $00
DB      $00
DB      $22          ;"B"
DB      $00
DB      $21          ;"A"
DB      $00
DB      $32          ;"R"
DB      $00

```

TXT3L = *-TXT3 ;length

** TXT4 - "SH" Text

```

TXT4 DB      $00
DB      $33,$28      ;"SH"
DB      $00

```

TXT4L = *-TXT4 ;length

** TXT5 - "SH" Text

TXT5 = TXT4
TXT5L = TXT4L ;length

** TXT6 - "B S" Text

TXT6 DB \$22 ;"B"
 DB \$00
 DB \$33 ;"S"

TXT6L = *-TXT6 ;length

** TXT8 - Control Key

TXT8 DB \$00
 DB \$5C
 DB \$00

TXT8L = *-TXT8 ;length

** TXT9 - "VOICE #" Text

TXT9 DB \$36,\$2F,\$29,\$23,\$25 ;"VOICE"
 DB \$00
 DB \$03 ;"#"

TXT9L = *-TXT9 ;length

Peripheral Handler Loading Facility, Part 2

```
**      CLT - Checksum Linkage Table
*
*      ENTRY JSR    CLT
*              ZCHAIN - ZCHAIN+1 = address of linkage table
*
*      EXIT
*              A = checksum of linkage table
*
*      CHANGES
*              Y
*
*      CALLS
*              -none-
*
*      MODS
*              R. S. Scheiman      1982-04-01
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01
```

```
CLT      =      *              ;entry
          LDY     #17          ;offset to last byte to sum
          LDA     #0          ;initial sum
          CLC
CLT1     ADC     (ZCHAIN),Y    ;add byte
          DEY
          BPL     CLT1        ;if not done
          ADC     #0          ;add final carry
          EOR     #$FF        ;complement
          RTS                ;return
```

International Character Set

FIX ICSORG

** International Character Set

DB	\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00	;\$00 - space
DB	\$00,\$18,\$18,\$18,\$18,\$00,\$18,\$00	;\$01 - !
DB	\$00,\$66,\$66,\$66,\$66,\$00,\$00,\$00	;\$02 - "
DB	\$00,\$66,\$FF,\$66,\$66,\$FF,\$66,\$00	;\$03 - #
DB	\$18,\$3E,\$60,\$3C,\$06,\$7C,\$18,\$00	;\$04 - \$
DB	\$00,\$66,\$6C,\$18,\$30,\$66,\$46,\$00	;\$05 - %
DB	\$1C,\$36,\$1C,\$38,\$6F,\$66,\$3B,\$00	;\$06 - &
DB	\$00,\$18,\$18,\$18,\$00,\$00,\$00,\$00	;\$07 - '
DB	\$00,\$0E,\$1C,\$18,\$18,\$1C,\$0E,\$00	;\$08 - (
DB	\$00,\$70,\$38,\$18,\$18,\$38,\$70,\$00	;\$09 -)
DB	\$00,\$66,\$3C,\$FF,\$3C,\$66,\$00,\$00	;\$0A - asterisk
DB	\$00,\$18,\$18,\$7E,\$18,\$18,\$00,\$00	;\$0B - plus
DB	\$00,\$00,\$00,\$00,\$00,\$18,\$18,\$30	;\$0C - comma
DB	\$00,\$00,\$00,\$7E,\$00,\$00,\$00,\$00	;\$0D - minus
DB	\$00,\$00,\$00,\$00,\$00,\$18,\$18,\$00	;\$0E - period
DB	\$00,\$06,\$0C,\$18,\$30,\$60,\$40,\$00	;\$0F - /
DB	\$00,\$3C,\$66,\$6E,\$76,\$66,\$3C,\$00	;\$10 - 0
DB	\$00,\$18,\$38,\$18,\$18,\$18,\$7E,\$00	;\$11 - 1
DB	\$00,\$3C,\$66,\$0C,\$18,\$30,\$7E,\$00	;\$12 - 2
DB	\$00,\$7E,\$0C,\$18,\$0C,\$66,\$3C,\$00	;\$13 - 3
DB	\$00,\$0C,\$1C,\$3C,\$6C,\$7E,\$0C,\$00	;\$14 - 4
DB	\$00,\$7E,\$60,\$7C,\$06,\$66,\$3C,\$00	;\$15 - 5
DB	\$00,\$3C,\$60,\$7C,\$66,\$66,\$3C,\$00	;\$16 - 6
DB	\$00,\$7E,\$06,\$0C,\$18,\$30,\$30,\$00	;\$17 - 7
DB	\$00,\$3C,\$66,\$3C,\$66,\$66,\$3C,\$00	;\$18 - 8
DB	\$00,\$3C,\$66,\$3E,\$06,\$0C,\$38,\$00	;\$19 - 9
DB	\$00,\$00,\$18,\$18,\$00,\$18,\$18,\$00	;\$1A - colon
DB	\$00,\$00,\$18,\$18,\$00,\$18,\$18,\$30	;\$1B - semicolon
DB	\$06,\$0C,\$18,\$30,\$18,\$0C,\$06,\$00	;\$1C - <
DB	\$00,\$00,\$7E,\$00,\$00,\$7E,\$00,\$00	;\$1D - =
DB	\$60,\$30,\$18,\$0C,\$18,\$30,\$60,\$00	;\$1E - >
DB	\$00,\$3C,\$66,\$0C,\$18,\$00,\$18,\$00	;\$1F - ?
DB	\$00,\$3C,\$66,\$6E,\$6E,\$60,\$3E,\$00	;\$20 - @
DB	\$00,\$18,\$3C,\$66,\$66,\$7E,\$66,\$00	;\$21 - A
DB	\$00,\$7C,\$66,\$7C,\$66,\$66,\$7C,\$00	;\$22 - B
DB	\$00,\$3C,\$66,\$60,\$60,\$66,\$3C,\$00	;\$23 - C
DB	\$00,\$78,\$6C,\$66,\$66,\$6C,\$78,\$00	;\$24 - D
DB	\$00,\$7E,\$60,\$7C,\$60,\$60,\$7E,\$00	;\$25 - E
DB	\$00,\$7E,\$60,\$7C,\$60,\$60,\$60,\$00	;\$26 - F
DB	\$00,\$3E,\$60,\$60,\$6E,\$66,\$3E,\$00	;\$27 - G
DB	\$00,\$66,\$66,\$7E,\$66,\$66,\$66,\$00	;\$28 - H
DB	\$00,\$7E,\$18,\$18,\$18,\$18,\$7E,\$00	;\$29 - I
DB	\$00,\$06,\$06,\$06,\$06,\$66,\$3C,\$00	;\$2A - J
DB	\$00,\$66,\$6C,\$78,\$78,\$6C,\$66,\$00	;\$2B - K
DB	\$00,\$60,\$60,\$60,\$60,\$60,\$7E,\$00	;\$2C - L
DB	\$00,\$63,\$77,\$7F,\$6B,\$63,\$63,\$00	;\$2D - M
DB	\$00,\$66,\$76,\$7E,\$7E,\$6E,\$66,\$00	;\$2E - N
DB	\$00,\$3C,\$66,\$66,\$66,\$66,\$3C,\$00	;\$2F - O
DB	\$00,\$7C,\$66,\$66,\$7C,\$60,\$60,\$00	;\$30 - P
DB	\$00,\$3C,\$66,\$66,\$66,\$6C,\$36,\$00	;\$31 - Q
DB	\$00,\$7C,\$66,\$66,\$7C,\$6C,\$66,\$00	;\$32 - R
DB	\$00,\$3C,\$60,\$3C,\$06,\$06,\$3C,\$00	;\$33 - S
DB	\$00,\$7E,\$18,\$18,\$18,\$18,\$18,\$00	;\$34 - T
DB	\$00,\$66,\$66,\$66,\$66,\$66,\$7E,\$00	;\$35 - U
DB	\$00,\$66,\$66,\$66,\$66,\$3C,\$18,\$00	;\$36 - V

DB	\$00,\$63,\$63,\$6B,\$7F,\$77,\$63,\$00	;\$37 - W
DB	\$00,\$66,\$66,\$3C,\$3C,\$66,\$66,\$00	;\$38 - X
DB	\$00,\$66,\$66,\$3C,\$18,\$18,\$18,\$00	;\$39 - Y
DB	\$00,\$7E,\$0C,\$18,\$30,\$60,\$7E,\$00	;\$3A - Z
DB	\$00,\$1E,\$18,\$18,\$18,\$18,\$1E,\$00	;\$3B - [
DB	\$00,\$40,\$60,\$30,\$18,\$0C,\$06,\$00	;\$3C - \
DB	\$00,\$78,\$18,\$18,\$18,\$18,\$78,\$00	;\$3D -]
DB	\$00,\$08,\$1C,\$36,\$63,\$00,\$00,\$00	;\$3E - ^
DB	\$00,\$00,\$00,\$00,\$00,\$00,\$FF,\$00	;\$3F - underline
DB	\$0C,\$18,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$40 - acute accent a
DB	\$30,\$18,\$00,\$66,\$66,\$66,\$3E,\$00	;\$41 - acute accent u
DB	\$36,\$6C,\$00,\$76,\$76,\$7E,\$6E,\$00	;\$42 - tilde N
DB	\$0C,\$18,\$7E,\$60,\$7C,\$60,\$7E,\$00	;\$43 - acute accent E
DB	\$00,\$00,\$3C,\$60,\$60,\$3C,\$18,\$30	;\$44 - cedilla c
DB	\$3C,\$66,\$00,\$3C,\$66,\$66,\$3C,\$00	;\$45 - circumflex o
DB	\$30,\$18,\$00,\$3C,\$66,\$66,\$3C,\$00	;\$46 - grave accent o
DB	\$30,\$18,\$00,\$38,\$18,\$18,\$3C,\$00	;\$47 - grave accent i
DB	\$1C,\$30,\$30,\$78,\$30,\$30,\$7E,\$00	;\$48 - U.K. currency
DB	\$00,\$66,\$00,\$38,\$18,\$18,\$3C,\$00	;\$49 - diaeresis i
DB	\$00,\$66,\$00,\$66,\$66,\$66,\$3E,\$00	;\$4A - umlaut u
DB	\$36,\$00,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$4B - umlaut a
DB	\$66,\$00,\$3C,\$66,\$66,\$66,\$3C,\$00	;\$4C - umlaut O
DB	\$0C,\$18,\$00,\$66,\$66,\$66,\$3E,\$00	;\$4D - grave accent u
DB	\$0C,\$18,\$00,\$3C,\$66,\$66,\$3C,\$00	;\$4E - acute accent o
DB	\$00,\$66,\$00,\$3C,\$66,\$66,\$3C,\$00	;\$4F - umlaut o
DB	\$66,\$00,\$66,\$66,\$66,\$66,\$7E,\$00	;\$50 - umlaut U
DB	\$3C,\$66,\$1C,\$06,\$3E,\$66,\$3E,\$00	;\$51 - circumflex a
DB	\$3C,\$66,\$00,\$66,\$66,\$66,\$3E,\$00	;\$52 - circumflex u
DB	\$3C,\$66,\$00,\$38,\$18,\$18,\$3C,\$00	;\$53 - circumflex i
DB	\$0C,\$18,\$3C,\$66,\$7E,\$60,\$3C,\$00	;\$54 - acute accent e
DB	\$30,\$18,\$3C,\$66,\$7E,\$60,\$3C,\$00	;\$55 - grave accent e
DB	\$36,\$6C,\$00,\$7C,\$66,\$66,\$66,\$00	;\$56 - tilde n
DB	\$3C,\$C3,\$3C,\$66,\$7E,\$60,\$3C,\$00	;\$57 - circumflex e
DB	\$18,\$00,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$58 - ring a
DB	\$30,\$18,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$59 - grave accent a
DB	\$18,\$00,\$18,\$3C,\$66,\$7E,\$66,\$00	;\$5A - ring A
DB	\$78,\$60,\$78,\$60,\$7E,\$18,\$1E,\$00	;\$5B - display escape
DB	\$00,\$18,\$3C,\$7E,\$18,\$18,\$18,\$00	;\$5C - up arrow
DB	\$00,\$18,\$18,\$18,\$7E,\$3C,\$18,\$00	;\$5D - down arrow
DB	\$00,\$18,\$30,\$7E,\$30,\$18,\$00,\$00	;\$5E - left arrow
DB	\$00,\$18,\$0C,\$7E,\$0C,\$18,\$00,\$00	;\$5F - right arrow
DB	\$18,\$00,\$18,\$18,\$18,\$18,\$18,\$00	;\$60 - Spanish !
DB	\$00,\$00,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$61 - a
DB	\$00,\$60,\$60,\$7C,\$66,\$66,\$7C,\$00	;\$62 - b
DB	\$00,\$00,\$3C,\$60,\$60,\$60,\$3C,\$00	;\$63 - c
DB	\$00,\$06,\$06,\$3E,\$66,\$66,\$3E,\$00	;\$64 - d
DB	\$00,\$00,\$3C,\$66,\$7E,\$60,\$3C,\$00	;\$65 - e
DB	\$00,\$0E,\$18,\$3E,\$18,\$18,\$18,\$00	;\$66 - f
DB	\$00,\$00,\$3E,\$66,\$66,\$3E,\$06,\$7C	;\$67 - g
DB	\$00,\$60,\$60,\$7C,\$66,\$66,\$66,\$00	;\$68 - h
DB	\$00,\$18,\$00,\$38,\$18,\$18,\$3C,\$00	;\$69 - i
DB	\$00,\$06,\$00,\$06,\$06,\$06,\$06,\$3C	;\$6A - j
DB	\$00,\$60,\$60,\$6C,\$78,\$6C,\$66,\$00	;\$6B - k
DB	\$00,\$38,\$18,\$18,\$18,\$18,\$3C,\$00	;\$6C - l
DB	\$00,\$00,\$66,\$7F,\$7F,\$6B,\$63,\$00	;\$6D - m
DB	\$00,\$00,\$7C,\$66,\$66,\$66,\$66,\$00	;\$6E - n
DB	\$00,\$00,\$3C,\$66,\$66,\$66,\$3C,\$00	;\$6F - o
DB	\$00,\$00,\$7C,\$66,\$66,\$7C,\$60,\$60	;\$70 - p
DB	\$00,\$00,\$3E,\$66,\$66,\$3E,\$06,\$06	;\$71 - q
DB	\$00,\$00,\$7C,\$66,\$60,\$60,\$60,\$00	;\$72 - r
DB	\$00,\$00,\$3E,\$60,\$3C,\$06,\$7C,\$00	;\$73 - s
DB	\$00,\$18,\$7E,\$18,\$18,\$18,\$0E,\$00	;\$74 - t
DB	\$00,\$00,\$66,\$66,\$66,\$66,\$3E,\$00	;\$75 - u

DB	\$00,\$00,\$66,\$66,\$66,\$3C,\$18,\$00	;\$76 - v
DB	\$00,\$00,\$63,\$6B,\$7F,\$3E,\$36,\$00	;\$77 - w
DB	\$00,\$00,\$66,\$3C,\$18,\$3C,\$66,\$00	;\$78 - x
DB	\$00,\$00,\$66,\$66,\$66,\$3E,\$0C,\$78	;\$79 - y
DB	\$00,\$00,\$7E,\$0C,\$18,\$30,\$7E,\$00	;\$7A - z
DB	\$66,\$66,\$18,\$3C,\$66,\$7E,\$66,\$00	;\$7B - umlaut A
DB	\$18,\$18,\$18,\$18,\$18,\$18,\$18,\$18	;\$7C -
DB	\$00,\$7E,\$78,\$7C,\$6E,\$66,\$06,\$00	;\$7D - display clear
DB	\$08,\$18,\$38,\$78,\$38,\$18,\$08,\$00	;\$7E - display backspace
DB	\$10,\$18,\$1C,\$1E,\$1C,\$18,\$10,\$00	;\$7F - display tab

Self-test, Part 3

FIX \$D000
LOC \$5000 ;\$D000 - \$D7FF mapped to \$5000 - \$57FF

```
**      STH - Self-test Hardware
*
*      ENTRY JSR      STH
*
*      NOTES
*          Problem: this is superfluous; SLFTSV could vector to
*          EST.
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

STH = * ;entry
 JMP EST ;execute self-test

```
**      EMS - Execute Memory Self-test
*
*      ENTRY JSR      EMS
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

EMS = * ;entry
 JSR IST ;initialize self-test
 JMP STM ;self-test memory

```
**      EST - Execute Self-test
*
*      ENTRY JSR      EST
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

EST = * ;entry
 JSR IST ;initialize self-test
; **JMP** SEL ;self-test

```
**      SEL - Self-test
*
*      ENTRY JSR      SEL
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

SEL = * ;entry
; Initialize.


```

LDA    #0
STA    STTIME        ;clear main screen timeout timer
STA    STTIME+1
STA    STAUT         ;clear auto-mode flag
STA    AUDCTL        ;initialize audio control register
LDA    #$03          ;initialize POKEY
STA    SKCTL         ;serial port control
JSR    SAS           ;silence all sounds
LDA    #$40          ;disable DLI
STA    NMIEEN        ;NMI enable
LDX    #0            ;main screen colors
JSR    SUC           ;set up colors
LDX    #low DISL1    ;display list for main screen
LDY    #high DISL1
JSR    SDL           ;set up display list
LDA    #low PMD       ;process main screen DLI routine
STA    VDSLST        ;display list NMI address
LDA    #high PMD
STA    VDSLST+1
LDX    #3*4          ;main screen bold lines
LDA    #$AA          ;color 1
JSR    SVR           ;set value in range

;    Wait for all screen DLI's to clear and for VBLANK.

LDX    #0

SEL1   STX    WSYNC        ;wait for HBLANK synchronization
      INX
      BNE    SEL1         ;if not done waiting

;    Wait until beam close to top (main screen DLI near).

SEL2   LDA    VCOUNT
      CMP    #24
      BCS    SEL2         ;if not done waiting

;    Preset for self-test type determination.

      LDA    #$10         ;initially select memory test
      STA    STPASS       ;pass indicator
      LDA    #$C0         ;enable DLI
      STA    NMIEEN

;    Determine type of self-test.

SEL3   LDA    CONSOL      ;console switches
      AND    #$01         ;START key indicator
      BNE    SEL3         ;if START key not pressed

      LDA    #$FF         ;clear character
      STA    CH

      LDA    STSEL        ;selection
      AND    #$0F         ;selection
      CMP    #$01         ;memory test indicator
      BEQ    SEL5         ;if memory test

      CMP    #$02         ;if audio-visual test
      BEQ    SEL6

      CMP    #$04         ;if keyboard test
      BEQ    SEL7

;    Self-test all.

SEL4   LDA    #$88        ;indicate all tests

```

```

        STA    STSEL        ;selection
        LDA    #$FF        ;auto-mode indicator
        STA    STAUT        ;auto-mode flag

;      Self-test memory.

SEL5    JMP     STM          ;self-test memory

;      Self-test audio-visual.

SEL6    JMP     STV          ;self-test audio-visual

;      Self-test keyboard.

SEL7    JMP     STK          ;self-test keyboard

```

```

**      IST - Initialize Self-test
*
*      ENTRY JSR    IST
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

IST      =      *          ;entry
        LDA    #$11        ;indicate memory test
        STA    STSEL        ;selection
        LDA    #$21
        STA    SDMCTL ;select small size playfield
        LDA    #$C0
        STA    NMIE        ;enable DLI
        LDA    #$41
        STA    STJMP        ;ANTIC jump instruction
        LDA    #$FF        ;clear code indicator
        STA    CH          ;key code
        RTS                ;return

```

```

**      SDL - Set Up Display List
*
*      ENTRY JSR    SDL
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SDL      =      *          ;entry

        STA    STKST        ;keyboard self-test flag
        TYA
        PHA                ;save high address
        TXA
        PHA                ;save low address
        LDA    #0
        STA    SDMCTL        ;DMACTL (DMA control) shadow
        STA    HELPG        ;HELP key flag
        LDA    #low POD      ;process DLI routine
        STA    VDSLST
        LDA    #high POD
        STA    VDSLST+1
        LDX    #0*4          ;screen memory
        TXA                ;value is 0
        JSR     SVR          ;set value in range

```

```

PLA                ;saved low address
TAX
PLA                ;saved high address
TAY
STX    SDLSTL      ;low display list address
STX    STJMP+1      ;low display list address
STY    SDLSTH      ;high display list address
STY    STJMP+2      ;high display list address
LDA    #$21
STA    SDMCTL
RTS                ;return

```

```

**      PMD - Process Main Screen DLI
*
*      1) IF MAIN SCREEN IS ON FOR MORE than FIVE MINUTES
*      THEN 'ALL TESTS' SELECTION IS SELECTED AND EXECUTED
*      2) COLORS FOR CURRENTLY SELECTED CHOICE AND THE
*      NON-SELECTED CHOICES ARE DISPLAYED ON FLY
*      3) SELECTION PROCESS IS HANDLED
*
*      ENTRY JMP     PMD
*
*      EXIT
*          Exits via RTI
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PMD      =      *      ;entry

;      Initialize.

      PHA                ;save A
      TXA
      PHA                ;save X

;      Check for 4th time.

PMD1     LDX    #$7A      ;assume non-selected color
      LDA    STPASS      ;pass indicator
      CMP    #$01      ;4th time indicator
      BEQ    PMD3      ;if 4th time

;      Check for selection.

      AND    #$01      ;selection indicator
      BEQ    PMD2      ;if selected

;      Increment and check blink counter.

      INC    STBL      ;increment blink counter
      LDA    STBL      ;blink counter
      AND    #$20      ;blink indicator
      BEQ    PMD2      ;if not to blink

      LDX    #$2C      ;use selected color

;      Set color.

PMD2     STX    WSYNC     ;wait for HBLANK synchronization
      STX    COLPF0      ;playfield 0 color
      CLC
      ROR    STPASS      ;advance pass indicator

```

```

    LDA    #0
    STA    ATRACT

;    Exit.

    PLA
    TAX            ;restore X
    PLA            ;restore A
    RTI            ;return

;    Check for SELECT previously pressed.

PMD3    LDA    STSPP ;SELECT previously pressed flag
        BNE    PMD4 ;if SELECT previously pressed

;    Check for SELECT pressed.

    LDA    CONSOL ;console switches
    AND    #$02   ;SELECT key indicator
    BNE    PMD5   ;if SELECT not pressed, exit

;    Process SELECT pressed.

    LDA    STSEL ;current selection
    ROL    A
    ROL    STSEL ;next selection
    LDA    #$20  ;blink indicator
    STA    STBL  ;blink counter
    LDA    #$FF  ;SELECT previously pressed indicator
    STA    STSPP ;SELECT previously pressed flag
    BNE    PMD5

;    Process SELECT previously pressed.

PMD4    LDA    CONSOL ;console switches
    AND    #$02   ;SELECT key indicator
    BEQ    PMD5   ;if SELECT still pressed

    LDA    #0      ;SELECT not previously pressed indicator
    STA    STSPP   ;SELECT previously pressed flag

PMD5    LDA    STSEL ;selection
    AND    #$0F
    ORA    #$10    ;reset indicate memory test
    STA    STPASS ;pass indicator

;    Advance main screen timer.

    INC    STTIME
    BNE    PMD6   ;if low not zero

    INC    STTIME+1

;    Check main screen timer.

PMD6    LDA    STTIME+1
    CMP    #250  ;main screen timeout
    BNE    PMD7  ;if main screen timed out

;    Process main screen timeout.

    CLI
    JMP    SEL4  ;self-test all

;    Continue.

PMD7    JMP    PMD1 ;continue

```

**** DISL1 - Display List for Main Screen**

```
DISL1 DB    $70,$70,$70,$70,$70
      DB    $47
      DW    SMEM1
      DB    $70,$70,$70
      DB    $4E
      DW    ST3000
      DB    $70
      DB    $F0
      DB    $C6
      DW    SMEM2
      DB    $70,$86
      DB    $70,$86
      DB    $70,$06
      DB    $70,$70
      DB    $4E
      DW    ST3000
      DB    $70,$70,$70
      DB    $42
      DW    SMEM3
      DB    $41
      DW    DISL1
```

**** SMEM1 - "SELF TEST" Text**

```
SMEM1 DB    $00,$00,$00,$00
      DB    $33,$25,$2C,$26      ;"SELF"
      DB    $00
      DB    $34,$25,$33,$34      ;"TEST"
      DB    $00,$00,$00
```

**** SMEM2 - "MEMORY AUDIO-VISUAL KEYBOARD ALL TESTS" Text**

```
SMEM2 DB    $00,$00
      DB    $2D,$25,$2D,$2F,$32,$39      ;"MEMORY"
      DB    $00,$00,$00,$00,$00
      DB    $00,$00,$00,$00,$00
      DB    $21,$35,$24,$29,$2F          ;"AUDIO"
      DB    $0D                          ;"- "
      DB    $36,$29,$33,$35,$21,$2C      ;"VISUAL"
      DB    $00,$00,$00,$00
      DB    $2B,$25,$39,$22,$2F,$21,$32,$24 ;"KEYBOARD"
      DB    $00,$00,$00,$00,$00,$00,$00,$00
      DB    $21,$2C,$2C                  ;"ALL"
      DB    $00
      DB    $34,$25,$33,$34,$33          ;"TESTS"
      DB    $00,$00,$00,$00,$00
```

**** SMEM3 - "SELECT, START OR RESET" Text**

```
SMEM3 DB    $00,$00,$00,$00
      DB    $42
      DB    $B3,$A5,$AC,$A5,$A3,$B4      ;"SELECT"
      DB    $56
      DB    $0C                          ;", "
      DB    $42
      DB    $B3,$B4,$A1,$B2,$B4          ;"START"
      DB    $56
      DB    $2F,$32                      ;"OR"
      DB    $42
```

```

DB      $B2,$A5,$B3,$A5,$B4      ;"RESET"
DB      $56
DB      $00,$00,$00

```

** DISL2 - Display List for Memory Test

```

DISL2 DB      $70,$70,$70
      DB      $46
      DW      ST3000
      DB      $70
      DB      $70,$06
      DB      $70,$08
      DB      $70
      DB      $70,$06
      DB      $70,$08
      DB      $70,$08
      DB      $70,$08
      DB      $70,$08
      DB      $70,$70,$70
      DB      $01
      DW      DISL3

```

** DISL3 - Display List for Exit Text

```

DISL3 DB      $A0,$40
      DB      $42
      DW      SMEM4
      DB      $01
      DW      STJMP

```

** SMEM4 - "RESET OR HELP TO EXIT" Text

```

SMEM4 DB      $00,$00,$00,$00,$00
      DB      $42
      DB      $B2,$A5,$B3,$A5,$B4      ;"RESET"
      DB      $56
      DB      $2F,$32                  ;"OR"
      DB      $42
      DB      $A8,$A5,$AC,$B0          ;"HELP"
      DB      $56
      DB      $34,$2F                  ;"TO"
      DB      $00
      DB      $25,$38,$29,$34          ;"EXIT"
      DB      $00,$00,$00,$00,$00

```

** DISL4 - Display List for Keyboard Test

```

DISL4 DB      $70,$70,$70,$70
      DB      $46
      DW      ST3000
      DB      $70,$70,$70
      DB      $70,$02
      DB      $70
      DB      $70,$02
      DB      $70,$02
      DB      $70,$02
      DB      $70,$02
      DB      $70,$02
      DB      $70,$70
      DB      $01
      DW      DISL3

```

**** DISL5 - Display List for Audio-visual Test**

```
DISL5 DB    $70,$70,$70,$70
      DB    $46
      DW    SMEM5
      DB    $70,$06
      DB    $70,$70
      DB    $4B
      DW    ST3100
      DB    $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
      DB    $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
      DB    $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
      DB    $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
      DB    $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
      DB    $0B,$0B
      DB    $70
      DB    $46
      DW    ST3000
      DB    $70
      DB    $01
      DW    DISL3
```

**** SMEM5 - "AUDIO-VISUAL TEST" Text**

```
SMEM5 DB    $00,$00
      DB    $21,$35,$24,$29,$2F      ;"AUDIO"
      DB    $0D                      ;"-"
      DB    $36,$29,$33,$35,$21,$2C  ;"VISUAL"
      DB    $00,$00,$00,$00
      DB    $00,$00,$00,$00
      DB    $34,$25,$33,$34          ;"TEST"
      DB    $00,$00,$00,$00,$00,$00
```

**** STM - Self-test Memory**

```
*
* STM verifies ROM and RAM by verifying the ROM checksums and
* writing and reading all possible values to each byte of RAM.
*
* ENTRY JSR    STM
*
* NOTES
*     Problem: searches beyond end of TMNT.
*
* MODS
*     M. W. Colburn      1982-10-26
*     1. Bring closer to Coding Standard (object unchanged).
*     R. K. Nordin      1983-11-01
```

```
STM    =    *                ;entry

;    Initialize.

LDX    #low DISL2    ;memory test display list
LDY    #high DISL2
LDA    #0            ;indicate not keyboard self-test
JSR    SDL            ;set up display list
LDX    #1            ;memory test colors
JSR    SUC            ;set up colors
LDX    #0            ;offset to "MEMORY TEST  ROM" test
JSR    SSM            ;set screen memory
LDX    #1            ;offset to "RAM" text
JSR    SSM            ;set screen memory
```

```

;      Test first 8K ROM.

STM1  LDA    ST3020
      CMP    #$AA          ;color 1 for failure
      BEQ    STM4          ;if first 8K ROM already failed

      LDA    #$55          ;color 0 for test
      JSR    DFS           ;display first ROM status
      JSR    DMW           ;delay a middling while
      JSR    VFR           ;verify first 8K ROM
      BCS    STM2          ;if ROM failed

      LDA    #$FF          ;color 2 for success
      JMP    STM3

STM2  LDA    #$AA          ;color 1 for failure

STM3  JSR    DFS           ;display first ROM status

;      Test second 8K ROM.

STM4  LDA    ST3024
      CMP    #$AA          ;color 1 for failure
      BEQ    STM7          ;if second 8K ROM already failed

      LDA    #$55          ;color 0 for test
      JSR    DSS           ;display second ROM status
      JSR    DMW           ;delay a middling while
      JSR    VSR           ;verify second 8K ROM
      BCS    STM5          ;if ROM failed

      LDA    #$FF          ;color 2 for success
      JMP    STM6

STM5  LDA    #$AA          ;color 1 for failure

STM6  JSR    DSS           ;display second ROM status

;      Test RAM.

STM7  LDA    #$C0          ;mask for left side of a screen byte
      STA    STSMM
      LDA    #$04          ;initially select LED 1 off
      STA    STLMM         ;LED mask
      LDA    #0
      STA    STSMP
      STA    STPAG         ;initialize current page
      STA    STPAG+1
      STA    ST1K          ;initialize current 1K to test

;      Test 1K of RAM.

STM8  LDX    STSMP         ;screen memory pointer
      LDA    ST3038,X
      AND    STSMM
      CMP    #$80
      BEQ    STM17         ;if already failed

      CMP    #$08
      BEQ    STM17         ;if already failed

      LDA    #$44          ;color 0 for test
      JSR    DRS           ;display RAM block status
      LDA    STLMM         ;LED mask
      JSR    SLD           ;set LED's
      LDA    STLMM         ;current LED mask
      EOR    #$0C          ;complement LED's selected

```



```

        STA     STLM           ;update LED mask
;       Check for memory not to test.

        LDX     #TMNTL-1+2     ;2 bytes beyond last byte of table
STM9    LDA     TMNT,X         ;range to test
        CMP     STPAG+1       ;high current page
        BEQ     STM15         ;if not to test, indicate success

        DEX
        BPL     STM9          ;if not done
;       Test 1K of RAM.

        LDA     #4             ;number of pages to test
        STA     STPC          ;page count
;       Write initial list to page.

STM10   LDX     #0             ;initial value to write
;       Write list to page.

STM11   LDY     #0             ;offset to first byte of page
STM12   TXA
        STA     (STPAG),Y     ;byte of page
        INX
        INY
        BNE     STM12         ;if not done writing page
;       Verify list written to page.

        STX     STMVAL        ;first correct value to test
        LDY     #0            ;offset to first byte of page
STM13   LDA     (STPAG),Y     ;byte of page
        CMP     STMVAL        ;correct value
        BNE     STM14         ;if not correct value

        INC     STMVAL        ;increment value to test
        INY
        BNE     STM13         ;if not done verifying page
;       Increment and test initial value to write.

        INX
        BNE     STM11         ;increment initial value to write
;                                     ;if not done, write another list
;       Decrement and test page counter.

        INC     STPAG+1       ;increment high current page
        DEC     STPC          ;decrement page count
        BNE     STM10         ;if not done testing pages

        BEQ     STM16         ;indicate success
;       Display failure.

STM14   JSR     DMW           ;delay a middling while
        LDA     #$88          ;color 1 for failure
        JSR     DRS           ;display RAM block status
        JMP     STM17
;       Delay for simulating test of memory not to test.

```

```

STM15 JSR    DLW          ;delay a long while
;      Display success.

STM16 LDA    #$CC        ;color 2 for success
      JSR    DRS          ;display RAM block status

STM17 LDA    STSMM
      BMI    STM20

      LDA    #$C0
      STA    STSMM
      INC    STSMP        ;increment screen memory pointer

STM18 CLC
      LDA    ST1K          ;current 1K to test
      ADC    #high $0400   ;add 1K
      STA    STPAG+1       ;high current page
      STA    ST1K          ;update current 1K to test
      CMP    RAMSIZ        ;RAM size
      BNE    STM8          ;if not done testing RAM

;      Check for auto-mode.

      LDA    STAUT        ;auto-mode flag
      BNE    STM19        ;if auto-mode, perform audio-visual test

;      Test memory again.

      JMP    STM1          ;test memory again

;      Process auto-mode.

STM19 LDA    #$0C        ;indicate LED 1 and 2 off
      JSR    SLD          ;set LED's
      JSR    DLW          ;delay a long while
      JMP    STV          ;self-test audio-visual

STM20 LDA    #$0C
      STA    STSMM
      BNE    STM18

```

```

**      DFS - Display First ROM Status
*
*      ENTRY JSR    DFS
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DFS    =      *          ;entry
      LDX    #1*4        ;first 8K ROM display
      JSR    SVR          ;set value in range
      AND    #$FC
      STA    ST3020+3
      RTS                ;return

```

```

**      DSS - Display Second ROM Status
*
*      ENTRY JSR    DSS
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DSS      =      *          ;entry
          LDX      #2*4      ;second 8K ROM display
          JSR      SVR        ;set value in range
          AND      #$FC
          STA      ST3024+3
          RTS

```

```

**      SLD - Set LED's
*
*      ENTRY JSR    SLD
*          A = LED mask (bit 3 - LED 2, bit 2 - LED 1)
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SLD      =      *          ;entry
          STA      STTMP5     ;save LED mask
          LDA      PORTB
          AND      #$F3       ;clear LED control
          ORA      STTMP5     ;set LED control according to mask
          STA      PORTB     ;update port B memory control
          RTS

```

```

**      DMW - Delay a Middling While
*
*      ENTRY JSR    DMW
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DMW      =      *          ;entry
          LDX      #60        ;60-VBLANK delay
          BNE      DAW        ;delay a while

```

```

**      DLW - Delay a Long While
*
*      ENTRY JSR    DLW
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DLW      =      *          ;entry
          LDX      #150       ;150-VBLANK delay
          BNE      DAW        ;delay a while, return

```

```

**      DAW - Delay a While
*
*      ENTRY JSR    DAW
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DAW      =      *      ;entry

DAW1     LDY      #$FF      ;initialize inner loop counter

DAW2     STY      WSYNC      ;wait for HBLANK synchronization
        DEY
        BNE      DAW2      ;if inner loop not done

        DEX
        BNE      DAW1      ;if outer loop not done

        RTS              ;return

```

```

**      DRS - Display RAM Block Status
*
*      ENTRY JSR    DRS
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DRS      =      *      ;entry
        PHA              ;save color
        LDX      STSMP
        LDA      STSMM
        EOR      #$FF      ;complement
        AND      ST3038,X
        STA      ST3038,X
        PLA              ;saved color
        AND      STSMM
        ORA      ST3038,X
        STA      ST3038,X
        RTS              ;return

```

```

**      POD - Process Other DLI's
*
*      POD turns the last line on the screen into white on black,
*      handles keyboard self-test display of console switches, handles
*      HELP key for exit, and ensures no attract-mode.
*
*      ENTRY JMP    POD
*
*      EXIT
*          Exits via RTI
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

POD      =      *      ;entry

;      Initialize.

```

```

        PHA            ;save A
;       Select colors.

        LDA    #$0C    ;white color
        STA    COLPF1 ;playfield 1 color
        LDA    COLOR4 ;background color
        STA    COLPF2 ;playfield 2 color

;       Ensure no attract-mode.

        LDA    #0      ;no attract-mode
        STA    ATTRACT ;attract-mode timer/flag

;       Check HELP key.

        LDA    HELPPG ;HELP key flag
        BEQ    POD1    ;if HELP not pressed

;       Process HELP key.

        LDA    #0      ;HELP key not pressed indicator
        STA    HELPPG ;HELP key flag
        LDA    #$0C    ;LED's off
        JSR    SLD      ;set LED's
        CLI
        JMP    SEL      ;start over with main screen

;       Check for keyboard self-test.
POD1    LDA    STKST    ;keyboard self-test flag
        BEQ    POD10    ;if not keyboard self-test, exit

;       Set display of console switches pressed.

        LDA    CONSOL  ;console switches
        AND    #$01    ;START key indicator
        BEQ    POD2    ;if START key pressed

        LDA    #$B3
        BNE    POD3    ;set display
POD2    LDA    #$33
POD3    STA    ST301C ;set START key display

        LDA    CONSOL  ;console switches
        AND    #$02    ;SELECT key indicator
        BEQ    POD4    ;if SELECT key pressed

        LDA    #$F3
        BNE    POD5    ;set display
POD4    LDA    #$73
POD5    STA    ST301E ;set SELECT key display

        LDA    CONSOL  ;console switches
        AND    #$04    ;OPTION key indicator
        BEQ    POD6    ;if OPTION key pressed

        LDA    #$AF
        BNE    POD7    ;set display
POD6    LDA    #$2F

```

```

POD7  STA    ST3020 ;set OPTION key display
;      Sound tone if console switches pressed.

      LDA    CONSOL ;console switches
      AND    #$07  ;key indicators
      CMP    #$07  ;no keys pressed
      BEQ    POD8  ;if no keys pressed

      LDA    #100   ;frequency
      STA    AUDF2  ;set frequency of voice 2
      LDA    #$A8   ;pure tone, half volume
      BNE    POD9   ;set control of voice 2

POD8  LDA    #0      ;zero volume

POD9  STA    AUDC2   ;set control of voice 2

;      Exit.

POD10 PLA                ;restore A
      RTI              ;return

```

```

**      TMNT - Table of Memory Not to Test
*
*      NOTES
*      Problem: bytes wasted by redundant entries.

```

```

TMNT  DB      high $0000  ;$0000 - $03FF, zero page and stack
      DB      high $5000  ;$5000 - $53FF, self-test ROM
      DB      high $5400  ;$5400 - $57FF, self-test ROM
      DB      high ST3000 ;ST3000 - ST3000+$03FF, screen memory
      DB      high ST3000 ;ST3000 - ST3000+$03FF, screen memory
      DB      high ST3000 ;ST3000 - ST3000+$03FF, screen memory

TMNTL =      *-TMNT ;length

```

```

**      STK - Self-test Keyboard
*
*      STK verifies the operation of the keyboard by displaying
*      keys as they are pressed. In auto-mode, the verification
*      is simulated.
*
*      ENTRY JSR    STK
*
*      NOTES
*      Problem: one too many bytes taken from TSKP table.
*      Problem: wasted bytes for extra LDA CH.
*      Problem: logic is convoluted (due to SBT and SAS
*      subroutines appearing in the middle of STK).
*
*      MODS
*      M. W. Colburn      1982-10-26
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

STK   =      *              ;entry

;      Initialize.

      LDX    #0
      STX    STSKP          ;initialize simulated keypress index
      LDX    #3              ;keyboard test colors
      JSR    SUC            ;set up colors

```

```

    LDX    #low DISL4    ;keyboard display list
    LDY    #high DISL4
    LDA    #$FF          ;indicate keyboard self-test
    JSR    SDL           ;set up display list

;    Test keyboard.

STK1  LDX    #2          ;offset to "KEYBOARD TEST" text
    JSR    SSM          ;set screen memory
    LDX    #7           ;offset to keyboard text
    JSR    SSM          ;set screen memory

;    Check auto-mode.

    LDA    STAUT        ;auto-mode flag
    BEQ    STK3         ;if not auto-mode

;    Simulate keypress.

STK2  LDX    STSKP      ;offset to next simulated keypress
    LDA    TSKP,X       ;simulated keypress
    INC    STSKP        ;advance offset to simulated keypress
    LDX    STSKP        ;offset to simulated keypress
    CPX    #TSKPL+1     ;last offset+1+1
    BNE    STK4         ;if last keypress not processed

;    Self-test memory.

    JSR    DLW          ;delay a long while
    JMP    STM          ;self-test memory

;    Get a keypress.

STK3  LDA    CH         ;key code
    CMP    #$FF         ;clear code indicator
    BEQ    STK3         ;if no key pressed

    CMP    #$C0
    BCS    STK3

    LDA    CH           ;key code

;    Process keypress.

STK4  LDX    #$FF       ;clear code indicator
    STX    CH           ;key code
    PHA                    ;save key code
    AND    #$80
    BEQ    STK5         ;if not CTRL

    LDX    #8           ;offset to control key text
    JSR    SSM          ;set screen memory

;    Check for shift key.

STK5  PLA              ;saved key code
    PHA              ;save key code
    AND    #$40
    BEQ    STK6       ;if not shift key

;    Process keyboard shift key display.

    LDX    #5          ;offset to "SH"
    JSR    SSM          ;set screen memory
    LDX    #4          ;offset to "SH"
    JSR    SSM          ;set screen memory

```

```

;      Check for special keys.

STK6   PLA                ;saved key code
      AND    #$3F
      CMP    #$21
      BEQ    KSB          ;if space bar, process display

      CMP    #$2C
      BEQ    KTK          ;if tab key, process display

      CMP    #$34
      BEQ    KBK          ;if backspace key, process display

      CMP    #$0C
      BEQ    KRK          ;if return key, process display

;      Process other key displays.

      TAX
      LDA    TSMC,X       ;key code
      PHA                ;display character
                        ;save display character

      LDA    #low ST3021
      STA    STTMP1       ;screen pointer
      LDA    #high ST3021
      STA    STTMP1+1

;      Find display character in screen memory.

      PLA                ;saved display character
      LDY    #$FF         ;preset offset

STK7   INY
      CMP    (STTMP1),Y
      BNE    STK7         ;if not found

;      Display inverse video.

      LDA    (STTMP1),Y
      EOR    #$80         ;invert video
      STA    (STTMP1),Y

;      Check auto-mode.

STK8   LDA    STAUT       ;auto-mode flag
      BEQ    STK9         ;if not auto-mode

;      Process auto-mode.

      JSR    SBT          ;sound beep tone
      LDX    #20          ;20-VBLANK delay
      JSR    DAW          ;delay a while
      JSR    SAS          ;silence all sounds
      LDX    #10          ;10-VBLANK delay
      JSR    DAW          ;delay a while
      JMP    STK1         ;get next simulated keypress

;      Process manual mode.

STK9   JSR    SBT          ;sound beep tone

STK10  LDA    SKSTAT
      AND    #$04
      BEQ    STK10

      JSR    SAS          ;silence all sounds
      JMP    STK1         ;get next keypress

```



```

**      SBT - Sound Beep Tone
*
*      ENTRY JSR    SBT
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SBT      =      *      ;entry
          LDA      #$64    ;frequency
          STA      AUDF1    ;set frequency
          LDA      #$A8    ;pure tone, half volume
          STA      AUDC1    ;set control
          RTS          ;return

```

```

**      SAS - Silence All Sounds
*
*      ENTRY JSR    SAS
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SAS      =      *      ;entry
          LDA      #0      ;volume 0
          STA      AUDC1    ;silence voice 1
          STA      AUDC2    ;silence voice 2
          STA      AUDC3    ;silence voice 3
          STA      AUDC4    ;silence voice 4
          RTS          ;return

```

```

**      KSB - Process Keyboard Space Bar Display
*
*      ENTRY JSR    KSB
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

KSB      =      *      ;entry
          LDX      #3      ;offset to "S P A C E   B A R" text
          JSR      SSM      ;set screen memory
          JMP      STK8     ;continue

```

```

**      KBK - Process Keyboard Backspace Key Display
*
*      ENTRY JSR    KBK
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

KBK      =      *      ;entry
          LDX      #6      ;offset to "B S" text
          JSR      SSM      ;set screen memory
          JMP      STK8     ;continue

```

```

**      KTK - Process Keyboard Tab Key Display
*
*      ENTRY JSR    KTK
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

KTK      =      *      ;entry
          LDA      #$7F
          STA      ST3052
          STA      ST3052+1
          BNE      STK8      ;continue

```

```

**      KRK - Process Keyboard Return Key Display
*
*      ENTRY JSR    KRK
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

KRK      =      *      ;entry
          LDA      #$32
          STA      ST306D
          LDA      #$34
          STA      ST306D+1
          BNE      STK8      ;continue

```

```

**      TSKP - Table of Simulated Keypresses

```

```

TSKP      DB      $52,$08,$0A,$2B,$28,$0D,$3D,$39,$2D      ;"Copyright"
          DB      $1F,$30,$35,$1A                          ;"1983"
          DB      $7F,$2D,$3F,$28,$0D                      ;"Atari"

TSKPL     =      *-TSKP ;length

```

```

**      STV - Self-test Audio-visual
*
*      STV verifies the operation of the display and voices by
*      displaying and playing a tune.
*
*      ENTRY JSR    STV
*
*      MODS  M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

STV      =      *      ;entry

;      Initialize.

          LDX      #2      ;audio-visual test colors
          JSR      SUC      ;set up colors

;      Test audio-visual.

STV1     LDA      #0
          STA      STVOC      ;initialize voice indicator

```

```

;      Test voice.

STV2  LDA    #0
      STA    STNOT      ;initialize note counter
      LDX    #low DISL5 ;audio-visual display list
      LDY    #high DISL5
      LDA    #0          ;indicate not keyboard self-test
      JSR    SDL          ;set up display list

;      Display voice number.

      LDX    #9          ;offset to "VOICE #" text
      JSR    SSM          ;set screen memory
      LDA    STVOC        ;voice indicator
      LSR    A            ;voice number
      CLC
      ADC    #$11         ;adjust for screen memory
      STA    ST300B       ;voice number display

;      Display staff.

      LDX    #$0F        ;offset to last byte of staff lines

STV3  LDA    #$FF         ;color 2
      STA    ST3150,X     ;byte of first line of staff
      STA    ST31B0,X     ;byte of second line of staff
      STA    ST3210,X     ;byte of third line of staff
      STA    ST3270,X     ;byte of fourth line of staff
      STA    ST32D0,X     ;byte of fifth line of staff
      DEX
      BPL    STV3         ;if not done

;      Display cleft.

      LDA    #0          ;offset to first cleft display address
      STA    STCDI        ;cleft display pointer
      LDA    #2*6
      STA    STCDA        ;cleft data pointer

STV4  LDX    STCDI        ;cleft display pointer
      LDA    TCDA+1,X     ;high address of cleft display
      TAY
      LDA    TCDA,X       ;low address of cleft display
      TAX
      LDA    STCDA        ;cleft data pointer
      JSR    DVN
      CLC
      LDA    STCDA        ;cleft data pointer
      ADC    #6
      STA    STCDA        ;update cleft data pointer
      INC    STCDI        ;increment cleft display pointer
      INC    STCDI
      LDA    STCDI        ;cleft display pointer
      CMP    #TCDAL       ;length of cleft display table
      BNE    STV4         ;if not done

;      Delay.

      JSR    DMW          ;delay a middling while

;      Display and play first note.

      LDX    #low ST3154
      LDY    #high ST3154
      LDA    #0*6
      JSR    DVN

```

```

    LDA    #$51            ;first note frequency
    JSR    SVN

;    Display and play second note.

    LDX    #low ST3186
    LDY    #high ST3186
    LDA    #0*6
    JSR    DVN

    LDA    #$5B            ;second note frequency
    JSR    SVN

;    Display and play third note.

    LDX    #low ST30F8
    LDY    #high ST30F8
    LDA    #12*6
    JSR    DVN
    LDX    #low ST30C7
    LDY    #high ST30C7
    LDA    #14*6
    JSR    DVN
    LDX    #low ST3248
    LDY    #high ST3248
    LDA    #13*6
    JSR    DVN

    LDA    #$44            ;third note frequency
    JSR    SVN

;    Display and play fourth note.

    LDX    #low ST30CA
    LDY    #high ST30CA
    LDA    #12*6
    JSR    DVN
    LDX    #low ST321A
    LDY    #high ST321A
    LDA    #13*6
    JSR    DVN
    LDX    #low ST31CA
    LDY    #high ST31CA
    LDA    #1*6
    JSR    DVN

    LDA    #$3C            ;fourth note frequency
    JSR    SVN

;    Display and play fifth note.

    LDX    #low ST303C
    LDY    #high ST303C
    LDA    #12*6
    JSR    DVN
    LDX    #low ST318C
    LDY    #high ST318C
    LDA    #13*6
    JSR    DVN
    LDX    #low ST313C
    LDY    #high ST313C
    LDA    #1*6
    JSR    DVN

    LDA    #$2D            ;fifth note frequency
    JSR    SVN

```

```

;      Display and play sixth note.

LDX    #low ST309E
LDY    #high ST309E
LDA    #12*6
JSR    DVN
LDX    #low ST31EE
LDY    #high ST31EE
LDA    #13*6
JSR    DVN

LDA    #$35          ;sixth note frequency
JSR    SVN

;      Delay.

JSR    DLW          ;delay a long while

;      Advance to next voice.

INC     STVOC        ;increment voice indicator
INC     STVOC
LDA     STVOC        ;voice indicator
CMP     #8           ;last voice indicator
BNE     STV5         ;if all voices not processed

;      Process test completion.

LDA     STAUT        ;auto-mode flag
BNE     STV6         ;if auto-mode, perform keyboard test

JMP     STV1         ;repeat audio-visual test

;      Test next voice.

STV5    JMP     STV2         ;test next voice

;      Self-test keyboard.

STV6    JSR     DLW        ;delay a long while
        JSR     STK        ;self-test keyboard

```

```

**      SVN - Sound Tone
*
*      ENTRY JSR     SVN
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SVN     =      *          ;entry

;      Sound note.

LDY     STVOC          ;current voice indicator
STA     AUDF1,Y        ;set frequency
LDA     #$A8           ;pure tone, half volume
STA     AUDC1,Y        ;set control

;      Delay a while.

LDX     STNOT          ;current note
LDA     TNDD,X         ;delay time

```

```

TAX
JSR    DAW            ;delay a while

;      Increment note counter.

INC     STNOT          ;increment note counter

;      Exit.

JSR     SAS            ;silence all sounds
RTS

```

```

**      DVN - Display
*
*      ENTRY JSR    DVN
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DVN      =      *            ;entry
          STX     STTMP2
          STY     STTMP2+1
          TAX
          LDY     #0
          LDA     #16
          STA     STTMP3
          LDA     #6
          STA     STTMP4

DVN1     LDA     TAVD,X
          ORA     (STTMP2),Y
          STA     (STTMP2),Y
          JSR     AST            ;add 16
          DEC     STTMP3
          BNE     DVN1

          INC     STTMP3
          INX
          DEC     STTMP4
          BNE     DVN1

          RTS            ;return

```

```

**      AST - Add Sixteen
*
*      ENTRY JSR    AST
*
*      MODS
*          M. W. Colburn      1982-10-26
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

AST      =      *            ;entry
          CLC
          LDA     STTMP2        ;current low value
          ADC     #16           ;add 16
          STA     STTMP2        ;new low value
          BCC     AST1          ;if no carry

          INC     STTMP2+1      ;adjust high value

AST1     RTS            ;return

```

**** TNDD - Table of Note Duration Delays**

TNDD	DB	32	;0 - first note
	DB	32	;1 - second note
	DB	32	;2 - third note
	DB	16	;3 - fourth note
	DB	16	;4 - fifth note
	DB	32	;5 - sixth note

**** TAVD - Table of Audio-visual Test Display Data**

TAVD	DB	\$01,\$1F,\$3F,\$7F,\$3E,\$1C	;0
	DB	\$00,\$41,\$42,\$4C,\$70,\$40	;1
	DB	\$00,\$01,\$02,\$04,\$08,\$10	;2
	DB	\$00,\$43,\$44,\$48,\$48,\$48	;3
	DB	\$00,\$44,\$22,\$10,\$08,\$07	;4
	DB	\$00,\$04,\$08,\$05,\$02,\$00	;5
	DB	\$00,\$30,\$48,\$88,\$84,\$84	;6
	DB	\$00,\$88,\$88,\$90,\$A0,\$C0	;7
	DB	\$00,\$F0,\$88,\$84,\$82,\$82	;8
	DB	\$00,\$82,\$82,\$84,\$88,\$F0	;9
	DB	\$00,\$00,\$00,\$00,\$00,\$80	;10
	DB	\$80,\$80,\$80,\$80,\$80,\$80	;11
	DB	\$00,\$1C,\$3E,\$7F,\$7E,\$7C	;12
	DB	\$40,\$00,\$00,\$00,\$00,\$00	;13
	DB	\$00,\$04,\$04,\$06,\$05,\$06	;14

**** TCDA - Table of Cleft Display Addresses**

TCDA	DW	ST30C1 ;0
	DW	ST3121 ;1
	DW	ST3181 ;2
	DW	ST31F1 ;3
	DW	ST3002 ;4
	DW	ST3062 ;5
	DW	ST3122 ;6
	DW	ST3182 ;7
	DW	ST30C2 ;8
	DW	ST31C2 ;9

TCDAL = *-TCDA ;length

```

** SVR - Set Value in Range
*
* ENTRY JSR SVR
*       A = value to set
*       X = offset to TARS range
*
* EXIT
*       A = value set
*
* MODS
*       M. W. Colburn      1982-10-26
*       1. Bring closer to Coding Standard (object unchanged).
*       R. K. Nordin      1983-11-01

```

SVR = * ;entry

; Initialize.

PHA ;save value

```

;      Set address range.

      LDA    TARS,X          ;start of range
      STA    STADR1
      LDA    TARS+1,X
      STA    STADR1+1
      LDA    TARS+2,X        ;end of range
      STA    STADR2
      LDA    TARS+3,X
      STA    STADR2+1

;      Set value in range.

      LDY    #0              ;offset to first byte

SVR1   PLA          ;saved value
      STA    (STADR1),Y      ;byte of range
      INC    STADR1          ;increment low address
      BNE    SVR2            ;if no carry

      INC    STADR1+1        ;adjust high address

SVR2   PHA          ;save value
      LDA    STADR1          ;low current address
      CMP    STADR2          ;low end of range
      BNE    SVR1            ;if definitely not done

      LDA    STADR1+1        ;high current address
      CMP    STADR2+1        ;high end of range
      BNE    SVR1            ;if not done

;      Exit.

      PLA          ;restore value
      RTS          ;return

```

```

**      SSM - Set Screen Memory
*
*      ENTRY JSR    SSM
*
*      MODS
*      M. W. Colburn      1982-10-26
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin       1983-11-01

```

```

SSM    =      *          ;entry
      LDA    TST0,X      ;offset to source
      TAY
      LDA    TSTL,X      ;length of source
      STA    STADR1      ;length
      LDA    TSTD,X      ;offset to destination
      TAX

SSM1   LDA    TTX0,Y      ;byte of source
      STA    ST3000,X     ;byte of destination
      INY
      INX
      DEC    STADR1      ;decrement length
      BNE    SSM1        ;if not done

      RTS                ;return

```



```

**      SUC - Set Up Colors
*
*      ENTRY JSR      SUC
*              X = 0, if main screen colors
*              = 1, if memory test colors
*              = 2, if keyboard test colors
*              = 3, if audio-visual test colors
*
*      EXIT
*              COLOR0, COLOR1, COLOR2 and COLOR4 set.
*
*      CHANGES
*              A
*
*      CALLS
*              -none-
*
*      MODS
*              M. W. Colburn      1982-10-26
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

SUC = * ;entry

LDA SUCA,X
STA COLOR0 ;playfield 0 color

LDA SUCB,X
STA COLOR1 ;playfield 1 color

LDA SUCC,X
STA COLOR2 ;playfield 2 color

LDA SUCD,X
STA COLOR4 ;background color

RTS ;return

SUCA DB \$2C ;0 - main screen playfield 0 color
DB \$0C ;1 - memory test playfield 0 color
DB \$2A ;2 - keyboard test playfield 0 color
DB \$18 ;3 - audio-visual test playfield 0 color

SUCB DB \$0F ;0 - main screen playfield 1 color
DB \$32 ;1 - memory test playfield 1 color
DB \$0C ;2 - keyboard test playfield 1 color
DB \$0E ;3 - audio-visual test playfield 1 color

SUCC DB \$D2 ;0 - main screen playfield 2 color
DB \$D6 ;1 - memory test playfield 2 color
DB \$00 ;2 - keyboard test playfield 2 color
DB \$B4 ;3 - audio-visual test playfield 2 color

SUCD DB \$D2 ;0 - main screen background color
DB \$A0 ;1 - memory test background color
DB \$30 ;2 - keyboard test background color
DB \$B4 ;3 - audio-visual test background color

```

**      TSMC - Table of Screen Memory Character Codes
*
*      Entry n is the screen memory character code for key code n.

```

```

TSMC  DB    $2C    ;$00 - L key
      DB    $2A    ;$01 - J key
      DB    $1B    ;$02 - semicolon key
      DB    $91    ;$03
      DB    $92    ;$04
      DB    $2B    ;$05 - K key
      DB    $0B    ;$06 - plus key
      DB    $0A    ;$07 - asterisk key
      DB    $2F    ;$08 - 0 key
      DB    $00    ;$09
      DB    $30    ;$0A - P key
      DB    $35    ;$0B - U key
      DB    $B2    ;$0C - RETURN key
      DB    $29    ;$0D - I key
      DB    $0D    ;$0E - minus key
      DB    $1D    ;$0F - = key

      DB    $36    ;$10 - V key
      DB    $A8    ;$11
      DB    $23    ;$12 - C key
      DB    $93    ;$13
      DB    $94    ;$14
      DB    $22    ;$15 - B key
      DB    $38    ;$16 - X key
      DB    $3A    ;$17 - Z key
      DB    $14    ;$18 - 4 key
      DB    $00    ;$19
      DB    $13    ;$1A - 3 key
      DB    $16    ;$1B - 6 key
      DB    $5B    ;$1C - ESC key
      DB    $15    ;$1D - 5 key
      DB    $12    ;$1E - 2 key
      DB    $11    ;$1F - 1 key

      DB    $0C    ;$20 - comma key
      DB    $00    ;$21 - space key
      DB    $0E    ;$22 - period key
      DB    $2E    ;$23 - N key
      DB    $00    ;$24
      DB    $2D    ;$25 - M key
      DB    $0F    ;$26 - / key
      DB    $A1    ;$27 - inverse video key
      DB    $32    ;$28 - R key
      DB    $00    ;$29
      DB    $25    ;$2A - E key
      DB    $39    ;$2B - Y key
      DB    $FF    ;$2C - TAB key
      DB    $34    ;$2D - T key
      DB    $37    ;$2E - W key
      DB    $31    ;$2F - Q key

      DB    $19    ;$30 - 9 key
      DB    $00    ;$31
      DB    $10    ;$32 - 0 key
      DB    $17    ;$33 - 7 key
      DB    $A2    ;$34 - backspace key
      DB    $18    ;$35 - 8 key
      DB    $1C    ;$36 - < key
      DB    $1E    ;$37 - > key
      DB    $26    ;$38 - F key
      DB    $28    ;$39 - H key

```

DB	\$24	;\$3A - D key
DB	\$00	;\$3B
DB	\$A3	;\$3C - CAPS key
DB	\$27	;\$3D - G key
DB	\$33	;\$3E - S key
DB	\$21	;\$3F - A key

** TARS - Table of Address Ranges to Set

TARS	DW	ST3000,ST3000+\$0EFF	;0 - screen memory
	DW	ST3020,ST3020+4	;1 - memory test first 8K ROM
	DW	ST3024,ST3024+4	;2 - memory test second 8K ROM
	DW	ST3000,ST3000+32	;3 - main screen bold lines

** TSTL - Table of Self-test Text Lengths
--

TSTL	DB	TXT0L	;0 - length of "MEMORY TEST ROM" text
	DB	TXT1L	;1 - length of "RAM" text
	DB	TXT2L	;2 - length of "KEYBOARD TEST" text
	DB	TXT3L	;3 - length of "S P A C E B A R" text
	DB	TXT4L	;4 - length of "SH" text
	DB	TXT5L	;5 - length of "SH" text
	DB	TXT6L	;6 - length of "B S" text
	DB	TXT7L	;7 - length of keyboard text
	DB	TXT8L	;8 - length of control key text
	DB	TXT9L	;9 - length of "VOICE #" text

** TSTD - Table of Self-test Text Destination Offsets
--

TSTD	DB	ST3000-ST3000	;0 - offset to "MEMORY TEST ROM" text
	DB	ST3028-ST3000	;1 - offset to "RAM" text
	DB	ST3000-ST3000	;2 - offset to "KEYBOARD TEST" text
	DB	ST30B7-ST3000	;3 - offset to "S P A C E B A R" text
	DB	ST3092-ST3000	;4 - offset to "SH" text
	DB	ST30AB-ST3000	;5 - offset to "SH" text
	DB	ST304C-ST3000	;6 - offset to "B S" text
	DB	ST3022-ST3000	;7 - offset to keyboard text
	DB	ST3072-ST3000	;8 - offset to control key text
	DB	ST3004-ST3000	;9 - offset to "VOICE #" text

Floating Point Package

```
*** (C) Copyright 1978 Shepardson Microsystems, Inc.
```

```
FIX $D800
```

```
*** FPP - Floating Point Package
*
* FPP is a collection of routines for floating point
* computations. A floating point number is represented
* in 6 bytes:
*
* Byte 0
*      Bit 7      Sign of mantissa
*      Bits 0 - 6  BCD exponent, biased by $40
*
* Bytes 1 - 5      BCD mantissa
*
* MODS
*      Shepardson Microsystems
*
*      Produce 2K version.
*      M. Lorenzen 1981-09-06
```

```
FIX AFP
```

```
** AFP - Convert ASCII to Floating Point
*
* ENTRY JSR AFP
*      INBUFF = line buffer pointer
*      CIX = offset to first byte of number
*
* EXIT
*      C clear, if valid number
*      C set, if invalid number
*
* NOTES
*      Problem: bytes wasted by check for "-", near AFP7.
*
* MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin 1983-11-01
```

```
;AFP = * ;entry

; Initialize.

JSR SLB ;skip leading blanks

; Check for number.

JSR TVN ;test for valid number character
BCS AFP5 ;if not number character

; Set initial values.

LDX #EEXP ;exponent
LDY #4 ;indicate 4 bytes to clear
JSR ZXLY
LDX #$FF
STX DIGRT ;number of digits after decimal point
```

```

    JSR    ZFR0    ;zero FR0
    BEQ    AFP2    ;get first character

;    Indicate not first character.

AFP1    LDA    #$FF    ;indicate not first character
        STA    FCHFLG ;first character flag

;    Get next character.

AFP2    JSR    GNC    ;get next character
        BCS    AFP6    ;if character not numeric

;    Process numeric character.

        PHA                    ;save digit
        LDX    FR0M            ;first byte
        BNE    AFP3            ;if not zero

        JSR    S0L            ;shift FR0 left 1 digit
        PLA                    ;saved digit
        ORA    FR0M+FMPREC-1 ;insert into last byte
        STA    FR0M+FMPREC-1 ;update last byte

;    Check for decimal point.

        LDX    DIGRT ;number of digits after decimal point
        BMI    AFP1    ;if no decimal point, process next character

;    Increment number of digits after decimal point.

        INX                    ;increment number of digits
        STX    DIGRT ;number of digits after decimal point
        BNE    AFP1    ;process next character

;    Increment exponent, if necessary.

AFP3    PLA                    ;clean stack
        LDX    DIGRT ;number of digits after decimal point
        BPL    AFP4            ;if already have decimal point

        INC    EEXP    ;increment number of digits more than 9

;    Process next character.

AFP4    JMP    AFP1    ;process next character

;    Exit.

AFP5    RTS                    ;return

;    Process non-numeric character.

AFP6    CMP    #'.'
        BEQ    AFP8    ;if ".", process decimal point

        CMP    #'E'
        BEQ    AFP9    ;if "E", process exponent

        LDX    FCHFLG ;first character flag
        BNE    AFP16    ;if not first character, process end of input

        CMP    #'+'
        BEQ    AFP1    ;if "+", process next character

        CMP    #'-'
        BEQ    AFP7    ;if "-", process negative sign

```

```

;      Process negative sign.
AFP7  STA    NSIGN ;sign of number
      BEQ    AFP1  ;process next character

;      Process decimal point.
AFP8  LDX    DIGRT ;number of digits after decimal point
      BPL    AFP16 ;if already have decimal point

      INX          ;zero
      STX    DIGRT ;number of digits after decimal point
      BEQ    AFP1  ;process next character

;      Process exponent.
AFP9  LDA    CIX    ;offset to character
      STA    FRX    ;save offset to character
      JSR    GNC    ;get next character
      BCS    AFP13  ;if not numeric

;      Process numeric character in exponent.
AFP10 TAX          ;first character of exponent
      LDA    EEXP   ;number of digits more than 9
      PHA          ;save number of digits more than 9
      STX    EEXP   ;first character of exponent

;      Process second character of exponent.
      JSR    GNC    ;get next character
      BCS    AFP11  ;if not numeric, no second digit

      PHA          ;save second digit
      LDA    EEXP   ;first digit
      ASL    A      ;2 times first digit
      STA    EEXP   ;2 times first digit
      ASL    A      ;4 times first digit
      ASL    A      ;8 times first digit
      ADC    EEXP   ;add 2 times first digit
      STA    EEXP   ;save 10 times first digit
      PLA          ;saved second digit
      CLC
      ADC    EEXP   ;insert in exponent
      STA    EEXP   ;update exponent

;      Process third character of exponent.
      LDY    CIX    ;offset to third character
      JSR    ICX    ;increment offset
AFP11 LDA    ESIGN  ;sign of exponent
      BEQ    AFP12  ;if no sign on exponent

;      Process negative exponent.
      LDA    EEXP   ;exponent
      EOR    #$FF   ;complement exponent
      CLC
      ADC    #1     ;add 1 for 2's complement
      STA    EEXP   ;update exponent

;      Add in number of digits more than 9.
AFP12 PLA          ;saved number of digits more than 9
      CLC

```

```

        ADC    EEXP    ;add exponent
        STA    EEXP    ;update exponent
        BNE    AFP16    ;process end of input

;      Process non-numeric in exponent.

AFP13  CMP     #'+'
        BEQ    AFP14    ;if "+", process next character

        CMP     #'-'
        BNE    AFP15

        STA     ESIGN    ;save sign of exponent

;      Process next character.

AFP14  JSR     GNC      ;get next character
        BCC    AFP10    ;if numeric, process numeric character

;      Process other non-numeric in exponent.

AFP15  LDA     FRX      ;saved offset
        STA     CIX      ;restore offset

;      Process end of input.

AFP16  DEC     CIX      ;decrement offset
        LDA     EEXP      ;exponent
        LDX     DIGRT    ;number of digits after decimal point
        BMI    AFP17    ;if no decimal point

        BEQ    AFP17    ;if no digits after decimal point

        SEC
        SBC     DIGRT    ;subtract number of digits after decimal point

AFP17  PHA
        ROL     A        ;set C with sign of exponent
        PLA
        ROR     A        ;shift right
        STA     EEXP      ;save power of 100
        BCC    AFP18    ;if no carry, process even number

        JSR     S0L      ;shift FR0 left 1 digit

AFP18  LDA     EEXP      ;exponent
        CLC
        ADC     #$40+4    ;add bias plus 4 for normalization
        STA     FR0      ;save exponent

        JSR     NORM      ;normalize number
        BCS    AFP20    ;if error

;      Check sign of number.

        LDX     NSIGN    ;sign of number
        BEQ    AFP19    ;if sign of number not negative

;      Process negative number.

        LDA     FR0      ;first byte of mantissa
        ORA     #$80      ;indicate negative
        STA     FR0      ;update first byte of mantissa

;      Exit.

AFP19  CLC                ;indicate valid number

```

AFP20 **RTS** ;return

FIX FASC

```
** FASC - Convert Floating Point Number to ASCII
*
* ENTRY JSR FASC
* FR0 - FR0+5 = number to convert
*
* EXIT
* INBUFF = pointer to start of number
* High order bit of last character set
*
* MODS
* Original Author Unknown
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01
```

;FASC = * ;entry

; Initialize.

```
JSR ILP ;initialize line buffer pointer
LDA #'0'
STA LBPR2 ;put "0" in front of line buffer
```

; Check for E format required.

```
LDA FR0 ;exponent
BEQ FASC2 ;if exponent zero, number zero

AND #$7F ;clear sign
CMP #$40-1 ;bias-1
BCC FASC3 ;if exponent < bias-1, E format required

CMP #$40+5 ;bias+5
BCS FASC3 ;if >= bias+5, E format required
```

; Process E format not required.

```
SEC
SBC #$40-1 ;subtract bias-1, yielding decimal position
JSR C0A ;convert FR0 to ASCII
JSR FNZ ;find last non-zero character
ORA #$80 ;set high order bit
STA LBUFF,X ;update last character
LDA LBUFF ;first character
CMP # '.'
BEQ FASC1 ;if decimal point
```

```
JMP FASC10
```

```
FASC1 JSR DLP ;decrement line buffer pointer
JMP FASC11 ;perform final adjustment
```

; Process zero.

```
FASC2 LDA #$80+'0' ;"0" with high order bit set
STA LBUFF ;put zero character in line buffer
RTS ;return
```

; Process E format required.

```
FASC3 LDA #1
```



```

    JSR    C0A    ;convert FR0 to ASCII
    JSR    FNZ    ;find last non-zero character
    INX     ;increment offset to last character
    STX    CIX    ;save offset to last character

;    Adjust exponent.

    LDA    FR0    ;exponent
    ASL    A      ;double exponent
    SEC
    SBC    #$40*2 ;subtract 2 times bias

;    Check first character for "0".

    LDX    LBUFF ;first character
    CPX    #'0'
    BEQ    FASC5 ;if "0"

;    Put decimal after first character.

    LDX    LBUFF+1 ;second character
    LDY    LBUFF+2 ;decimal point
    STX    LBUFF+2 ;decimal point
    STY    LBUFF+1 ;third character
    LDX    CIX     ;offset
    CPX    #2      ;former offset to decimal point
    BNE    FASC4   ;if offset pointed to second character

    INC    CIX     ;increment offset

FASC4 CLC
    ADC    #1      ;adjust exponent for movement of decimal point

;    Convert exponent to ASCII.

FASC5 STA    EEXP   ;exponent
    LDA    #'E'
    LDY    CIX     ;offset
    JSR    SAL     ;store ASCII character in line buffer
    STY    CIX     ;save offset
    LDA    EEXP     ;exponent
    BPL    FASC6   ;if exponent positive

    LDA    #0
    SEC
    SBC    EEXP     ;complement exponent
    STA    EEXP     ;update exponent
    LDA    #'-'
    BNE    FASC7   ;store "-"

FASC6 LDA    #'+'

FASC7 JSR    SAL     ;store ASCII character in line buffer
    LDX    #0       ;initial number of 10's
    LDA    EEXP     ;exponent

FASC8 SEC
    SBC    #10      ;subtract 10
    BCC    FASC9    ;if < 0, done

    INX     ;increment number of 10's
    BNE    FASC8    ;continue

FASC9 CLC
    ADC    #10      ;add back 10
    PHA     ;save remainder
    TXA     ;number of 10's

```

```

    JSR    SNL    ;store number in line buffer
    PLA     ;saved remainder
    ORA    #$80   ;set high order bit
    JSR    SNL    ;store number in line buffer

;    Perform final adjustment.

FASC10 LDA    LBUF  ;first character
    CMP    #'0'
    BNE    FASC11

;    Increment pointer to point to non-zero character.

    CLC
    LDA    INBUFF    ;line buffer pointer
    ADC    #1         ;add 1
    STA    INBUFF    ;update line buffer pointer
    LDA    INBUFF+1
    ADC    #0
    STA    INBUFF+1

;    Check for positive exponent.

FASC11 LDA    FR0     ;exponent
    BPL    FASC12     ;if exponent positive, exit

;    Process negative exponent.

    JSR    DLP        ;decrement line buffer pointer
    LDY    #0         ;offset to first character
    LDA    #'-'
    STA    (INBUFF),Y ;put "-" in line buffer

;    Exit.

FASC12 RTS          ;return

    FIX    IFP

```

```

**    IFP - Convert Integer to Floating Point Number
*
*    ENTRY JSR    IFP
*           FR0 - FR0+1 = integer to convert
*
*    EXIT
*           FR0 - FR0+5 = floating point number
*
*    MODS
*           Original Author Unknown
*           1. Bring closer to Coding Standard (object unchanged).
*           R. K. Nordin    1983-11-01

```

```

;IFP = * ;entry

;    Initialize.

    LDA    FR0     ;low integer
    STA    ZTEMP4+1 ;save low integer
    LDA    FR0+1   ;high integer
    STA    ZTEMP4   ;save high integer
    JSR    ZFR0     ;zero FR0

;    Convert to floating point.

    SED

```

```

        LDY    #16            ;number of bits in integer
IFP1    ASL     ZTEMP4+1      ;shift integer
        ROL     ZTEMP4       ;shift integer, setting C if bit present

        LDX    #3            ;offset to last possible byte of number
IFP2    LDA     FR0,X         ;byte of number
        ADC     FR0,X         ;double byte, adding in carry
        STA     FR0,X         ;update byte of number
        DEX
        BNE     IFP2         ;if not done

        DEY
        BNE     IFP1         ;decrement count of integer bits
                                ;if not done

        CLD

;      Set exponent.

        LDA     #$40+2       ;indicate decimal after last digit
        STA     FR0          ;exponent

;      Exit.

        JMP     NORM         ;normalize, return
        FIX     FPI

```

```

**      FPI - Convert Floating Point Number to Integer
*
*      ENTRY JSR    FPI
*              FR0 - FR0+5 = floating point number
*
*      EXIT
*              C set, if error
*              C clear, if no error
*              FR0 - FR0+1 = integer
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin    1983-11-01

```

```

;FPI = * ;entry

;      Initialize.

        LDA     #0
        STA     ZTEMP4       ;zero integer
        STA     ZTEMP4+1

;      Check exponent.

        LDA     FR0          ;exponent
        BMI     FPI4         ;if sign of exponent is negative, error

        CMP     #$40+3       ;bias+3
        BCS     FPI4         ;if number too big, error

        SEC
        SBC     #$40         ;subtract bias
        BCC     FPI2         ;if number less than 1, test for round

;      Compute number of digits to convert.

```

```

ADC    #0          ;add carry
ASL    A           ;2 times exponent-$40+1
STA    ZTEMP1      ;number of digits to convert

; Convert.

FPI1   JSR          ;shift integer left
       BCS          ;if number too big, error

       LDA    ZTEMP4      ;2 times integer
       STA    ZTEMP3      ;save 2 times integer
       LDA    ZTEMP4+1
       STA    ZTEMP3+1
       JSR          ;shift integer left
       BCS          ;if number too big, error

       JSR          ;shift integer left
       BCS          ;if number too big, error

       CLC
       LDA    ZTEMP4+1      ;8 times integer
       ADC    ZTEMP3+1      ;add 2 times integer
       STA    ZTEMP4+1      ;10 times integer
       LDA    ZTEMP4
       ADC    ZTEMP3
       STA    ZTEMP4
       BCS    FPI4          ;if overflow, error

       JSR    GND          ;get next digit
       CLC
       ADC    ZTEMP4+1      ;insert digit
       STA    ZTEMP4+1
       LDA    ZTEMP4
       ADC    #0           ;add carry
       BCS    FPI4          ;if overflow, error

       STA    ZTEMP4
       DEC    ZTEMP1        ;decrement count of digits to convert
       BNE    FPI1         ;if not done

; Check for round required.

FPI2   JSR    GND          ;get next digit
       CMP    #5
       BCC    FPI3         ;if digit less than 5, do not round

; Round.

       CLC
       LDA    ZTEMP4+1
       ADC    #1           ;add 1 to round
       STA    ZTEMP4+1
       LDA    ZTEMP4
       ADC    #0
       STA    ZTEMP4

; Return integer.

FPI3   LDA    ZTEMP4+1      ;low integer
       STA    FR0          ;low integer result
       LDA    ZTEMP4        ;high integer
       STA    FR0+1        ;high integer result
       CLC                ;indicate success
       RTS                ;return

; Return error.

```

```

FPI4  SEC          ;indicate error
      RTS          ;return

      FIX          ZFR0

```

```

**      ZFR0 - Zero FR0
*
*      ENTRY JSR    ZFR0
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

;ZFR0 =      *      ;entry

      LDX      #FR0  ;indicate zero FR0
;      JNE      ZF1   ;zero floating point number, return

      FIX      ZF1

```

```

**      ZF1 - Zero Floating Point Number
*
*      ENTRY JSR    ZF1
*      X = offset to register
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

;ZF1 =      *      ;entry

      LDY      #6    ;number of bytes to zero
;      JNE      ZXLY  ;zero bytes, return

```

```

**      ZXLY - Zero Page Zero Location X for Length Y
*
*      ENTRY JSR    ZXLY
*      X = offset
*      Y = length
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

ZXLY =      *      ;entry

      LDA      #0

ZXLY1 STA      $0000,X      ;zero byte
      INX
      DEY
      BNE      ZXLY1 ;if not done

      RTS          ;return

```

```

**      ILP - Initialize Line Buffer Pointer
*
*      ENTRY JSR      ILP
*
*      EXIT
*          INBUFF - INBUFF+1 = line buffer address
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

ILP      =      *          ;entry
          LDA      #high LBUFF ;high buffer address
          STA      INBUFF+1    ;high line buffer pointer
          LDA      #low LBUFF  ;low buffer address
          STA      INBUFF      ;low line buffer pointer
          RTS

```

```

**      SIL - Shift Integer Left
*
*      ENTRY JSR      SIL
*          ZTEMP4 - ZTEMP4+1 = number (high, low) to shift
*
*      EXIT
*          ZTEMP4 - ZTEMP4+1 shifted left 1
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SIL      =      *          ;entry
          CLC
          ROL      ZTEMP4+1    ;shift low
          ROL      ZTEMP4      ;shift high
          RTS
          FIX      FSUB

```

```

**      FSUB - Perform Floating Point Subtract
*
*      FSUB subtracts FR1 from FR0.
*
*      ENTRY JSR      FSUB
*          FR0 - FR0+5 = minuend
*          FR1 - FR1+5 = subtrahend
*
*      EXIT
*          C set, if error
*          C clear, if no error
*          FR0 - FR0+5 = difference
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

;FSUB =      *          ;entry
;
;      Complement sign of subtrahend and add.
          LDA      FR1      ;subtrahend exponent

```

```

EOR    #$80    ;complement sign of subtrahend
STA    FR1     ;update subtrahend exponent
;       BNE     FADD    ;perform add, return

      FIX      FADD

```

```

**      FADD - Perform Floating Point Add
*
*      ENTRY JSR      FADD
*              FR0 - FR0+5 = augend
*              FR1 - FR1+5 = addend
*
*      EXIT
*              C set, if error
*              C clear, if no error
*              FR0 - FR0+5 = sum
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;FADD =      *      ;entry

;      Initialize.

FADD1 LDA    FR1     ;exponent of addend
      AND    #$7F    ;clear sign of addend mantissa
      STA    ZTEMP4  ;save addend exponent
      LDA    FR0     ;exponent of augend
      AND    #$7F    ;clear sign of augend mantissa
      SEC
      SBC    ZTEMP4  ;subtract addend exponent
      BPL    FADD3   ;if augend exponent >= addend exponent

;      Swap augend and addend.

      LDX    #FPREC-1    ;offset to last byte

FADD2 LDA    FR0,X    ;byte of augend
      LDY    FR1,X    ;byte of addend
      STA    FR1,X    ;move byte of augend to addend
      TYA
      STA    FR0,X    ;move byte of addend to augend
      DEX
      BPL    FADD2    ;if not done

      BMI    FADD1    ;re-initialize

;      Check alignment.

FADD3 BEQ    FADD4    ;if exponent difference zero, already aligned

      CMP    #FMPREC    ;mantissa precision
      BCS    FADD6    ;if exponent difference < mantissa precision

;      Align.

      JSR    S1R      ;shift FR1 right

;      Check for like signs of mantissas.

FADD4 SED
      LDA    FR0     ;augend exponent
      EOR    FR1     ;EOR with addend exponent

```

```

        BMI      FADD8      ;if signs differ, subtract
;      Add.
        LDX      #FMPREC-1      ;offset to last byte of mantissa
        CLC
FADD5   LDA      FR0M,X      ;byte of augend mantissa
        ADC      FR1M,X      ;add byte of addend mantissa
        STA      FR0M,X      ;update byte of result mantissa
        DEX
        BPL      FADD5      ;if not done
        CLD
        BCS      FADD7      ;if carry, process carry
;      Exit.
FADD6   JMP      NORM      ;normalize, return
;      Process carry.
FADD7   LDA      #1      ;indicate shift 1
        JSR      S0R      ;shift FR0 right
        LDA      #1      ;carry
        STA      FR0M      ;set carry in result
;      Exit.
        JMP      NORM      ;normalize, return
;      Subtract.
FADD8   LDX      #FMPREC-1      ;offset to last byte of mantissa
        SEC
FADD9   LDA      FR0M,X      ;byte of augend mantissa
        SBC      FR1M,X      ;subtract byte of addend mantissa
        STA      FR0M,X      ;update byte of result mantissa
        DEX
        BPL      FADD9      ;if not done
        BCC      FADD10     ;if borrow, process borrow
;      Exit.
        CLD
        JMP      NORM      ;normalize, return
;      Process borrow.
FADD10  LDA      FR0      ;result exponent
        EOR      #$80      ;complement sign of result
        STA      FR0      ;update result exponent
        SEC
        LDX      #FMPREC-1      ;offset to last byte of mantissa
FADD11  LDA      #0      ;complement byte of result mantissa
        SBC      FR0M,X      ;update byte of result mantissa
        STA      FR0M,X      ;update byte of result mantissa
        DEX
        BPL      FADD11     ;if not done
;      Exit.
        CLD

```



```

JMP    NORM            ;normalize, return
FIX    FMUL

```

```

**      FMUL - Perform Floating Point Multiply
*
*      ENTRY JSR    FMUL
*              FR0 - FR0+5 = multiplicand
*              FR1 - FR1+5 = multiplier
*
*      EXIT
*              C set, if error
*              C clear, if no error
*              FR0 - FR0+5 = product
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;FMUL =      *      ;entry

;      Check for zero multiplicand.

      LDA    FR0      ;multiplicand exponent
      BEQ    FMUL8     ;if multiplicand exponent zero, result is zero

;      Check for zero multiplier.

      LDA    FR1      ;multiplier exponent
      BEQ    FMUL7     ;if multiplier exponent zero, result is zero

      JSR    SUE      ;set up exponent
      SEC
      SBC    #$40     ;subtract bias
      SEC      ;add 1
      ADC    FR1      ;add multiplier exponent
      BMI    FMUL9     ;if overflow, error

;      Set up.

      JSR    SUP      ;set up

;      Compute number of times to add multiplicand.

FMUL1  LDA    FRE+FPREC-1 ;last byte of FRE
      AND    #$0F      ;extract low order digit
      STA    ZTEMP1+1

;      Check for completion.

FMUL2  DEC    ZTEMP1+1   ;decrement counter
      BMI    FMUL3      ;if done

      JSR    FRA10      ;add FR1 to FR0
      JMP    FMUL2      ;continue

;      Compute number of times to add 10 times multiplicand.

FMUL3  LDA    FRE+FPREC-1 ;last byte of FRE
      LSR    A
      LSR    A
      LSR    A
      LSR    A          ;high order digit
      STA    ZTEMP1+1

```

```

;      Check for completion.
FMUL4 DEC    ZTEMP1+1    ;decrement counter
      BMI    FMUL5      ;if done
      JSR    FRA20      ;add FR2 to FR0
      JMP    FMUL4      ;continue

;      Set up for next set of adds.
FMUL5 JSR    S0ER       ;shift FR0/FRE right

;      Decrement counter and test for completion.
      DEC    ZTEMP1     ;decrement
      BNE    FMUL1      ;if not done

;      Set exponent.
FMUL6 LDA    EEXP       ;exponent
      STA    FR0        ;result exponent
      JMP    N0E        ;normalize, return

;      Return zero result.
FMUL7 JSR    ZFR0       ;zero FR0

;      Return no error.
FMUL8 CLC              ;indicate no error
      RTS               ;return

;      Return error.
FMUL9 SEC              ;indicate error
      RTS               ;return
      FIX    FDIV

```

```

**      FDIV - Perform Floating Point Divide
*
*      ENTRY JSR    FDIV
*              FR0 - FR0+5 = dividend
*              FR1 - FR1+5 = divisor
*
*      EXIT
*              C clear, if no error
*              C set, if error
*              FR0 - FR0+5 = quotient
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;FDIV =      *      ;entry

;      Check for zero divisor.
      LDA    FR1        ;divisor exponent
      BEQ    FMUL9      ;if divisor exponent zero, error

;      Check for zero dividend.

```

```

    LDA    FR0    ;dividend exponent
    BEQ    FMUL8  ;if dividend exponent zero, result is zero

    JSR    SUE    ;set up exponent
    SEC
    SBC    FR1    ;subtract divisor exponent
    CLC
    ADC    #$40   ;add bias
    BMI    FMUL9  ;if overflow, error

    JSR    SUP    ;set up
    INC    ZTEMP1 ;divide requires extra pass
    JMP    FDIV3  ;skip shift

;      Shift FR0/FRE left one byte.

FDIV1 LDX    #0          ;offset to first byte to shift

FDIV2 LDA    FR0+1,X      ;byte to shift
      STA    FR0,X        ;byte of destination
      INX
      CPX    #FMPREC*2+2 ;number of bytes to shift
      BNE    FDIV2        ;if not done

;      Subtract 2 times divisor from dividend.

FDIV3 LDY    #FPREC-1     ;offset to last byte
      SEC
      SED

FDIV4 LDA    FRE,Y        ;byte of dividend
      SBC    FR2,Y        ;subtract byte of 2*divisor
      STA    FRE,Y        ;update byte of dividend
      DEY
      BPL    FDIV4        ;if not done

      CLD
      BCC    FDIV5        ;if difference < 0

      INC    QTEMP        ;increment
      BNE    FDIV3        ;continue

;      Adjust.

FDIV5 JSR    FRA2E ;add FR2 to FR0

;      Shift last byte of quotient left one digit.

      ASL    QTEMP
      ASL    QTEMP
      ASL    QTEMP
      ASL    QTEMP

;      Subtract divisor from dividend.

FDIV6 LDY    #FPREC-1     ;offset to last byte
      SEC
      SED

FDIV7 LDA    FRE,Y        ;byte of dividend
      SBC    FR1,Y        ;subtract byte of divisor
      STA    FRE,Y        ;update byte of dividend
      DEY
      BPL    FDIV7        ;if not done

      CLD
      BCC    FDIV8        ;if difference < 0

```

```

        INC    QTEMP      ;increment
        BNE    FDIV6      ;continue

;    Adjust.

FDIV8   JSR     FRA1E      ;add FR1 to FR0
        DEC    ZTEMP1     ;decrement
        BNE    FDIV1      ;if not done

;    Clear exponent.

        JSR     S0ER       ;shift FR0/FRE right

;    Exit.

        JMP     FMUL6

```

```

**      GNC - Get Next Character
*
*      ENTRY JSR    GNC
*              INBUFF - INBUFF+1 = line buffer pointer
*              CIX = offset to character
*
*      EXIT
*              C set, if character not numeric
*              A = non-numeric character
*              C clear, if character numeric
*              CIX = offset to next character
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

GNC      =      *      ;entry
        JSR     TNC      ;test for numeric character
        LDY     CIX      ;offset
        BCC     ICX      ;if numeric, increment offset, return

        LDA     (INBUFF),Y ;character
;        JSR     ICX      ;increment offset, return

```

```

**      ICX - Increment Character Offset
*
*      ENTRY JSR    ICX
*              Y = offset
*
*      EXIT
*              CIX = offset to next character
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

ICX      =      *      ;entry
        INY     ;increment offset
        STY     CIX     ;offset
        RTS      ;return

```

```

**      SLB - Skip Leading Blanks
*
*      ENTRY JSR      SLB
*              INBUFF - INBUFF+1 = line buffer pointer
*              CIX = offset
*
*      EXIT
*              CIX = offset to first non-blank character
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

SLB      =      *              ;entry

;      Initialize.

      LDY      CIX              ;offset to character
      LDA      #' '

;      Search for first non-blank character.

SLB1     CMP      (INBUFF),Y    ;character
      BNE      SLB2            ;if non-blank character

      INY
      BNE      SLB1            ;if not done

;      Exit.

SLB2     STY      CIX            ;offset to first non-blank character
      RTS                    ;return

```

```

**      TNC - Test for Numeric Character
*
*      ENTRY JSR      TNC
*              INBUFF - INBUFF+1 = line buffer pointer
*              CIX = offset
*
*      EXIT
*              C set, if numeric
*              C clear if non-numeric
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

TNC      =      *              ;entry
      LDY      CIX              ;offset
      LDA      (INBUFF),Y      ;character
      SEC
      SBC      #'0'
      BCC      TVN2            ;if < "0", return failure

      CMP      #'9'-'0'+1      ;return success or failure
      RTS                    ;return

```

```

**      TVN - Test for Valid Number Character
*
*      ENTRY JSR      TVN
*
*      EXIT
*          C set, if not number
*          C clear, if number
*
*      NOTES
*          Problem: bytes wasted by BCC TVN5.
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

TVN      =      *      ;entry
;      Initialize.

      LDA      CIX      ;offset
      PHA      ;save offset
;      Check next character.

      JSR      GNC      ;get next character
      BCC      TVN5     ;if numeric, return success

      CMP      #'.'
      BEQ      TVN4     ;if ".", check next character

      CMP      #'+'
      BEQ      TVN3     ;if "+", check next character

      CMP      #'-'
      BEQ      TVN3     ;if "-", check next character
;      Clean stack.
TVN1     PLA      ;clean stack
;      Return failure.
TVN2     SEC      ;indicate failure
      RTS      ;return
;      Check character after "+" or "-".
TVN3     JSR      GNC      ;get next character
      BCC      TVN5     ;if numeric, return success

      CMP      #'.'
      BNE      TVN1     ;if not ".", return failure
;      Check character after ".".
TVN4     JSR      GNC      ;get next character
      BCC      TVN5     ;if numeric, return success

      BCS      TVN1     ;return failure
;      Return success.
TVN5     PLA      ;saved offset
      STA      CIX      ;restore offset

```

```
CLC      ;indicate success
RTS      ;return
```

```
**      S2L - Shift FR2 Left One Digit
*
*      ENTRY JSR      S2L
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
S2L      =      *      ;entry
LDX      #FR2+1 ;indicate shift of FR2 mantissa
BNE      SML      ;shift mantissa left 1 digit, return
```

```
**      S0L - Shift FR0 Left One Digit
*
*      ENTRY JSR      S0L
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
S0L      =      *      ;entry
LDX      #FR0M    ;indicate shift of FR0 mantissa
; BNE      SML      ;shift mantissa left 1 digit, return
```

```
**      SML - Shift Mantissa Left One Digit
*
*      ENTRY JSR      SML
*
*      EXIT
*          FRX = excess digit
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
SML      =      *      ;entry
LDY      #4      ;number of bits to shift
```

```
SML2     CLC
ROL      $0004,X    ;shift 5th byte left 1 bit
ROL      $0003,X    ;shift 4th byte left 1 bit
ROL      $0002,X    ;shift 3rd byte left 1 bit
ROL      $0001,X    ;shift 2nd byte left 1 bit
ROL      $0000,X    ;shift 1st byte left 1 bit
ROL      FRX        ;shift excess digit left 1 bit
DEY
BNE      SML2      ;if not done
RTS      ;return
```

```

**      NORM - Normalize FR0
*
*      ENTRY JSR    NORM
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

NORM    =      *          ;entry
        LDX     #0
        STX     FRE       ;byte to shift in
;      JMP     N0E        ;normalize FR0/FRE, return

```

```

**      N0E - Normalize FR0/FRE
*
*      ENTRY JSR    N0E
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

N0E      =      *          ;entry
        LDX     #FMPREC-1 ;mantissa size
        LDA     FR0       ;exponent
        BEQ     N0E5      ;if exponent zero, number is zero

N0E1     LDA     FR0M      ;first byte of mantissa
        BNE     N0E3      ;if not zero, no shift

;      Shift mantissa left 1 byte.

        LDY     #0        ;offset to first byte of mantissa

N0E2     LDA     FR0M+1,Y  ;byte to shift
        STA     FR0M,Y    ;byte of destination
        INY
        CPY     #FMPREC   ;size of mantissa
        BCC     N0E2      ;if not done

;      Decrement exponent and check for completion.

        DEC     FR0       ;decrement exponent
        DEX
        BNE     N0E1      ;if not done

;      Check first byte of mantissa.

        LDA     FR0M      ;first byte of mantissa
        BNE     N0E3      ;if mantissa not zero

;      Zero exponent.

        STA     FR0       ;zero exponent
        CLC
        RTS              ;return

;      Check for overflow.

N0E3     LDA     FR0       ;exponent
        AND     #$7F      ;clear sign
        CMP     #$40+49   ;bias+49
        BCC     N0E4      ;if exponent < 49, no overflow

```



```

;      Return error.

;      SEC          ;indicate error
      RTS          ;return

;      Check for underflow.

N0E4   CMP    #$40-49
      BCS    N0E5    ;if exponent >= -49, no underflow

;      Zero result.

      JSR    ZFR0    ;zero FR0

;      Exit.

N0E5   CLC          ;indicate no error
      RTS          ;return

```

```

**      S0R - Shift FR0 Right
*
*      ENTRY JSR    S0R
*              A = shift count
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin    1983-11-01

```

```

S0R    =      *      ;entry
      LDX    #FR0    ;indicate shift of FR0
      BNE    SRR      ;shift register right, return

```

```

**      S1R - Shift FR1 Right
*
*      ENTRY JSR    S1R
*              A = shift count
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin    1983-11-01

```

```

S1R    =      *      ;entry
      LDX    #FR1    ;indicate shift of FR1
;      BNE    SRR      ;shift register right, return

```

```

**      SRR - Shift Register Right
*
*      ENTRY JSR    SRR
*              X = offset to register
*              A = shift count
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin    1983-11-01

```

```

SRR    =      *      ;entry
      STX    ZTEMP3    ;register
      STA    ZTEMP4    ;shift count
      STA    ZTEMP4+1  ;save shift count

```

```

SRR1  LDY    #FMPREC-1    ;mantissa size-1

SRR2  LDA    $0004,X      ;byte to shift
      STA    $0005,X      ;byte of destination
      DEX
      DEY
      BNE    SRR2          ;if not done

      LDA    #0
      STA    $0005,X      ;first byte of mantissa
      LDX    ZTEMP3        ;register
      DEC    ZTEMP4        ;decrement shift count
      BNE    SRR1          ;if not done

;      Adjust exponent.

      LDA    $0000,X      ;exponent
      CLC
      ADC    ZTEMP4+1      ;subtract shift count
      STA    $0000,X      ;update exponent
      RTS                  ;return

```

```

**      S0ER - Shift FR0/FRE Right
*
*      ENTRY JSR    S0ER
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

S0ER  =      *            ;entry
      LDX    #FMPREC*2    ;number of bytes to shift

S0ER1 LDA    FR0,X        ;byte to shift
      STA    FR0+1,X      ;byte of destination
      DEX
      BPL    S0ER1        ;if not done

      LDA    #0
      STA    FR0          ;shift in 0
      RTS                  ;return

```

```

**      C0A - Convert FR0 to ASCII
*
*      ENTRY JSR    C0A
*      A = decimal point position
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

C0A   =      *            ;entry

;      Initialize.

      STA    ZTEMP4 ;decimal point position counter
      LDX    #0      ;offset to first byte of FR0M
      LDY    #0      ;offset to first byte of LBUF

;      Convert next byte.

```

```

C0A1 JSR    TDP    ;test for decimal point
      SEC
      SBC    #1    ;decrement decimal point position
      STA    ZTEMP4 ;update decimal point position counter

;      Convert first digit of next byte.

      LDA    FROM,X ;byte
      LSR    A
      LSR    A
      LSR    A
      LSR    A      ;first digit
      JSR    SNL    ;store number in line buffer

;      Convert second digit of next byte.

      LDA    FROM,X ;byte
      AND    #$0F   ;extract second digit
      JSR    SNL    ;store number in line buffer
      INX
      CPX    #FMPREC ;number of bytes
      BCC    C0A1   ;if not done

;      Exit.

;      JMP    TDP    ;test for decimal point, return

```

```

**      TDP - Test for Decimal Point
*
*      ENTRY JSR    TDP
*              ZTEMP4 = decimal point position counter
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

TDP    =    *      ;entry

;      Check decimal point position counter.

      LDA    ZTEMP4 ;decimal point position counter
      BNE    TDP1   ;if not decimal point position, exit

;      Insert decimal point.

      LDA    #'.'
      JSR    SAL    ;store ASCII character in line buffer

;      Exit.

TDP1   RTS          ;return

```

```

**      SNL - Store Number in Line Buffer
*
*      ENTRY JSR    SNL
*              A = digit to store
*              Y = offset
*
*      EXIT
*              ASCII digit placed in line buffer
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

SNL      =      *      ;entry
          ORA     #$30  ;convert digit to ASCII
          ;      JMP     SAL      ;store ASCII character in line buffer, return

```

```

**      SAL - Store ASCII Character in Line Buffer
*
*      ENTRY JSR    SAL
*              Y = offset
*              A = character
*
*      EXIT
*              Character placed in line buffer
*              Y = incremented offset
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

SAL      =      *      ;entry
          STA     LBUF,Y ;store character in line buffer
          INY     ;increment offset
          RTS     ;return

```

```

**      FNZ - Find Last Non-zero Character in Line Buffer
*
*      FNZ returns the last non-zero character. If the last
*      non-zero character is ".", FNZ returns the character
*      preceding the ".". If no other non-zero character is
*      encountered, FNZ returns the first character.
*
*      ENTRY JSR    FNZ
*
*      EXIT
*              A = character
*              X = offset to character
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

FNZ      =      *      ;entry
;      Initialize.
          LDX     #10    ;offset to last possible character
;      Check next character.

```

```

FNZ1  LDA    LBUF,X      ;character
      CMP    #'.'
      BEQ    FNZ2      ;if ".", return preceding character

      CMP    #'0'
      BNE    FNZ3      ;if not "0", exit

;      Decrement offset and check for completion.

      DEX
      BNE    FNZ1      ;if not done

;      Return character preceding "." or first character.

FNZ2  DEX          ;offset to character
      LDA    LBUF,X      ;character

;      Exit.

FNZ3  RTS          ;return

```

```

**      GND - Get Next Digit
*
*      ENTRY JSR    GND
*              FR0 - FR0+5 = number
*
*      EXIT
*              A = digit
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

GND    =      *      ;entry
      JSR    S0L      ;shift FR0 left 1 digit
      LDA    FRX      ;excess digit
      AND    #$0F     ;extract low order digit
      RTS          ;return

```

```

**      DLP - Decrement Line Buffer Pointer
*
*      ENTRY JSR    DLP
*              INBUFF - INBUFF+1 = line buffer pointer
*
*      EXIT
*              INBUFF - INBUFF+1 = incremented line buffer pointer
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

DLP    =      *      ;entry
      SEC
      LDA    INBUFF    ;line buffer pointer
      SBC    #1        ;subtract 1
      STA    INBUFF    ;update line buffer pointer
      LDA    INBUFF+1
      SBC    #0
      STA    INBUFF+1
      RTS          ;return

```

```

**      SUE - Set Up Exponent for Multiply or Divide
*
*      ENTRY JSR    SUE
*
*      EXIT
*          A = FR0 exponent (without sign)
*          FR1 = FR1 exponent (without sign)
*          FRSIGN = sign of result
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SUE      =      *      ;entry
          LDA      FR0      ;FR0 exponent
          EOR      FR1      ;EOR with FR1 exponent
          AND      #$80     ;extract sign
          STA      FRSIGN   ;sign of result
          ASL      FR1      ;shift out FR1 sign
          LSR      FR1      ;FR1 exponent without sign
          LDA      FR0      ;FR0 exponent
          AND      #$7F     ;FR0 exponent without sign
          RTS

```

```

**      SUP - Set Up for Multiply or Divide
*
*      ENTRY JSR    SUP
*          A = exponent
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SUP      =      *      ;entry
          ORA      FRSIGN   ;place sign in exponent
          STA      EEXP     ;exponent
          LDA      #0
          STA      FR0      ;clear FR0 exponent
          STA      FR1      ;clear FR0 exponent
          JSR      M12      ;move FR1 to FR2
          JSR      S2L      ;shift FR2 left 1 digit
          LDA      FRX      ;excess digit
          AND      #$0F     ;extract low order digit
          STA      FR2      ;shift in low order digit
          LDA      #FMPREC   ;mantissa size
          STA      ZTEMP1   ;mantissa size
          JSR      M0E      ;move FR0 to FRE
          JSR      ZFR0     ;zero FR0
          RTS

```

```

**      FRA10 - Add FR1 to FR0
*
*      ENTRY   JSR      FRA10
*              FR0 - FR0+5 = augend
*              FR1 - FR1+5 = addend
*
*      EXIT
*              FR0 - FR0+5 = sum
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

FRA10 =      *              ;entry
        LDX   #FR0+FPREC-1 ;offset to last byte of FR0
        BNE   F1R

```

```

**      FRA20 - Add FR2 to FR0
*
*      ENTRY   JSR      FRA20
*              FR0 - FR0+5 = augend
*              FR2 - FR2+5 = addend
*
*      EXIT
*              FR0 - FR0+5 = sum
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

FRA20 =      *              ;entry
        LDX   #FR0+FPREC-1 ;offset to last byte of FR0
        BNE   F2R

```

```

**      FRA1E - Add FR1 to FRE
*
*      ENTRY   JSR      FRA1E
*              FRE - FRE+5 = augend
*              FR1 - FR1+5 = addend
*
*      EXIT
*              FRE - FRE+5 = sum
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

FRA1E =      *              ;entry
        LDX   #FRE+FPREC-1 ;offset to last byte of FRE
;        JMP   F1R          ;add FR1 to register, return

```

```

**      F1R - Add FR1 to Register
*
*      ENTRY   JSR      F1R
*              X = offset to last byte of augend register
*              FR1 - FR1+5 = addend
*
*      EXIT
*              Sum in augend register
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

F1R      =      *              ;entry
          LDY      #FR1+FPREC-1 ;offset to last byte of FR1
          BNE      FARR

```

```

**      FRA2E - Add FR2 to FRE
*
*      ENTRY   JSR      FRA2E
*              FRE - FRE+5 = augend
*              FR2 - FR2+5 = addend
*
*      EXIT
*              FRE - FRE+5 = sum
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

FRA2E    =      *              ;entry
          LDX      #FRE+FPREC-1 ;offset to last byte of FRE
          BNE      F2R

```

```

**      F2R - Add FR2 to Register
*
*      ENTRY   JSR      F2R
*              X = offset to last byte of augend register
*              FR2 - FR2+5 = addend
*
*      EXIT
*              Sum in augend register
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

F2R      =      *              ;entry
          LDY      #FR2+FPREC-1 ;offset to last byte of FR2
          BNE      FARR

```



```

**      FARR - Add Register to Register
*
*      ENTRY JSR      FARR
*              X = offset to last byte of augend register
*              Y = offset to last byte of addend register
*
*      EXIT
*              Sum in augend register
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

FARR    =      *              ;entry

;      Initialize.

      LDA      #FPREC-1      ;floating point number size-1
      STA      ZTEMP4        ;byte count
      CLC
      SED

;      Add.

FARR1   LDA      $0000,X      ;byte of augend
      ADC      $0000,Y      ;add byte of addend
      STA      $0000,X      ;update byte of augend
      DEX
      DEY
      DEC      ZTEMP4        ;decrement byte count
      BPL      FARR1        ;if not done

;      Exit.

      CLD
      RTS                  ;return

```

```

**      M12 - Move FR1 to FR2
*
*      ENTRY JSR      M12
*              FR1 - FR1+5 = number to move
*
*      EXIT
*              FR2 - FR2+5 = moved number
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

M12     =      *              ;entry
      LDY      #FPREC-1      ;offset to last byte

M121    LDA      FR1,Y      ;byte of source
      STA      FR2,Y      ;byte of destination
      DEY
      BPL      M121        ;if not done

      RTS                  ;return

```

```

**      M0E - Move FR0 to FRE
*
*      ENTRY JSR      M0E
*              FR0 - FR0+5 = number to move
*
*      EXIT
*              FRE - FRE+5 = moved number
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

M0E      =      *      ;entry
          LDY      #FPREC-1      ;offset to last byte

M0E1     LDA      FR0,Y          ;byte of source
          STA      FRE,Y          ;byte of destination
          DEY
          BPL      M0E1          ;if not done

          RTS                    ;return

          FIX      PLYEVL

```

```

**      PLYEVL - Evaluate Polynomial
*
*      Y = A(0)+A(1)*X+A(2)*X^2+...+A(N)*X^N
*
*      ENTRY JSR      PLYEVL
*              X = low address of coefficient table
*              Y = high address of coefficient table
*              FR0 - FR0+5 = X argument
*              A = N+1
*
*      EXIT
*              FR0 - FR0+5 = Y result
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;PLYEVL      =      *      ;entry

          STX      FPTR2          ;save pointer to coefficients
          STY      FPTR2+1
          STA      PLYCNT          ;degree
          LDX      #low PLYARG
          LDY      #high PLYARG
          JSR      FST0R          ;save argument
          JSR      FMOVE          ;move argument to FR1
          LDX      FPTR2
          LDY      FPTR2+1
          JSR      FLD0R          ;initialize sum in FR0
          DEC      PLYCNT          ;decrement degree
          BEQ      PLY3          ;if complete, exit

PLY1        JSR      FMUL          ;argument times current sum
          BCS      PLY3          ;if overflow

          CLC
          LDA      FPTR2          ;current low coefficient address
          ADC      #FPREC          ;add floating point number size

```

	STA	FPTR2	;update low coefficient address
	BCC	PLY2	;if no carry
	LDA	FPTR2+1	;current high coefficient address
	ADC	#0	;adjust high coefficient address
	STA	FPTR2+1	;update high coefficient address
PLY2	LDX	FPTR2	;low coefficient address
	LDY	FPTR2+1	;high coefficient address
	JSR	FLD1R	;get next coefficient
	JSR	FADD	;add coefficient to argument times sum
	BCS	PLY3	;if overflow
	DEC	PLYCNT	;decrement degree
	BEQ	PLY3	;if complete, exit
	LDX	#low PLYARG	;low argument address
	LDY	#high PLYARG	;high argument address
	JSR	FLD1R	;get argument
	BMI	PLY1	;continue
PLY3	RTS		;return
	FIX	FLD0R	

```

**      FLD0R - Load FR0
*
*      ENTRY JSR      FLD0R
*              X = low pointer
*              Y = high pointer
*
*      EXIT
*              FR0 loaded
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;FLD0R =      *              ;entry
              STX      FLPTR      ;low pointer
              STY      FLPTR+1    ;high pointer
;              JMP      FLD0P      ;load FR0, return
              FIX      FLD0P

```

```

**      FLD0P - Load FR0
*
*      ENTRY JSR      FLD0P
*              FLPTR - FLPTR+1 = pointer
*
*      EXIT
*              FR0 loaded
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;FLD0P =      *              ;entry
              LDY      #FPREC-1   ;offset to last byte
FLD01 LDA      (FLPTR),Y          ;byte of source

```

```

STA    FR0,Y        ;byte of destination
DEY
BPL    FLD01        ;if not done

RTS                                ;return

FIX    FLD1R

```

```

**      FLD1R - Load FR1
*
*      ENTRY JSR    FLD1R
*              X = low pointer
*              Y = high pointer
*
*      EXIT
*              FR1 loaded
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;FLD1R =      *          ;entry
              STX    FLPTR      ;low pointer
              STY    FLPTR+1    ;high pointer
;          LDY    FLD1P      ;load FR1, return
              FIX    FLD1P

```

```

**      FLD1P - Load FR1
*
*      ENTRY JSR    FLD1P
*              FLPTR - FLPTR+1 = pointer
*
*      EXIT
*              FR1 loaded
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;FLD1P =      *          ;entry
              LDY    #FPREC-1    ;offset to last byte

FLD11 LDA    (FLPTR),Y      ;byte of source
              STA    FR1,Y      ;byte of destination
              DEY
              BPL    FLD11      ;if not done
              RTS                                ;return
              FIX    FST0R

```

```

**      FST0R - Store FR0
*
*      ENTRY JSR      FST0R
*              FR0 - FR0+5 = number
*              X = low pointer
*              Y = high pointer
*
*      EXIT
*              FR0 stored
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;FST0R =      *              ;entry
              STX      FLPTR      ;low pointer
              STY      FLPTR+1      ;high pointer
;          LDY      FST0P      ;store FR0, return
              FIX      FST0P

```

```

**      FST0P - Store FR0
*
*      ENTRY JSR      FST0P
*              FR0 - FR0+5 = number
*              FLPTR - FLPTR+1 = pointer
*
*      EXIT
*              FR0 stored
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;FST0P =      *              ;entry
              LDY      #FPREC-1      ;offset to last byte
FST01 LDA      FR0,Y      ;byte of source
              STA      (FLPTR),Y      ;byte of destination
              DEY
              BPL      FST01      ;if not done
              RTS              ;return
              FIX      FMOVE

```

```

**      FMOVE - Move FR0 to FR1
*
*      ENTRY JSR      FMOVE
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;FMOVE =      *              ;entry
              LDX      #FPREC-1      ;offset to last byte
FM01  LDA      FR0,X      ;byte of source

```

```

STA    FR1,X        ;byte of destination
DEX
BPL    FM01          ;if not done
RTS
FIX    EXP

```

```

**      EXP - Compute Power of e
*
*      ENTRY JSR     EXP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

;EXP =      *          ;entry

;      Initialize.

LDX    #low LOG10E    ;base 10 logarithm of e
LDY    #high LOG10E
JSR    FLD1R          ;load FR1

;      Compute X*LOG10(E).

JSR    FMUL           ;multiply
BCS    EXP6           ;if overflow, error

;      Compute result = 10^(X*LOG10(E)).

;      JMP     EXP10    ;compute power of 10, return

FIX    EXP10

```

```

**      EXP10 - Compute Power of 10
*
*      ENTRY JSR     EXP10
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

;EXP10 =      *          ;entry

;      Initialize.

LDA    #0
STA    XFMFLG         ;zero integer part
LDA    FR0
STA    SGNFLG         ;save argument sign
AND    #$7F           ;extract absolute value
STA    FR0            ;update argument

;      Check for argument less than 1.

SEC
SBC    #$40           ;subtract bias
BMI    EXP1           ;if argument < 1

;      Extract integer and fractional parts of exponent.

```

```

CMP    #FPREC-2
BPL    EXP6            ;if argument too big, error

LDX    #low FPSCR
LDY    #high FPSCR
JSR    FST0R           ;save argument
JSR    FPI             ;convert argument to integer
LDA    FR0
STA    XFMFLG          ;save integer part
LDA    FR0+1           ;most significant byte of integer part
BNE    EXP6            ;if integer part too large, error

JSR    IFP             ;convert integer part to floating point
JSR    FMOVE
LDX    #low FPSCR
LDY    #high FPSCR
JSR    FLD0R           ;argument
JSR    FSUB            ;subtract to get fractional part

;    Compute 10 to fractional exponent.

EXP1   LDA    #NPC0EF
LDX    #low P10COF
LDY    #high P10COF
JSR    PLYEVL          ;P(X)
JSR    FMOVE
JSR    FMUL            ;P(X)*P(X)

;    Check integer part.

LDA    XFMFLG          ;integer part
BEQ    EXP4            ;if integer part zero

;    Compute 10 to integer part.

CLC
ROR    A               ;integer part divided by 2
STA    FR1             ;exponent
LDA    #1              ;assume mantissa 1
BCC    EXP2            ;if integer part even

LDA    #$10            ;substitute mantissa 10

EXP2   STA    FR1M      ;mantissa
LDX    #FMPREC-1      ;offset to last byte of mantissa
LDA    #0

EXP3   STA    FR1M+1,X  ;zero byte of mantissa
DEX
BPL    EXP3            ;if not done

LDA    FR1             ;exponent
CLC
ADC    #$40            ;add bias
BCS    EXP6            ;if too big, error

BMI    EXP6            ;if underflow, error

STA    FR1             ;10 to integer part

;    Compute product of 10 to integer part and 10 to fractional part.

JSR    FMUL            ;multiply to get result

;    Invert result if argument < 0.

EXP4   LDA    SGNFLG    ;argument sign

```

```

    BPL    EXP5            ;if argument >= 0

    JSR    FMOVE
    LDX    #low FONE
    LDY    #high FONE
    JSR    FLD0R           ;load FR0
    JSR    FDIV            ;divide to get result

;    Exit.

EXP5     RTS              ;return

;    Return error.

EXP6     SEC              ;indicate error
         RTS              ;return

```

** P10COF - Power of 10 Coefficients

```

P10COF DB    $3D,$17,$94,$19,$00,$00    ;0.0000179419
        DB    $3D,$57,$33,$05,$00,$00    ;0.0000573305
        DB    $3E,$05,$54,$76,$62,$00    ;0.0005547662
        DB    $3E,$32,$19,$62,$27,$00    ;0.0032176227
        DB    $3F,$01,$68,$60,$30,$36    ;0.0168603036
        DB    $3F,$07,$32,$03,$27,$41    ;0.0732032741
        DB    $3F,$25,$43,$34,$56,$75    ;0.2543345675
        DB    $3F,$66,$27,$37,$30,$50    ;0.6627373050
        DB    $40,$01,$15,$12,$92,$55    ;1.15129255
        DB    $3F,$99,$99,$99,$99,$99    ;0.9999999999

```

```

NPCOEF =    [*-P10COF]/FPREC

```

** LOG10E - Base 10 Logarithm of e

```

LOG10E DB    $3F,$43,$42,$94,$48,$19    ;base 10 logarithm of e

```

** FONE - 1.0

```

FONE  DB    $40,$01,$00,$00,$00,$00    ;1.0

```

** XFORM - Transform

```

*
*      Z = (X-C)/(X+C)
*
*      ENTRY JSR    XFORM
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

XFORM =    *              ;entry
          STX    FPTR2
          STY    FPTR2+1
          LDX    #low PLYARG
          LDY    #high PLYARG
          JSR    FST0R      ;save argument
          LDX    FPTR2
          LDY    FPTR2+1
          JSR    FLD1R      ;load FR1
          JSR    FADD        ;X+C
          LDX    #low FPSCR

```



```

LDY    #high FPSCR
JSR     FST0R      ;store FR0
LDX     #low PLYARG
LDY     #high PLYARG
JSR     FLD0R      ;load FR0
LDX     FPTR2
LDY     FPTR2+1
JSR     FLD1R      ;load FR1
JSR     FSUB       ;X-C
LDX     #low FPSCR
LDY     #high FPSCR
JSR     FLD1R      ;load FR1
JSR     FDIV       ;divide to get result
RTS

```

FIX LOG

```

**      LOG - Compute Base e Logarithm
*
*      ENTRY JSR     LOG
*              FR0 - FR0+5 = argument
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;LOG = * ;entry

LDA     #1      ;indicate base e logarithm
BNE     LOGS    ;compute logarithm, return

FIX     LOG10

```

```

**      LOG10 - Compute Base 10 Logarithm
*
*      ENTRY JSR     LOG10
*              FR0 - FR0+5 = argument
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

;LOG10 = * ;entry

LDA     #0      ;indicate base 10 logarithm
INE     LOGS    ;compute logarithm, return
;

```

```

**      LOGS - Compute Logarithm
*
*      ENTRY JSR      LOGS
*              A = 0, if base 10 logarithm
*              = 1, if base e logarithm
*              FR0 - FR0+5 = argument
*
*      EXIT
*              C set, if error
*              C clear, if no error
*              FR0 - FR0+5 = result
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

LOGS    =      *      ;entry

;      Initialize.

      STA      SGNFLG ;save logarithm base indicator

;      Check argument.

      LDA      FR0      ;argument exponent
      BEQ      LOGS1    ;if argument zero, error

      BMI      LOGS1    ;if argument negative, error

;      X = F*(10^Y), 1<F<10
;      10^Y HAS SAME EXP BYTE AS X
;      & MANTISSA BYTE = 1 OR 10

      JMP      LOGQ

;      Return error.

LOGS1   SEC              ;indicate error
      RTS              ;return

```

```

**      LOGC - Complete Computation of Logarithm
*
*      ENTRY JSR      LOGC
*              SGNFLG = 0, if base 10 logarithm
*              = 1, if base e logarithm
*
*      NOTES
*              Problem: logic is convoluted because LOGQ code
*              was moved.
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

LOGC    =      *      ;entry

;      Initialize.

      SBC      #$40
      ASL      A
      STA      XFMFLG      ;save Y

```

```

        LDA    FR0+1
        AND    #$F0
        BNE    LOGC2

        LDA    #1           ;mantissa is 1
        BNE    LOGC3       ;set mantissa

LOGC2   INC    XFMFLG       ;increment Y
        LDA    #$10       ;mantissa is 10

LOGC3   STA    FR1M        ;mantissa
        LDX    #FMPREC-1  ;offset to last byte of mantissa
        LDA    #0

LOGC4   STA    FR1M+1,X    ;zero byte of mantissa
        DEX
        BPL    LOGC4       ;if not done

        JSR    FDIV        ;X = X/(10^Y), S.B. IN (1,10)

;      Compute LOG10(X), 1 <= X <= 10.

        LDX    #low SQR10
        LDY    #high SQR10
        JSR    XFORM       ;Z = (X-C)/(X+C); C*C = 10
        LDX    #low FPSCR
        LDY    #high FPSCR
        JSR    FST0R       ;SAVE Z
        JSR    FMOVE
        JSR    FMUL        ;Z*Z
        LDA    #NLCOEF
        LDX    #low LGCOEF
        LDY    #high LGCOEF
        JSR    PLYEVL      ;P(Z*Z)
        LDX    #low FPSCR
        LDY    #high FPSCR
        JSR    FLD1R       ;load FR1
        JSR    FMUL        ;Z*P(Z*Z)
        LDX    #low FHALF
        LDY    #high FHALF
        JSR    FLD1R
        JSR    FADD        ;0.5 + Z*P(Z*Z)
        JSR    FMOVE
        LDA    #0
        STA    FR0+1
        LDA    XFMFLG
        STA    FR0
        BPL    LOGC5

        EOR    #-$01       ;complement sign
        CLC
        ADC    #1
        STA    FR0

LOGC5   JSR    IFP         ;convert integer to floating point
        BIT    XFMFLG
        BPL    LOGC6

        LDA    #$80
        ORA    FR0
        STA    FR0         ;update exponent

LOGC6   JSR    FADD        ;LOG(X) = LOG(X)+Y

;      Check base of logarithm.

        LDA    SGNFLG      ;logarithm base indicator

```

```

    BEQ    LOGC7          ;if LOG10 (not LOG)

;    Compute base e logarithm.

    LDX    #low LOG10E    ;base 10 logarithm of e
    LDY    #high LOG10E
    JSR    FLD1R          ;load FR1
    JSR    FDIV           ;result is LOG(X) divided by LOG10(e)

;    Exit.

LOGC7 CLC                ;indicate success
    RTS                 ;return

```

** SQR10 - Square Root of 10

```

SQR10 DB    $40,$03,$16,$22,$77,$66    ;square root of 10

```

** FHALF - 0.5

```

FHALF DB    $3F,$50,$00,$00,$00,$00    ;0.5

```

** LGCOEF - Logarithm Coefficients

```

LGCOEF DB    $3F,$49,$15,$57,$11,$08    ;0.4915571108
DB    $BF,$51,$70,$49,$47,$08    ;-0.5170494708
DB    $3F,$39,$20,$57,$61,$95    ;0.3920576195
DB    $BF,$04,$39,$63,$03,$55    ;-0.0439630355
DB    $3F,$10,$09,$30,$12,$64    ;0.1009301264
DB    $3F,$09,$39,$08,$04,$60    ;0.0939080460
DB    $3F,$12,$42,$58,$47,$42    ;0.1242584742
DB    $3F,$17,$37,$12,$06,$08    ;0.1737120608
DB    $3F,$28,$95,$29,$71,$17    ;0.2895297117
DB    $3F,$86,$85,$88,$96,$44    ;0.8685889644

```

```

NLCOE =    [*-LGCOEF]/FPREC

```

** ATCOEF - Arctangent Coefficients

*

* NOTES

*

Problem: not used.

```

DB    $3E,$16,$05,$44,$49,$00    ;0.001605444900
DB    $BE,$95,$68,$38,$45,$00    ;-0.009568384500
DB    $3F,$02,$68,$79,$94,$16    ;0.0268799416
DB    $BF,$04,$92,$78,$90,$80    ;-0.0492789080
DB    $3F,$07,$03,$15,$20,$00    ;0.0703152000
DB    $BF,$08,$92,$29,$12,$44    ;-0.0892291244
DB    $3F,$11,$08,$40,$09,$11    ;0.1108400911
DB    $BF,$14,$28,$31,$56,$04    ;-0.1428315604
DB    $3F,$19,$99,$98,$77,$44    ;0.1999987744
DB    $BF,$33,$33,$33,$31,$13    ;-0.3333333113
DB    $3F,$99,$99,$99,$99,$99    ;0.9999999999

DB    $3F,$78,$53,$98,$16,$34    ;pi/4 = arctan 1

```

```

**      LOGQ - Continue Computation of Logarithm
*
*      ENTRY JSR      LOGQ
*
*      NOTES
*          Problem: logic is convoluted because this code was
*          moved.
*          Problem: for readability, this might be relocated
*          before tables.
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

LOGQ  =      *      ;entry
      LDA     FR0
      STA     FR1
      SEC
      JMP     LOGC    ;complete computation of logarithm, return

```

Domestic Character Set

FIX DCSORG

** Domestic Character Set

DB	\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00	;\$00 - space
DB	\$00,\$18,\$18,\$18,\$18,\$00,\$18,\$00	;\$01 - !
DB	\$00,\$66,\$66,\$66,\$00,\$00,\$00,\$00	;\$02 - "
DB	\$00,\$66,\$FF,\$66,\$66,\$FF,\$66,\$00	;\$03 - #
DB	\$18,\$3E,\$60,\$3C,\$06,\$7C,\$18,\$00	;\$04 - \$
DB	\$00,\$66,\$6C,\$18,\$30,\$66,\$46,\$00	;\$05 - %
DB	\$1C,\$36,\$1C,\$38,\$6F,\$66,\$3B,\$00	;\$06 - &
DB	\$00,\$18,\$18,\$18,\$00,\$00,\$00,\$00	;\$07 - '
DB	\$00,\$0E,\$1C,\$18,\$18,\$1C,\$0E,\$00	;\$08 - (
DB	\$00,\$70,\$38,\$18,\$18,\$38,\$70,\$00	;\$09 -)
DB	\$00,\$66,\$3C,\$FF,\$3C,\$66,\$00,\$00	;\$0A - asterisk
DB	\$00,\$18,\$18,\$7E,\$18,\$18,\$00,\$00	;\$0B - plus
DB	\$00,\$00,\$00,\$00,\$00,\$18,\$18,\$30	;\$0C - comma
DB	\$00,\$00,\$00,\$7E,\$00,\$00,\$00,\$00	;\$0D - minus
DB	\$00,\$00,\$00,\$00,\$00,\$18,\$18,\$00	;\$0E - period
DB	\$00,\$06,\$0C,\$18,\$30,\$60,\$40,\$00	;\$0F - /
DB	\$00,\$3C,\$66,\$6E,\$76,\$66,\$3C,\$00	;\$10 - 0
DB	\$00,\$18,\$38,\$18,\$18,\$18,\$7E,\$00	;\$11 - 1
DB	\$00,\$3C,\$66,\$0C,\$18,\$30,\$7E,\$00	;\$12 - 2
DB	\$00,\$7E,\$0C,\$18,\$0C,\$66,\$3C,\$00	;\$13 - 3
DB	\$00,\$0C,\$1C,\$3C,\$6C,\$7E,\$0C,\$00	;\$14 - 4
DB	\$00,\$7E,\$60,\$7C,\$06,\$66,\$3C,\$00	;\$15 - 5
DB	\$00,\$3C,\$60,\$7C,\$66,\$66,\$3C,\$00	;\$16 - 6
DB	\$00,\$7E,\$06,\$0C,\$18,\$30,\$30,\$00	;\$17 - 7
DB	\$00,\$3C,\$66,\$3C,\$66,\$66,\$3C,\$00	;\$18 - 8
DB	\$00,\$3C,\$66,\$3E,\$06,\$0C,\$38,\$00	;\$19 - 9
DB	\$00,\$00,\$18,\$18,\$00,\$18,\$18,\$00	;\$1A - colon
DB	\$00,\$00,\$18,\$18,\$00,\$18,\$18,\$30	;\$1B - semicolon
DB	\$06,\$0C,\$18,\$30,\$18,\$0C,\$06,\$00	;\$1C - <
DB	\$00,\$00,\$7E,\$00,\$00,\$7E,\$00,\$00	;\$1D - =
DB	\$60,\$30,\$18,\$0C,\$18,\$30,\$60,\$00	;\$1E - >
DB	\$00,\$3C,\$66,\$0C,\$18,\$00,\$18,\$00	;\$1F - ?
DB	\$00,\$3C,\$66,\$6E,\$6E,\$60,\$3E,\$00	;\$20 - @
DB	\$00,\$18,\$3C,\$66,\$66,\$7E,\$66,\$00	;\$21 - A
DB	\$00,\$7C,\$66,\$7C,\$66,\$66,\$7C,\$00	;\$22 - B
DB	\$00,\$3C,\$66,\$60,\$60,\$66,\$3C,\$00	;\$23 - C
DB	\$00,\$78,\$6C,\$66,\$66,\$6C,\$78,\$00	;\$24 - D
DB	\$00,\$7E,\$60,\$7C,\$60,\$60,\$7E,\$00	;\$25 - E
DB	\$00,\$7E,\$60,\$7C,\$60,\$60,\$60,\$00	;\$26 - F
DB	\$00,\$3E,\$60,\$60,\$6E,\$66,\$3E,\$00	;\$27 - G
DB	\$00,\$66,\$66,\$7E,\$66,\$66,\$66,\$00	;\$28 - H
DB	\$00,\$7E,\$18,\$18,\$18,\$18,\$7E,\$00	;\$29 - I
DB	\$00,\$06,\$06,\$06,\$06,\$66,\$3C,\$00	;\$2A - J
DB	\$00,\$66,\$6C,\$78,\$78,\$6C,\$66,\$00	;\$2B - K
DB	\$00,\$60,\$60,\$60,\$60,\$60,\$7E,\$00	;\$2C - L
DB	\$00,\$63,\$77,\$7F,\$6B,\$63,\$63,\$00	;\$2D - M
DB	\$00,\$66,\$76,\$7E,\$7E,\$6E,\$66,\$00	;\$2E - N
DB	\$00,\$3C,\$66,\$66,\$66,\$66,\$3C,\$00	;\$2F - O
DB	\$00,\$7C,\$66,\$66,\$7C,\$60,\$60,\$00	;\$30 - P
DB	\$00,\$3C,\$66,\$66,\$66,\$6C,\$36,\$00	;\$31 - Q
DB	\$00,\$7C,\$66,\$66,\$7C,\$6C,\$66,\$00	;\$32 - R
DB	\$00,\$3C,\$60,\$3C,\$06,\$06,\$3C,\$00	;\$33 - S
DB	\$00,\$7E,\$18,\$18,\$18,\$18,\$18,\$00	;\$34 - T
DB	\$00,\$66,\$66,\$66,\$66,\$66,\$7E,\$00	;\$35 - U
DB	\$00,\$66,\$66,\$66,\$66,\$3C,\$18,\$00	;\$36 - V

DB	\$00,\$63,\$63,\$6B,\$7F,\$77,\$63,\$00	;\$37 - W
DB	\$00,\$66,\$66,\$3C,\$3C,\$66,\$66,\$00	;\$38 - X
DB	\$00,\$66,\$66,\$3C,\$18,\$18,\$18,\$00	;\$39 - Y
DB	\$00,\$7E,\$0C,\$18,\$30,\$60,\$7E,\$00	;\$3A - Z
DB	\$00,\$1E,\$18,\$18,\$18,\$18,\$1E,\$00	;\$3B - [
DB	\$00,\$40,\$60,\$30,\$18,\$0C,\$06,\$00	;\$3C - \
DB	\$00,\$78,\$18,\$18,\$18,\$18,\$78,\$00	;\$3D -]
DB	\$00,\$08,\$1C,\$36,\$63,\$00,\$00,\$00	;\$3E - ^
DB	\$00,\$00,\$00,\$00,\$00,\$00,\$FF,\$00	;\$3F - underline
DB	\$00,\$36,\$7F,\$7F,\$3E,\$1C,\$08,\$00	;\$40 - heart card
DB	\$18,\$18,\$18,\$1F,\$1F,\$18,\$18,\$18	;\$41 - mid left window
DB	\$03,\$03,\$03,\$03,\$03,\$03,\$03,\$03	;\$42 - right box
DB	\$18,\$18,\$18,\$F8,\$F8,\$00,\$00,\$00	;\$43 - low right window
DB	\$18,\$18,\$18,\$F8,\$F8,\$18,\$18,\$18	;\$44 - mid right window
DB	\$00,\$00,\$00,\$F8,\$F8,\$18,\$18,\$18	;\$45 - up right window
DB	\$03,\$07,\$0E,\$1C,\$38,\$70,\$E0,\$C0	;\$46 - right slant box
DB	\$C0,\$E0,\$70,\$38,\$1C,\$0E,\$07,\$03	;\$47 - left slant box
DB	\$01,\$03,\$07,\$0F,\$1F,\$3F,\$7F,\$FF	;\$48 - right slant solid
DB	\$00,\$00,\$00,\$00,\$0F,\$0F,\$0F,\$0F	;\$49 - low right solid
DB	\$80,\$C0,\$E0,\$F0,\$F8,\$FC,\$FE,\$FF	;\$4A - left slant solid
DB	\$0F,\$0F,\$0F,\$0F,\$00,\$00,\$00,\$00	;\$4B - up right solid
DB	\$F0,\$F0,\$F0,\$F0,\$00,\$00,\$00,\$00	;\$4C - up left solid
DB	\$FF,\$FF,\$00,\$00,\$00,\$00,\$00,\$00	;\$4D - top box
DB	\$00,\$00,\$00,\$00,\$00,\$00,\$FF,\$FF	;\$4E - bottom box
DB	\$00,\$00,\$00,\$00,\$F0,\$F0,\$F0,\$F0	;\$4F - low left solid
DB	\$00,\$1C,\$1C,\$77,\$77,\$08,\$1C,\$00	;\$50 - club card
DB	\$00,\$00,\$00,\$1F,\$1F,\$18,\$18,\$18	;\$51 - up left window
DB	\$00,\$00,\$00,\$FF,\$FF,\$00,\$00,\$00	;\$52 - mid box
DB	\$18,\$18,\$18,\$FF,\$FF,\$18,\$18,\$18	;\$53 - mid window
DB	\$00,\$00,\$3C,\$7E,\$7E,\$7E,\$3C,\$00	;\$54 - solid circle
DB	\$00,\$00,\$00,\$00,\$FF,\$FF,\$FF,\$FF	;\$55 - bottom solid
DB	\$C0,\$C0,\$C0,\$C0,\$C0,\$C0,\$C0,\$C0	;\$56 - left box
DB	\$00,\$00,\$00,\$FF,\$FF,\$18,\$18,\$18	;\$57 - up mid window
DB	\$18,\$18,\$18,\$FF,\$FF,\$00,\$00,\$00	;\$58 - low mid window
DB	\$F0,\$F0,\$F0,\$F0,\$F0,\$F0,\$F0,\$F0	;\$59 - left solid
DB	\$18,\$18,\$18,\$1F,\$1F,\$00,\$00,\$00	;\$5A - low left window
DB	\$78,\$60,\$78,\$60,\$7E,\$18,\$1E,\$00	;\$5B - display escape
DB	\$00,\$18,\$3C,\$7E,\$18,\$18,\$18,\$00	;\$5C - up arrow
DB	\$00,\$18,\$18,\$18,\$7E,\$3C,\$18,\$00	;\$5D - down arrow
DB	\$00,\$18,\$30,\$7E,\$30,\$18,\$00,\$00	;\$5E - left arrow
DB	\$00,\$18,\$0C,\$7E,\$0C,\$18,\$00,\$00	;\$5F - right arrow
DB	\$00,\$18,\$3C,\$7E,\$7E,\$3C,\$18,\$00	;\$60 - diamond card
DB	\$00,\$00,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$61 - a
DB	\$00,\$60,\$60,\$7C,\$66,\$66,\$7C,\$00	;\$62 - b
DB	\$00,\$00,\$3C,\$60,\$60,\$60,\$3C,\$00	;\$63 - c
DB	\$00,\$06,\$06,\$3E,\$66,\$66,\$3E,\$00	;\$64 - d
DB	\$00,\$00,\$3C,\$66,\$7E,\$60,\$3C,\$00	;\$65 - e
DB	\$00,\$0E,\$18,\$3E,\$18,\$18,\$18,\$00	;\$66 - f
DB	\$00,\$00,\$3E,\$66,\$66,\$3E,\$06,\$7C	;\$67 - g
DB	\$00,\$60,\$60,\$7C,\$66,\$66,\$66,\$00	;\$68 - h
DB	\$00,\$18,\$00,\$38,\$18,\$18,\$3C,\$00	;\$69 - i
DB	\$00,\$06,\$00,\$06,\$06,\$06,\$06,\$3C	;\$6A - j
DB	\$00,\$60,\$60,\$6C,\$78,\$6C,\$66,\$00	;\$6B - k
DB	\$00,\$38,\$18,\$18,\$18,\$18,\$3C,\$00	;\$6C - l
DB	\$00,\$00,\$66,\$7F,\$7F,\$6B,\$63,\$00	;\$6D - m
DB	\$00,\$00,\$7C,\$66,\$66,\$66,\$66,\$00	;\$6E - n
DB	\$00,\$00,\$3C,\$66,\$66,\$66,\$3C,\$00	;\$6F - o
DB	\$00,\$00,\$7C,\$66,\$66,\$7C,\$60,\$60	;\$70 - p
DB	\$00,\$00,\$3E,\$66,\$66,\$3E,\$06,\$06	;\$71 - q
DB	\$00,\$00,\$7C,\$66,\$60,\$60,\$60,\$00	;\$72 - r
DB	\$00,\$00,\$3E,\$60,\$3C,\$06,\$7C,\$00	;\$73 - s
DB	\$00,\$18,\$7E,\$18,\$18,\$18,\$0E,\$00	;\$74 - t
DB	\$00,\$00,\$66,\$66,\$66,\$66,\$3E,\$00	;\$75 - u

DB	\$00,\$00,\$66,\$66,\$66,\$3C,\$18,\$00	;\$76 - v
DB	\$00,\$00,\$63,\$6B,\$7F,\$3E,\$36,\$00	;\$77 - w
DB	\$00,\$00,\$66,\$3C,\$18,\$3C,\$66,\$00	;\$78 - x
DB	\$00,\$00,\$66,\$66,\$66,\$3E,\$0C,\$78	;\$79 - y
DB	\$00,\$00,\$7E,\$0C,\$18,\$30,\$7E,\$00	;\$7A - z
DB	\$00,\$18,\$3C,\$7E,\$7E,\$18,\$3C,\$00	;\$7B - spade card
DB	\$18,\$18,\$18,\$18,\$18,\$18,\$18,\$18	;\$7C -
DB	\$00,\$7E,\$78,\$7C,\$6E,\$66,\$06,\$00	;\$7D - display clear
DB	\$08,\$18,\$38,\$78,\$38,\$18,\$08,\$00	;\$7E - display backspace
DB	\$10,\$18,\$1C,\$1E,\$1C,\$18,\$10,\$00	;\$7F - display tab

Device Handler Vector Tables

FIX EDITRV

** EDITRV - Editor Handler Vector Table
--

```
DW EOP-1 ;perform editor OPEN
DW ECL-1 ;perform editor CLOSE
DW EGB-1 ;perform editor GET-BYTE
DW EPB-1 ;perform editor PUT-BYTE
DW SST-1 ;perform editor STATUS (screen STATUS)
DW ESP-1 ;perform editor SPECIAL
JMP SIN ;initialize editor (initialize screen)
DB 0 ;reserved
```

FIX SCRENV

** SCRENV - Screen Handler Vector Table
--

```
DW SOP-1 ;perform screen OPEN
DW ECL-1 ;perform screen CLOSE (editor CLOSE)
DW SGB-1 ;perform screen GET-BYTE
DW SPB-1 ;perform screen PUT-BYTE
DW SST-1 ;perform screen STATUS
DW SSP-1 ;perform screen SPECIAL
JMP SIN ;initialize screen
DB 0 ;reserved
```

FIX KEYBDV

** KEYBDV - Keyboard Handler Vector Table
--

```
DW SST-1 ;perform keyboard OPEN (screen STATUS)
DW SST-1 ;perform keyboard CLOSE (screen STATUS)
DW KGB-1 ;perform keyboard GET-BYTE
DW ESP-1 ;perform keyboard SPECIAL (editor SPECIAL)
DW SST-1 ;perform keyboard STATUS (screen STATUS)
DW ESP-1 ;perform keyboard SPECIAL (editor SPECIAL)
JMP SIN ;initialize keyboard (initialize screen)
DB 0 ;reserved
```

FIX PRINTV

** PRINTV - Printer Handler Vector Table

```
DW POP-1 ;perform printer OPEN
DW PCL-1 ;perform printer CLOSE
DW PSP-1 ;perform printer SPECIAL
DW PPB-1 ;perform printer PUT-BYTE
DW PST-1 ;perform printer STATUS
DW PSP-1 ;perform printer SPECIAL
JMP PIN ;initialize printer
DB 0 ;reserved
```

FIX CASETV

**	CASETV - Cassette Handler Vector Table
----	---

DW	COP-1	;perform cassette OPEN
DW	CCL-1	;perform cassette CLOSE
DW	CGB-1	;perform cassette GET-BYTE
DW	CPB-1	;perform cassette PUT-BYTE
DW	CST-1	;perform cassette STATUS
DW	CSP-1	;perform cassette SPECIAL
JMP	CIN	;initialize cassette
DB	0	;reserved

Jump Vectors

** Jump Vectors

FIX JMP	DINITV IDIO ;initialize DIO
FIX JMP	DSKINV DIO ;perform DIO
FIX JMP	CIOV CIO ;perform CIO
FIX JMP	SIOV PIO ;perform PIO
FIX JMP	SETVBV SVP ;set VBLANK parameters
FIX JMP	SYSVBV IVNM ;process immediate VBLANK NMI
FIX JMP	XITVBV DVNM ;process deferred VBLANK NMI
FIX JMP	SIOINV ISIO ;initialize SIO
FIX JMP	SENDEV ESS ;enable SIO SEND
FIX JMP	INTINV IIH ;initialize interrupt handler
FIX JMP	CIOINV ICIO ;initialize CIO
FIX JMP	BLKBDV PPD ;perform power-up display
FIX JMP	WARMSV PWS ;perform warmstart
FIX JMP	COLDSV PCS ;perform coldstart
FIX JMP	RBLOKV RCB ;read cassette block
FIX JMP	CSOPIV OCI ;open cassette for input
FIX JMP	PUPDIV PPD ;perform power-up display
FIX JMP	SLFTSV STH ;self-test hardware
FIX JMP	PHENTV PHE ;perform peripheral handler entry
FIX JMP	PHUNLV PHU ;perform peripheral handler unlinking

FIX	PHINIV
JMP	PHI ;perform peripheral handler initialization

Generic Parallel Device Handler Vector Table

FIX GPDVV

**	GPDVV - Generic Parallel Device Handler Vector Table
----	--

DW	GOP-1	;perform generic parallel device OPEN
DW	GCL-1	;perform generic parallel device CLOSE
DW	GGB-1	;perform generic parallel device GET-BYTE
DW	GPB-1	;perform generic parallel device PUT-BYTE
DW	GST-1	;perform generic parallel device STATUS
DW	GSP-1	;perform generic parallel device SPECIAL
JMP	GIN	;initialize generic parallel device

\$E4C0 Patch

FIX \$E4C0

<pre>** E4C0 - \$E4C0 Patch * * For compatibility with OS Revision B, return.</pre>

RTS ;return

Central Input/Output

```
**      ICIO - Initialize CIO
*
*      ENTRY JSR      ICIO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
ICIO  =      *          ;entry
;      Initialize IOCB's.

      LDX      #0          ;index of first IOCB

ICIO1 LDA      #IOCFRE          ;IOCB free indicator
      STA      ICHID,X          ;set IOCB free
      LDA      #low [IIN-1]
      STA      ICPTL,X          ;initialize PUT-BYTE routine address
      LDA      #high [IIN-1]
      STA      ICPH,X
      TXA
      CLC
      ADC      #IOCBSZ          ;add IOCB size
      TAX
      CMP      #MAXIOC          ;index of next IOCB
      BCC      ICIO1            ;index of first invalid IOCB
      BCC      ICIO1            ;if not done

      RTS                      ;return
```

```
**      IIN - Indicate IOCB Not Open Error
*
*      ENTRY JSR      IIN
*
*      EXIT
*          Y = IOCB Not Open error code
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
IIN  =      *          ;entry
      LDY      #NOTOPN          ;IOCB not open error
      RTS                      ;return
```

```
**      CIO - Central Input/Output
*
*      ENTRY JSR      CIO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
CIO  =      *          ;entry
;      Initialize.

      STA      CIOCHR ;save possible output byte value
```

```

    STX    ICIDNO ;save IOCB index
;    Check IOCB index validity.

    TXA    ;IOCB index
    AND    #$0F ;index modulo 16
    BNE    CI01 ;if IOCB not multiple of 16, error

    CPX    #MAXIOC ;index of first invalid IOCB
    BCC    CI02 ;if index within range

;    Indicate Invalid IOCB Index error.

CI01    LDY    #BADIOC ;invalid IOCB index error
        JMP    SSC ;set status and complete operation, return

;    Move part of IOCB to zero page IOCB.

CI02    LDY    #0 ;offset to first byte of page zero IOCB

CI03    LDA    IOCB,X ;byte of IOCB
        STA    IOCBAS,Y ;byte of zero page IOCB
        INX
        INY
        CPY    #ICSPRZ-IOCBAS ;offset to first undesired byte
        BCC    CI03 ;if not done

;    Check for provisionally open IOCB.

        LDA    ICHIDZ ;handler ID
        CMP    #$7F ;provisionally open indicator
        BNE    PCC ;if not provisionally open, perform command

;    Check for CLOSE command.

        LDA    ICCOMZ ;command
        CMP    #CLOSE
        BEQ    XCL ;if CLOSE command

;    Check handler load flag.

        LDA    HNDLOD
        BNE    LHO ;if handler load desired

;    Indicate nonexistent device error.

;    JMP    IND ;indicate nonexistent device error, return

```

```

**      IND - Indicate Nonexistent Device Error
*
*      ENTRY JSR    IND
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

IND    =    * ;entry
        LDY    #NONDEV ;nonexistent device error

IND1    JMP    SSC ;set status and complete operation, return

```



```

**      LHO - Load Peripheral Handler for OPEN
*
*      ENTRY JSR      LHO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

LHO      =      *      ;entry
          JSR      PHL      ;load and initialize peripheral handler
          BMI      IND1      ;if error

;          JMP      PCC      ;perform CIO command, return

```

```

**      PCC - Perform CIO Command
*
*      ENTRY JSR      PCC
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PCC      =      *      ;entry

;      Check command validity.

          LDY      #NVALID      ;assume invalid code
          LDA      ICCOMZ      ;command
          CMP      #OPEN      ;first valid command
          BCC      XOP1      ;if command invalid

          TAY      ;command

          CPY      #SPECIL      ;last valid command
          BCC      PCC1      ;if valid

          LDY      #SPECIL      ;substitute SPECIAL command

;      Obtain vector offset.

PCC1     STY      ICCOMT      ;save command
          LDA      TCV0-3,Y      ;vector offset for command
          BEQ      XOP      ;if OPEN command, process

;      Perform command.

          CMP      #2
          BEQ      XCL      ;if CLOSE command, process

          CMP      #8
          BCS      XSS      ;if STATUS or SPECIAL command, process

          CMP      #4
          BEQ      XGT      ;if GET command, process

          JMP      XPT      ;process PUT command, process

```

```

**      XOP - Execute OPEN Command
*
*      ENTRY JSR    XOP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

XOP      =      *      ;entry
;      Check IOCB free.

      LDA      ICHIDZ ;handler ID
      CMP      #IOCFRE      ;IOCB free indicator
      BEQ      XOP2      ;if IOCB free

;      Process error.

      LDY      #PRVOPN      ;IOCB previously open error
XOP1     JMP      SSC      ;set status and complete operation, return

;      Check handler load.

XOP2     LDA      HNDLOD
      BNE      PP0      ;if user wants unconditional poll

;      Search handler table.

      JSR      SHT      ;search handler table
      BCS      PP0      ;if not found, poll

;      Initialize status.

      LDA      #0
      STA      DVSTAT ;clear status
      STA      DVSTAT+1

;      Initialize IOCB.

;      JMP      II0      ;initialize IOCB for OPEN, return

```

```

**      II0 - Initialize IOCB for OPEN
*
*      ENTRY JSR    II0
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

II0      =      *      ;entry

;      Compute handler entry point.

      JSR      CEP      ;compute handler entry point
      BCS      XOP1      ;if error

;      Execute command.

      JSR      EHC      ;execute handler command

;      Set PUT-BYTE routine address in IOCB.

```

```

LDA    #PUTCHR
STA    ICCOMT ;command
JSR    CEP    ;compute handler entry point
LDA    ICSPRZ ;PUT-BYTE routine address
STA    ICPTLZ ;IOCB PUT-BYTE routine address
LDA    ICSPRZ+1
STA    ICPTHZ
JMP    CCO    ;complete CIO operation, return

```

```

**      PPO - Poll Peripheral for OPEN
*
*      ENTRY JSR    PPO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

PPO    =      *      ;entry
JSR    PH0      ;poll
JMP    SSC      ;set status and complete operation, return

```

```

**      XCL - Execute CLOSE Command
*
*      ENTRY JSR    XCL
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

XCL    =      *      ;entry
;      Initialize.

LDY    #SUCCES    ;assume success
STY    ICSTAZ     ;status
JSR    CEP        ;compute handler entry point
BCS    XCL1       ;if error

;      Execute command.

JSR    EHC        ;execute handler command

;      Close IOCB.

XCL1   LDA    #IOCFRE    ;IOCB free indicator
STA    ICHIDZ     ;indicate IOCB free
LDA    #high [IIN-1]
STA    ICPTHZ     ;reset initial PUT-BYTE routine address
LDA    #low [IIN-1]
STA    ICPTLZ
JMP    CCO        ;complete CIO operation, return

```

```

**      XSS - Execute STATUS and SPECIAL Commands
*
*      ENTRY   JSR    XSS
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

XSS      =      *      ;entry

;      Check IOCB free.

      LDA      ICHIDZ ;handler ID
      CMP      #IOCFRE
      BNE      XSS1   ;if IOCB not free

;      Open IOCB.

      JSR      SHT     ;search handler table
      BCS      XOP1    ;if error

;      Execute command.

XSS1     JSR      CEP     ;compute handler entry point
      JSR      EHC      ;execute handler command

;      Restore handler ID, in case IOCB implicitly opened.

      LDX      ICIDNO ;IOCB index
      LDA      ICHID,X   ;original handler ID
      STA      ICHIDZ ;restore zero page handler ID
      JMP      CCO      ;complete CIO operation, return

```

```

**      XGT - Execute GET Command
*
*      ENTRY   JSR    XGT
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

XGT      =      *      ;entry

;      Check GET validity.

      LDA      ICCOMZ ;command
      AND      ICAX1Z
      BNE      XGT2   ;if GET command valid

;      Process error.

      LDY      #WRONLY      ;IOCB opened for write only error

XGT1     JMP      SSC      ;set status and complete operation, return

;      Compute and check handler entry point.

XGT2     JSR      CEP     ;compute handler entry point
      BCS      XGT1    ;if error

;      Check buffer length.

```

```

    LDA    ICBLLZ          ;buffer length
    ORA    ICBLLZ+1
    BNE    XGT3            ;if buffer length non-zero

;    Get byte.

    JSR    EHC             ;execute handler command
    STA    CIOCHR ;data
    JMP    CC0             ;complete CIO operation, return

;    Fill buffer.

XGT3    JSR    EHC             ;execute handler command
        STA    CIOCHR         ;data
        BMI    XGT7           ;if error, end transfer

        LDY    #0
        STA    (ICBALZ),Y     ;byte of buffer
        JSR    IBP            ;increment buffer pointer
        LDA    ICCOMZ         ;command
        AND    #$02
        BNE    XGT4           ;if GET RECORD command

;    Check for EOL.

        LDA    CIOCHR ;data
        CMP    #EOL
        BNE    XGT4           ;if not EOL

;    Process EOL.

        JSR    DBL            ;decrement buffer length
        JMP    XGT7           ;clean up

;    Check buffer full.

XGT4    JSR    DBL            ;decrement buffer length
        BNE    XGT3           ;if buffer not full, continue

;    Check command.

        LDA    ICCOMZ ;command
        AND    #$02
        BNE    XGT7           ;if GET CHARACTER command, clean up

;    Process GET RECORD.

XGT5    JSR    EHC             ;execute handler command
        STA    CIOCHR ;data
        BMI    XGT6           ;if error

;    Check for EOL.

        LDA    CIOCHR ;data
        CMP    #EOL
        BNE    XGT5           ;if not EOL, continue

;    Process end of record.

        LDA    #TRNRCD        ;truncated record error
        STA    ICSTAZ ;status

;    Process error.

XGT6    JSR    DBP            ;decrement buffer pointer
        LDY    #0
        LDA    #EOL

```

```

    STA    (ICBALZ),Y    ;set EOL in buffer
    JSR    IBP           ;increment buffer pointer

;    Clean up.

XGT7     JSR    SFL      ;set final buffer length
         JMP    CCO      ;complete CIO operation, return

```

```

**      XPT - Execute PUT Command
*
*      ENTRY JSR    XPT
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

XPT      =      *      ;entry

;    Check PUT validity.

    LDA    ICCOMZ ;command
    AND    ICAX1Z
    BNE    XPT2    ;if PUT command valid

;    Process error.

    LDY    #RDONLY    ;IOCB opened for read only error

XPT1     JMP    SSC    ;set status and complete operation, return

;    Compute and check handler entry point.

XPT2     JSR    CEP    ;compute handler entry point
         BCS    XPT1    ;if error

;    Check buffer length.

    LDA    ICBLLZ ;buffer length
    ORA    ICBLLZ+1
    BNE    XPT3    ;if buffer length non-zero

;    Put byte.

    LDA    CIOCHR ;data
    INC    ICBLLZ ;set buffer length to 1
    BNE    XPT4    ;transfer one byte

;    Transfer data from buffer to handler.

XPT3     LDY    #0
    LDA    (ICBALZ),Y    ;byte from buffer
    STA    CIOCHR        ;data

XPT4     JSR    EHC      ;execute handler command
         PHP           ;save status
         JSR    IBP      ;increment buffer pointer
         JSR    DBL      ;decrement buffer length
         PLP           ;status
         BMI    XPT6      ;if error

;    Check command.

    LDA    ICCOMZ ;command
    AND    #$02

```

```

    BNE    XPT5    ;if PUT RECORD command
;
    Check for EOL.
    LDA    CIOCHR ;data
    CMP    #EOL
    BEQ    XPT6    ;if EOL, clean up
;
    Check for buffer empty.
XPT5  LDA    ICBLLZ        ;buffer length
    ORA    ICBLLZ+1
    BNE    XPT3          ;if buffer not empty, continue
;
    Check command.
    LDA    ICCOMZ ;command
    AND    #$02
    BNE    XPT6    ;if PUT CHARACTER command
;
    Write EOL.
    LDA    #EOL
    JSR    EHC    ;execute handler command
;
    Clean up.
XPT6  JSR    SFL    ;set final buffer length
    JMP    CCO    ;complete CIO operation, return

```

```

**      SSC - Set Status and Complete Operation
*
*      ENTRY JSR    SSC
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

SSC    =      *      ;entry
    STY    ICSTAZ ;status
;    JMP    CCO    ;complete CIO operation, return

```

```

**      CCO - Complete CIO Operation
*
*      ENTRY JSR    CCO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

CCO    =      *      ;entry
;
    Initialize.
    LDY    ICIDNO        ;IOCB index
;
    Restore buffer pointer.
    LDA    ICBAL,Y
    STA    ICBALZ        ;restore buffer pointer
    LDA    ICBAH,Y
    STA    ICBAHZ

```

```

;      Move part of zero page IOCB to IOCB.

      LDX    #0          ;first byte of zero page IOCB
      STX    HNDLOD

CC01  LDA    IOCBAS,X     ;byte of zero page IOCB
      STA    IOCB,Y      ;byte of IOCB
      INX
      INY
      CPX    #ICSPRZ-IOCBAS ;offset to first undesired byte
      BCC    CC01        ;if not done

;      Restore A, X and Y.

      LDA    CIOCHR      ;data
      LDX    ICIDNO      ;IOCB index
      LDY    ICSTAZ      ;status
      RTS               ;return

```

```

**      CEP - Compute Handler Entry Point
*
*      ENTRY JSR    CEP
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

CEP    =    *          ;entry

;      Check handler ID validity.

      LDY    ICHIDZ      ;handler ID
      CPY    #MAXDEV+1   ;first invalid ID
      BCC    CEP1        ;if handler ID within range

;      Process error.

      LDY    #NOTOPN     ;IOCB not open error
      BCS    CEP2        ;return

;      Compute entry point.

CEP1   LDA    HATABS+1,Y ;low address
      STA    ICSPRZ
      LDA    HATABS+2,Y ;high address
      STA    ICSPRZ+1
      LDY    ICCOMT      ;command
      LDA    TCV0-3,Y    ;vector offset for command
      TAY
      LDA    (ICSPRZ),Y  ;low vector address
      TAX              ;low vector address
      INY
      LDA    (ICSPRZ),Y  ;high vector address
      STA    ICSPRZ+1    ;set high address
      STX    ICSPRZ      ;set low address
      CLC               ;indicate success

;      Exit.

CEP2   RTS               ;return

```



```

**      DBL - Decrement Buffer Length
*
*      ENTRY JSR      DBL
*
*      EXIT
*          Z set if buffer length = 0
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DBL      =      *          ;entry
          LDA      ICBLLZ   ;low buffer length
          BNE      DBL1     ;if low buffer length non-zero

          DEC      ICBLLZ+1 ;decrement high buffer length

DBL1     DEC      ICBLLZ     ;decrement low buffer length
          LDA      ICBLLZ
          ORA      ICBLLZ+1 ;indicate buffer length status
          RTS

```

```

**      DBP - Decrement Buffer Pointer
*
*      ENTRY JSR      DBP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DBP      =      *          ;entry
          LDA      ICBALZ   ;low buffer address
          BNE      DBP1     ;if low buffer address non-zero

          DEC      ICBALZ+1 ;decrement high buffer address

DBP1     DEC      ICBALZ     ;decrement low buffer address
          RTS

```

```

**      IBP - Increment Buffer Pointer
*
*      ENTRY JSR      IBP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

IBP      =      *          ;entry
          INC      ICBALZ   ;increment low buffer address
          BNE      IBP1     ;if low buffer address non-zero

          INC      ICBALZ+1 ;increment high buffer address

IBP1     RTS

```

```

**      SFL - Set Final Buffer Length
*
*      ENTRY JSR    SFL
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin    1983-11-01

```

```

SFL      =      *          ;entry
          LDX      ICIDNO    ;IOCB index
          SEC
          LDA      ICBLL,X    ;initial length
          SBC      ICBLLZ     ;subtract byte count
          STA      ICBLLZ     ;update length
          LDA      ICBLH,X
          SBC      ICBLLZ+1
          STA      ICBLHZ
          RTS              ;return

```

```

**      EHC - Execute Handler Command
*
*      ENTRY JSR    EHC
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin    1983-11-01

```

```

EHC      =      *          ;entry
          LDY      #FNCNOT    ;assume function not defined error
          JSR      IDH        ;invoke device handler
          STY      ICSTAZ     ;status
          CPY      #0         ;set N accordingly
          RTS              ;return

```

```

**      IDH - Invoke Device Handler
*
*      ENTRY JSR    IDH
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin    1983-11-01

```

```

IDH      =      *          ;entry
          TAX              ;save A
          LDA      ICSPRZ+1 ;high vector
          PHA              ;put high vector on stack
          LDA      ICSPRZ   ;low vector
          PHA              ;put low vector on stack
          TXA              ;restore A
          LDX      ICIDNO   ;IOCB index
          RTS              ;invoke handler (address on stack)

```

```

**      SHT - Search Handler Table
*
*      ENTRY JSR      SHT
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

SHT      =      *      ;entry

;      Set device number.

      SEC
      LDY      #1
      LDA      (ICBALZ),Y      ;device number
      SBC      #'1'
      BMI      SHT1      ;if number less than "1"

      CMP      #'9'-'1'+1
      BCC      SHT2      ;if number in range "1" to "9"
SHT1     LDA      #0      ;substitute device number "1"
SHT2     STA      ICDNOZ      ;device number (0 through 8)
      INC      ICDNOZ      ;adjust number to range 1 through 9

;      Find device handler.

      LDY      #0      ;offset to device code
      LDA      (ICBALZ),Y      ;device code
;      JNE      FDH      ;find device handler, return

```

```

**      FDH - Find Device Handler
*
*      ENTRY JSR      FDH
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

FDH      =      *      ;entry

;      Check device code.

      BEQ      FDH2      ;if device code null

;      Search handler table for device.

      LDY      #MAXDEV      ;offset to last possible entry
FDH1     CMP      HATABS,Y      ;device code from table
      BEQ      FDH3      ;if device found

      DEY
      DEY
      DEY
      BPL      FDH1      ;if not done

;      Process device not found.

FDH2     LDY      #NONDEV      ;nonexistent device error
      SEC      ;indicate error

```

```

    RTS                ;return

;    Set handler ID.

FDH3  TYA              ;offset to device code in table
      STA    ICHIDZ    ;set handler ID
      CLC              ;indicate no error
      RTS              ;return

```

<pre> ** TCV0 - Table of Command Vector Offsets * * Entry n is the vector offset for command n+3. </pre>
--

```

TCV0  DB    0          ;3 - open
      DB    4          ;4
      DB    4          ;5 - get record
      DB    4          ;6
      DB    4          ;7 - get byte(s)
      DB    6          ;8
      DB    6          ;9 - put record
      DB    6          ;10
      DB    6          ;11 - put byte(s)
      DB    2          ;12 - close
      DB    8          ;13 - status
      DB    10         ;14 - special

```

Peripheral Handler Loading Facility, Part 3

```
**      PHR - Perform Peripheral Handler Loading Initialization
*
*      * Performs Power-up Polling, with Handler loading at MEMLO
*      and Initialization;
*      * Performs System Reset Re-initialization of all handlers.
*
*      Input Parameters:
*      WARMST (used to distinguish Cold and Warm Start).
*
*      Output Parameters:
*      None.
*
*      Modified:
*      Registers are not saved;
*      All kinds of side effects when any handler is loaded
*      (potentially MEMLO, DVSTAT thru DVSTAT+3, the DCB,
*      CHLINK, ZCHAIN, TEMP1, TEMP2, TEMP3.  This list may
*      not be complete.).
*
*      ENTRY JSR    PHR
*
*      MODS
*          R. S. Scheiman      1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin        1983-11-01
```

```
PHR      =      *      ;entry

;      Check for coldstart.

      LDA      WARMST      ;warmstart flag
      BEQ      PHR2        ;if coldstart

;      Process warmstart.

      LDA      #low [CHLINK-18]
      STA      ZCHAIN
      LDA      #high [CHLINK-18]
      STA      ZCHAIN+1

;      Check next link.

PHR1     LDY      #18      ;offset to link
      CLC
      LDA      (ZCHAIN),Y  ;low link
      TAX
      INY
      ADC      (ZCHAIN),Y  ;high link
      BEQ      PHR4        ;if forward link null

;      Re-initialize peripheral handler.

      LDA      (ZCHAIN),Y  ;high link
      STA      ZCHAIN+1
      STX      ZCHAIN
      JSR      CLT          ;checksum linkage table
      BNE      PHR4        ;if checksum bad

      JSR      PHW          ;re-initialize peripheral handler
      BCS      PHR4        ;if error

;      Continue with next handler.
```

```

        BCC    PHR1            ;continue with next handler
;      Process coldstart.
PHR2   LDA    #0
        STA    CHLINK          ;clear chain link
        STA    CHLINK+1
        LDA    #$4F            ;send POLL RESET poll
        BNE    PHR7
;      Perform type 3 poll.
PHR3   LDA    #0
        TAY
        JSR    PHP
        BPL    PHR5            ;if poll answered
;      Exit.
PHR4   RTS                    ;return
;      Process answered poll.
PHR5   CLC
        LDA    MEMLO
        ADC    DVSTAT
        STA    TEMP1
        LDA    MEMLO+1
        ADC    DVSTAT+1
        STA    TEMP1+1        ;(TEMP2 := MEMLO + handler size)
        SEC
        LDA    MEMTOP
        SBC    TEMP1
        LDA    MEMTOP+1
        SBC    TEMP1+1        ;(subtract MEMTOP)
        BCS    PHR8            ;if room to load
;      Prepare for another poll.
PHR6   LDA    #$4E            ;following any load or init fail,
                                ;prepare for another Type 3 Poll by
                                ;sending a "special" load command over
                                ;serial port.
;      Poll.
PHR7   TAY                    ;Send either "special" load command or
        JSR    PHP
        JMP    PHR3            ;go poll again
;      Load peripheral handler.
PHR8   LDA    DVSTAT+2        ;call the loader
        LDX    MEMLO
        STX    DVSTAT+2        ;(Parameter = load address)
        LDX    MEMLO+1
        STX    DVSTAT+3
        JSR    LPH            ;load peripheral handler
        BMI    PHR6            ;if load error, poll again
        SEC                    ;Call for initialize new handler:
        JSR    PHC            ;(Parameter = add size to MEMLO)
        BCS    PHR6            ;if init error, poll again
        BCC    PHR3            ;poll again normally

```

```

**      PHP - Perform Poll
*
*      Polling subroutine calls SIO for Type 3 or 4 Poll.
*
*      Input Parameters:
*      A      Value for AUX1
*      Y      Value for AUX2
*
*      Output Parameters:
*      Y      SIO status from poll
*      DVSTAT: Device minimum size (low), if poll answered
*      DVSTAT+1: Device minimum size (high), if poll answered
*      DVSTAT+2: Device address for loading, if poll answered
*      DVSTAT+3: Device version number, if poll answered
*
*      Modified:
*      The registers are not saved.
*
*      Subroutines called:
*      SIO (performs poll and returns to PHP's caller).
*
*      ENTRY JSR    PHP
*
*      MODS
*
*      R. S. Scheiman      1982-04-01
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

PHP      =      *      ;entry

;      Initialize.

      PHA      ;save parameter

;      Set up DCB.

      LDX      #PHPAL-1      ;offset to last byte of DCB data
PHP1     LDA      PHPA,X      ;byte of DCB data
      STA      DCB,X      ;byte of DCB
      DEX
      BPL      PHP1      ;if not done

;      Set parameters in DCB auxiliary bytes.

      STY      DAUX2
      PLA
      STA      DAUX1

;      Perform SIO.

      JMP      SIOV      ;vector to SIO, return

;      DCB Poll Request Data

PHPA     DB      $4F      ;device bus ID
      DB      1      ;unit number
      DB      '@'      ;type 3 or 4 poll command
      DB      $40      ;I/O direction
      DW      DVSTAT ;buffer
      DB      30      ;timeout
      DB      0
      DW      4      ;buffer length

PHPAL    =      *-PHPA ;length

```

```

**      LPH - Load Peripheral Handler
*
*      This subroutine calls the relocating loader to load
*      a handler from a peripheral.
*
*      Input parameters:
*      A      Peripheral serial address for load;
*      DVSTAT+2: Load address (low)
*      DVSTAT+3: Load address (high)
*
*      Output parameters:
*      From the relocating loader.
*
*      Modified:
*      TEMP1, TEMP2, TEMP3,
*      DVSTAT+3, DVSTAT+3 (forced even),
*      Relocating loader variables and parameters,
*      Registers not saved.
*
*      Subroutines called:
*      RLR (relocating loader).
*
*      ENTRY JSR    LPH
*
*      MODS
*          R. S. Scheiman      1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin        1983-11-01

```

```

LPH      =      *      ;entry

;      Initialize.

      STA      TEMP2 ;save peripheral address
      LDX      #0
      STX      TEMP1 ;set starting block number
      DEX
      STX      TEMP3 ;set starting byte number

;      Ensure load address even.

      LDA      DVSTAT+2      ;low load address
      ROR      A
      BCC      LPH1          ;if even

      INC      DVSTAT+2      ;increment low load address
      BNE      LPH1          ;if no carry

      INC      DVSTAT+3      ;increment high load address

;      Set up relocating loader parameters.

LPH1     LDA      DVSTAT+2      ;load address
      STA      LOADAD
      LDA      DVSTAT+3
      STA      LOADAD+1
      LDA      #low PHG      ;get-byte routine address
      STA      GBYTEA
      LDA      #high PHG
      STA      GBYTEA+1
      LDA      #$80          ;loader page zero load address
      STA      ZLOADA

;      Relocate routine.

```


JMP RLR ;relocate routine, return

```
**      PHG - Perform Peripheral Handler GET-BYTE
*
*      Get a byte subroutine for relocating loader passes
*      bytes from peripheral to relocating loader via
*      cassette buffer. Calls GNL each time new
*      buffer is needed.
*
*      Input parameters:
*      TEMP1: Next block number;
*      TEMP2: Peripheral address (for GNL);
*      TEMP3: Next byte number (index to CASBUF).
*
*      Output parameters (for relocating loader):
*      Carry bit on indicates error;
*      A      Next byte, if no error.
*
*      Modified:
*      Cassette buffer CASBUF;
*      TEMP3;
*      X, Y not saved.
*
*      Subroutines called:
*      GNL, which calls SIO to get load records.
*
*      ENTRY JSR     PHG
*
*      MODS
*
*      R. S. Scheiman     1982-04-01
*      1. Bring closer to Coding Standard (object unchanged).
*
*      R. K. Nordin     1983-11-01
```

```
PHG     =     *                    ;entry
;
;     Check for another byte in buffer.
;
;     LDX     TEMP3
;     INX
;     STX     TEMP3
;     BEQ     PHG2                ;if empty, load next block
;
;     Retrieve next byte.
PHG1    LDX     TEMP3
;     LDA     CASBUF-$80,X ;byte
;     CLC                    ;indicate no error
;     RTS                    ;return
;
;     Load next block and retrieve next byte.
PHG2    LDA     #-128 ;offset to first byte
;     STA     TEMP3
;     JSR     GNL     ;get next load block
;     BPL     PHG1     ;if no error, retrieve next byte
;
;     Process error.
;
;     SEC                    ;indicate error
;     RTS                    ;return
```

```

**      GNL - Get Next Load Block
*
*      Subroutine to get a load block from the peripheral.
*
*      Input parameters:
*      TEMP1: Block number.
*
*      Output parameters:
*      Negative bit is set by SIO if I/O error occurs.
*
*      Modified:
*      TEMP1;
*      the DCB (SIO);
*      Registers not saved.
*
*      Subroutines called:
*      SIO.
*
*      ENTRY JSR    GNL
*
*      MODS
*          R. S. Scheiman      1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin        1983-11-01

```

```

GNL      =      *      ;entry

;      Set up DCB.

      LDX      #GNLAL-1      ;offset to last DCB data byte

GNL1     LDA      GNLA,X      ;byte of DCB data
      STA      DCB,X      ;byte of DCB
      DEX
      BPL      GNL1      ;if not done

;      Set DCB parameters.

      LDX      TEMP1      ;block number
      STX      DAUX1      ;auxiliary 1
      INX
      STX      TEMP1      ;next block number
      LDA      TEMP2      ;device address
      STA      DDEVIC      ;device bus ID

;      Perform SIO.

      JMP      SIOV      ;vector to SIO, return

;      DCB Data

GNLA     DB      $00      ;dummy device bus ID
      DB      1      ;dummy unit number
      DB      '&'      ;load command
      DB      $40      ;I/O direction
      DW      CASBUF      ;buffer
      DB      30      ;timeout
      DB      0
      DW      128      ;buffer length
      DB      0      ;auxiliary 1
      DB      0      ;auxiliary 2

GNLAL    =      *-GNLA ;length

```

```

**      SHC - Search Handler Chain
*
*      Forward chain search searches for pointer to handler linkage
*      table whose address matches caller's parameter. If caller's
*      parameter is zero, this routine looks for the pointer to
*      the final linkage table since this table's forward pointer
*      is zero (null).
*
*      Input parameters:
*      A      Linkage table address to match (High)
*      Y      Linkage table address to match (Low)
*
*      Output parameters:
*      ZCHAIN points to linkage table whose forward pointer
*      contains the match (if match is found);
*      if the match is found just following the linkage
*      chain base CHLINK, then ZCHAIN points to CHLINK
*      minus 18;
*      If match successful, A (High) and X (Low) contain
*      matched address (equiv. to A and Y parms.);
*      Carry bit is set to indicate no match or checksum violation
*      along the chain. [Note: the linkage table pointed
*      to by ZCHAIN upon return is not checksum checked.]
*
*      Modified:
*      TEMP1, TEMP2, ZCHAIN;
*      The registers are not saved.
*
*      Subroutines called:
*      CLT.
*
*      ENTRY JSR      SHC
*
*      MODS
*
*      R. S. Scheiman      1982-04-01
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin        1983-11-01

```

```

SHC      =      *      ;entry

;      Initialize.

      STY      TEMP1
      STA      TEMP1+1
      LDA      #low [CHLINK-18]
      STA      ZCHAIN      ;start ZCHAIN at proper offset
      LDA      #high [CHLINK-18]
      STA      ZCHAIN+1

;      Check for match.

SHC1     LDY      #18
      LDA      (ZCHAIN),Y
      TAX
      INY
      LDA      (ZCHAIN),Y      ;high chain pointer
      CMP      TEMP2      ;check for match with parameter
      BNE      SHC2      ;if no match

      CPX      TEMP1
      BNE      SHC2      ;if no match

;      Exit.

      CLC      ;indicate match

```

```

    RTS                ;return
;    Check for end of chain.
SHC2  CMP    #0        ;end of chain indicator
      BNE    SHC4      ;if not end of chain

      CPX    #0
      BNE    SHC4      ;if not end of chain
;    Process end of chain or checksum error.
SHC3  SEC                ;return error (checksum or end)
      RTS                ;return
;    Set link to new linkage table.
SHC4  STX    ZCHAIN     ;link to new linkage table
      STA    ZCHAIN+1

      JSR    CLT        ;checksum linkage table
      BNE    SHC3      ;if error
;    Continue searching chain.
      BEQ    SHC1      ;continue searching chain

```

```

**    PHW - Perform Peripheral Handler Warmstart Initialization
*
*    PHC is the main entry. This performs full initialization,
*    including adding the new linkage table into the linkage
*    table chain;
*    PHW does all initialization except adding to the linkage
*    table chain (intended for warm start reinitialization);
*    PHI is the full initialization entry for calling
*    init from outside the OS.
*
*    The code does the following:
*    1)    Links new handler to end of chain;
*    2)    Calls handler init subroutine in handler;
*    3)    If 2 failed, unlinks handler from chain,
*           and returns with carry;
*    4)    Else, conditionally zeroes handler size entry of
*           handler linkage table (per parameter);
*    5)    Adds handler size entry (possibly zeroed) to MEMLO;
*    6)    If handler size entry is nonzero, MEMLO is then
*           forced even;
*    7)    Calculates and enters linkage table checksum;
*    8)    Returns with carry clear.
*
*    PHC is called by PHR when loading handlers at cold
*    initialization; and by PHL when loading a handler under
*    application request under CIO;
*    PHW is called by PHR to reinitialize a handler during
*    warm-start;
*    PHI is vectored by OS vector at $E49E and is intended
*    for use by system-level applications which load
*    handlers (ie., AUTORUN.SYS handler loader, etc.).
*
*    Input parameters:
*    PHC:
*        DVSTAT, DVSTAT+1 contain handler size (for
*        handler init, not used by this routine);
*        DVSTAT+2, DVSTAT+3 contain handler linkage table
*        address.
*    PHW:

```

```

*          DVSTAT+2, DVSTAT+3 same;
*          DVSTAT, DVSTAT+1 undefined.
*
*  PHI:
*
*          A and Y contain handler linkage table address;
*          they are copied into DVSTAT+3 and DVSTAT+2;
*          DVSTAT, DVSTAT+1 may or may not be significant--
*          any concern about these are up to the programmer
*          of the peripheral handler init routine and whoever
*          is making use of the non-OS-caller entry PHI.
*
*          For PHI and PHC, the Carry bit specifies whether
*          the handler size entry of the linkage table is to
*          be zeroed prior to adding to MEMLO: Carry set means
*          do NOT zero this entry.
*
*          Output parameters:
*          Carry indicates error (initialization failed);
*          The registers are not saved.
*
*          Modified:
*          DVSTAT+2, DVSTAT+3 are modified by PHI;
*          ZCHAIN, TEMP1, TEMP2;
*          MEMLO, MEMLO+1 conditionally incremented by handler's size.
*
*          Subroutines called:
*          SHC (to find end of linkage table chain);
*          PHU (to unlink handler if init. error);
*          CLT (to insert linkage table checksum);
*          loaded handler's INIT entry.
*
*          ENTRY JSR    PHW
*
*          MODS
*
*          R. S. Scheiman    1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PHW    =    *    ;entry
SEC    ;indicate not zeroing handler size
PHP
BCS    PHQ    ;initialize handler and update MEMLO

```

```

**      PHI - Perform Peripheral Handler Initialization with Chaining
*
*      ENTRY JSR    PHI
*
*      MODS
*
*          R. S. Scheiman    1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PHI    =    *    ;entry
STA    DVSTAT+3
STY    DVSTAT+2
;      PHC    ;perform coldstart initialization

```

```

**      PHC - Perform Peripheral Handler Coldstart Initialization
*
*      ENTRY JSR    PHC
*
*      MODS
*          R. S. Scheiman      1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin        1983-11-01

```

```

PHC      =      *      ;entry
;      Initialize.

      PHP

;      Search for end of chain.

      LDA      #0      ;indicate searching for end of chain
      TAY
      JSR      SHC      ;search handler chain
      BCS      PHQ1     ;if error, exit

;      Enter at end of chain.

      LDY      #18      ;offset
      LDA      DVSTAT+2
      STA      (ZCHAIN),Y ;low link
      TAX
      INY
      LDA      DVSTAT+3
      STA      (ZCHAIN),Y ;high link
      STX      ZCHAIN    ;link to new table
      STA      ZCHAIN+1
      LDA      #0      ;indicate end of chain
      STA      (ZCHAIN),Y ;low link
      DEY
      STA      (ZCHAIN),Y ;high link

;      Initialize handler.

;      JMP      PHQ      ;initialize handler, return

```

```

**      PHQ - Initialize Handler and Update MEMLO
*
*      ENTRY JSR    PHQ
*
*      MODS
*          R. S. Scheiman      1982-04-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin        1983-11-01

```

```

PHQ      =      *      ;entry
;      Initialize handler.

      JSR      PHX      ;initialize handler
      BCC      PHQ2     ;if no error

;      Process error.

      LDA      DVSTAT+3
      LDY      DVSTAT+2
      JSR      PHU      ;unlink handler

```

```

;      Exit, indicating error.

PHQ1  PLP                ;fix stack
      SEC                ;indicate error
      RTS                ;return

;      Check for zeroing handler size.

PHQ2  PLP
      BCS      PHQ3      ;if not to zero

;      Zero handler size.

      LDA      #0
      LDY      #16      ;offset
      STA      (ZCHAIN),Y ;zero size
      INY
      STA      (ZCHAIN),Y

;      Increase MEMLO by size.

PHQ3  CLC
      LDY      #16      ;offset to size
      LDA      MEMLO
      ADC      (ZCHAIN),Y ;add low size
      STA      MEMLO      ;new low MEMLO
      INY
      LDA      MEMLO+1
      ADC      (ZCHAIN),Y ;add high size
      STA      MEMLO+1    ;new high MEMLO

;      Put checksum in linkage table.

      LDY      #15      ;offset to checksum
      LDA      #0
      STA      (ZCHAIN),Y ;clear checksum
      JSR      CLT      ;checksum linkage table
      LDY      #15      ;offset to checksum
      STA      (ZCHAIN),Y ;checksum

;      Exit.

      CLC                ;indicate success
      RTS                ;return

```

```

**      PHX - Initialize Handler
*
*      ENTRY JSR      PHX
*
*      MODS
*      R. S. Scheiman      1982-04-01
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

PHX    =      *      ;entry
      CLC
      LDA      ZCHAIN
      ADC      #12
      STA      TEMP1      ;low handler initialization address
      LDA      ZCHAIN+1
      ADC      #0
      STA      TEMP1+1    ;high handler initialization address
      JMP      (TEMP1)    ;initialize handler, return

```

\$E912 Patch

FIX \$E912

****** **E912 - \$E912 Patch**

***** **For compatibility with OS Revision B, set VBLANK parameters.**

JMP **SVP** ;set VBLANK parameters, return

Peripheral Handler Loading Facility, Part 4

```
**      PHU - Perform Peripheral Handler Unlinking
*
*      Handler entry unlinking routine. This routine is called either
*      by the OS handler initialization to unlink a handler when its
*      initialization fails, or by the handler itself if it implements
*      the handler unload feature.      This routine is entered via the
*      OS vector at $E49B.
*
*      Input parameters:
*      A      Address of linkage table to unlink (High);
*      Y      Address of linkage table to unlink (Low).
*      COLDST: Tested to see if PHU is called during cold starting;
*               if so, chain entry is unlinked even if at MEMLO.
*
*      Output parameters:
*      Carry is set to indicate error; in this case,
*               no unlinking has occurred.
*
*      Modified:
*      TEMP1, TEMP2;
*      ZCHAIN,ZCHAIN+1;
*      The forward chain pointer in the predecessor of the linkage
*      table being removed is modified to point to the successor
*      of the removed table if the removal is successful--
*      this forward chain pointer may be CHLINK,CHLINK+1.
*
*      The registers are not saved.
*
*      Subroutines called:
*      SHC, CLT.
*
*      ENTRY JSR      PHU
*
*      MODS
*
*      R. S. Scheiman      1982-04-01
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01
```

```
PHU      =      *      ;entry
;      Search handler chain.
JSR      SHC      ;search handler chain
BCS      PHU3     ;if error
;      Perform unlinking.
TAY      ;(save return parameter)
LDA      ZCHAIN   ;save ZCHAIN (points to predecessor)
PHA
LDA      ZCHAIN+1
PHA
STX      ZCHAIN   ;make ZCHAIN point to linkage table
STY      ZCHAIN+1 ;to be removed
LDA      COLDST   ;coldstart flag
BNE      PHU1     ;if coldstart, unconditional unlink
LDY      #16      ;check if loaded at MEMLO...
CLC      ;by checking if size is nonzero
LDA      (ZCHAIN),Y
INY
ADC      (ZCHAIN),Y
BNE      PHU2     ;if handler size non-zero
```

```

    JSR    CLT          ;checksum linkage table
    BNE    PHU2         ;if checksum nonzero, bad checksum

PHU1  LDY    #18        ;take link from table being removed...
      LDA    (ZCHAIN),Y
      TAX
      INY
      LDA    (ZCHAIN),Y
      TAY
      PLA          ;Make ZCHAIN point to the predecessor...
      STA    ZCHAIN+1
      PLA
      STA    ZCHAIN
      TYA          ;And put forward link from table being
      LDY    #19        ;removed into its predecessors link.
      STA    (ZCHAIN),Y
      DEY
      TXA
      STA    (ZCHAIN),Y
      CLC          ;indicate success
      RTS          ;return

;      Clean stack and process error.

PHU2  PLA          ;Error return--restore stack
      PLA

;      Process error.

PHU3  SEC          ;indicate error
      RTS          ;return

```

\$E959 Patch

FIX \$E959

**	E959 - \$E959 Patch
*	
*	For compatibility with OS Revision B, perform PIO.

JMP PIO ;perform PIO, return

Serial Input/Output

```
**      ISIO - Initialize SIO
*
*      ENTRY JSR      ISIO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
ISIO    =      *      ;entry

        LDA      #MOTRST
        STA      PACTL ;turn off motor

        LDA      #NCOMHI
        STA      PBCTL ;raise NOT COMMAND line

        LDA      #$03 ;POKEY out of initialize mode
        STA      SSKCTL ;SKCTL shadow
        STA      SOUNDNR ;select noisy I/O
        STA      SKCTL

        RTS      ;return
```

```
**      SIO - Serial Input/Output
*
*      ENTRY JSR      SIO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
SIO      =      *      ;entry

;      Initialize.

        TSX
        STX      STACKP ;save stack pointer
        LDA      #1      ;critical section indicator
        STA      CRITIC ;indicate critical section

;      Check device ID.

        LDA      DDEVIC ;device ID
        CMP      #CASET
        BNE      SI01    ;if not cassette

;      Process cassette.

        JMP      PCI      ;process cassette I/O, return

;      Process intelligent device.

SI01     LDA      #0
        STA      CASFLG ;indicate not cassette

        LDA      #DRETRI
        STA      DRETRY ;set device retry count

SI02     LDA      #CRETRI
```

```

        STA    CRETRY ;set command frame retry count
;        Send command frame.
SI03  LDA    #low B19200
        STA    AUDF3      ;set baud rate to 19200
        LDA    #high B19200
        STA    AUDF4
;        Set up command buffer.
        CLC
        LDA    DDEVIC      ;device ID
        ADC    DUNIT      ;add unit number
        ADC    #$FF        ;subtract 1
        STA    CDEVIC      ;device bus ID
        LDA    DCOMND      ;command
        STA    CCOMND
        LDA    DAUX1        ;auxiliary information 1
        STA    CAUX1
        LDA    DAUX2        ;auxiliary information 2
        STA    CAUX2
;        Set buffer pointer to command frame buffer.
        CLC
        LDA    #low CDEVIC ;low buffer address
        STA    BUFRLO      ;low buffer address
        ADC    #4
        STA    BFENLO      ;low buffer end address
        LDA    #high CDEVIC ;high buffer address
        STA    BUFRHI      ;high buffer address
        STA    BFENHI      ;high buffer end address
;        Send command frame to device.
        LDA    #NCOMLO
        STA    PBCTL        ;lower NOT COMMAND line
        JSR    SID          ;send command frame
        LDA    ERRFLG        ;error flag
        BNE    SI04          ;if error received
        TYA
        BNE    SI05          ;if ACK received
;        Process NAK or timeout.
SI04  DEC    CRETRY ;decrement command frame retry count
        BPL    SI03      ;if retries not exhausted
;        Process command frame retries exhausted.
        JMP    SI010 ;process error
;        Process ACK.
SI05  LDA    DSTATS
        BPL    SI06      ;if no data to send
;        Send data frame to device.
        LDA    #CRETRI
        STA    CRETRY ;set command frame retry count
        JSR    SBP      ;set buffer pointers
        JSR    SID      ;send data frame
        BEQ    SI010 ;if error

```

```

;      Wait for complete.

SI06  JSR    GTO      ;set device timeout
      LDA    #0
      STA    ERRFLG ;clear error flag
      JSR    STW      ;set timer and wait
      BEQ    SI08      ;if timeout

;      Process no timeout.

      BIT    DSTATS
      BVS    SI07      ;if more data follows

      LDA    ERRFLG ;error flag
      BNE    SI010     ;if error

;      Process no error.

      BEQ    CS0      ;complete SIO operation

;      Receive data frame from device.

SI07  JSR    SBP      ;set buffer pointers
      JSR    REC      ;receive

;      Check error flag.

SI08  LDA    ERRFLG ;error flag
      BEQ    SI09      ;if no error preceded data

;      Process error.

      LDA    TSTAT ;temporary status
      STA    STATUS ;status

;      Check status.

SI09  LDA    STATUS ;status
      CMP    #SUCCES
      BEQ    CS0      ;if successful, complete operation, return

;      Process error.

SI010 DEC    DRETRY ;decrement device retry count
      BMI    CS0      ;if retries exhausted, complete, return

;      Retry.

      JMP    SI02      ;retry

```

```

**      CS0 - Complete SIO Operation
*
*      ENTRY JSR    CS0
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

CS0  =      *      ;entry
      JSR    DSR      ;disable SEND and RECEIVE
      LDA    #0        ;not critical section indicator
      STA    CRITIC ;critical section flag
      LDY    STATUS ;status
      STY    DSTATS ;status

```

RTS ;return

```
**      WCA - Wait for Completion or ACK
*
*      ENTRY JSR    WCA
*
*      EXIT
*          Y = 0, if failure
*          = $FF, if success
*
*      NOTES
*          Problem: WCA does not handle NAK correctly; the BNE WCA4
*          just before WCA3 should be removed.
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
WCA    =      *      ;entry

;      Initialize.

      LDA    #0
      STA    ERRFLG      ;clear error flag

;      Set buffer pointer.

      CLC
      LDA    #low TEMP    ;low temporary address
      STA    BUFRLO      ;low buffer address
      ADC    #1
      STA    BFENLO      ;low buffer end address
      LDA    #high TEMP   ;high temporary address
      STA    BUFRHI      ;high buffer address
      STA    BFENHI      ;high buffer end address
      LDA    #$FF
      STA    NOCKSM      ;indicate no checksum follows
      JSR    REC          ;receive
      LDY    #$FF        ;assume success
      LDA    STATUS      ;status
      CMP    #SUCCES
      BNE    WCA2          ;if failure

      LDA    TEMP        ;byte received
      CMP    #ACK
      BEQ    WCA4          ;if ACK, exit

      CMP    #COMPLT
      BEQ    WCA4          ;if complete, exit

      CMP    #ERROR
      BNE    WCA1          ;if device did not send back

;      Process unrecognized response.

      LDA    #DERROR
      STA    STATUS      ;indicate device error
      BNE    WCA2          ;check for timeout

;      Process nothing sent back.

WCA1   LDA    #DNACK
      STA    STATUS      ;indicate NAK
```

```

;      Check for timeout.

WCA2   LDA     STATUS      ;status
      CMP     #TIMOUT
      BEQ     WCA3        ;if timeout

;      Process other error.

      LDA     #$FF        ;error indicator
      STA     ERRFLG      ;indicate error
      BNE     WCA4        ;exit

;      Indicate failure.

WCA3   LDY     #0          ;failure indicator

;      Exit.

WCA4   LDA     STATUS      ;status
      STA     TSTAT       ;temporary status
      RTS              ;return

```

```

**      SEN - Send
*
*      SEN sends a buffer over the serial bus.
*
*      ENTRY JSR     SEN
*
*      NOTES
*          Problem: an interrupt may occur before CHKSUM is
*          initialized, causing an incorrect checksum calculation;
*          STA CHKSUM should precede STA SEROUT.
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SEN     =     *           ;entry

;      Initialize.

      LDA     #SUCCES     ;assume success
      STA     STATUS      ;status
      JSR     ESS         ;enable SIO SEND
      LDY     #0
      STY     CHKSUM      ;clear checksum
      STY     CHKSNT      ;clear checksum sent flag
      STY     XMTDON      ;clear transmit-frame done flag

;      Initiate TRANSMIT.

      LDA     (BUFRLO),Y  ;first byte from buffer
      STA     SEROUT      ;serial output register
      STA     CHKSUM      ;checksum

;      Check BREAK key.

SEN1    LDA     BRKKEY
      BNE     SEN2        ;if BREAK key not pressed

;      Process BREAK key.

      JMP     PBK         ;process BREAK key, return

```



```

;      Process BREAK key not pressed.

SEN2  LDA    XMTDON      ;transmit-frame done flag
      BEQ    SEN1        ;if transmit-frame not done

;      Exit.

      JSR    DSR          ;disable SEND and RECEIVE
      RTS                ;return

```

```

**      ORIR - Process Serial Output Ready IRQ
*
*      ENTRY JMP    ORIR
*
*      EXIT
*          Exits via RTI
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

ORIR  =      *          ;entry

;      Initialize.

      TYA
      PHA          ;save Y
      INC    BUFRLO ;increment low buffer pointer
      BNE    ORI1  ;if low buffer pointer non-zero

      INC    BUFRHI ;increment high buffer pointer

;      Check end of buffer.

ORI1  LDA    BUFRLO ;buffer address
      CMP    BFENLO ;buffer end address
      LDA    BUFRHI
      SBC    BFENHI
      BCC    ORI4   ;if not past end of buffer

;      Process end of buffer.

      LDA    CHKSNT ;checksum sent flag
      BNE    ORI2   ;if checksum already sent

;      Send checksum.

      LDA    CHKSUM ;checksum
      STA    SEROUT ;serial output register
      LDA    #$FF
      STA    CHKSNT ;indicate checksum sent
      BNE    ORI3

;      Enable TRANSMIT done interrupt.

ORI2  LDA    POKMSK
      ORA    #$08
      STA    POKMSK
      STA    IRQEN

;      Exit.

ORI3  PLA
      TAY          ;restore Y

```

```

        PLA          ;restore A
        RTI          ;return

;      Transmit next byte from buffer.

ORI4   LDY    #0
        LDA    (BUFRLO),Y    ;byte from buffer
        STA    SEROUT        ;serial output register
        CLC
        ADC    CHKSUM        ;add byte to checksum
        ADC    #0
        STA    CHKSUM        ;update checksum
        JMP    ORI3          ;exit

```

```

**      OCIR - Process Serial Output Complete IRQ
*
*      ENTRY  JMP    OCIR
*
*      EXIT
*      Exits via RTI
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

OCIR   =    *      ;entry

;      Check checksum sent.

        LDA    CHKSNT ;checksum sent flag
        BEQ    OCI1   ;if checksum not yet sent

;      Process checksum sent.

        STA    XMTDON ;indicate transmit-frame done

;      Disable TRANSMIT done interrupt.

        LDA    POKMSK
        AND    #$F7
        STA    POKMSK
        STA    IRQEN

;      Exit.

OCI1   PLA          ;restore A
        RTI          ;return

```

```

**      REC - Receive
*
*      ENTRY  JSR    REC
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

REC    =    *      ;entry

;      Initialize.

        LDA    #0
        LDY    CASFLG

```

```

    BNE    REC1    ;if cassette

    STA    CHKSUM ;initialize checksum

REC1  STA    BUFRFL ;clear buffer full flag
      STA    RECVDN ;clear receive-frame done flag
      LDA    #SUCCES      ;assume success
      STA    STATUS ;status
      JSR    ESR      ;enable SIO RECEIVE
      LDA    #NCOMHI
      STA    PBCTL

;      Check BREAK key.

REC2  LDA    BRKKEY
      BNE    REC3    ;if BREAK key not pressed

;      Process BREAK key.

      JMP    PBK      ;process BREAK key, return

;      Process BREAK key not pressed.

REC3  LDA    TIMFLG ;timeout flag
      BEQ    ITO      ;if timeout, indicate timeout

;      Process no timeout.

      LDA    RECVDN ;receive-frame done flag
      BEQ    REC2      ;if receive-frame done, continue

;      Exit.

      RTS              ;return

```

```

**      ITO - Indicate Timeout
*
*      ENTRY JSR    ITO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

ITO    =      *      ;entry
      LDA    #TIMOUT      ;timeout indicator
      STA    STATUS ;indicate timeout
      RTS              ;return

```

```

**      IRIR - Process Serial Input Ready IRQ
*
*      ENTRY JMP    IRIR
*
*      EXIT
*          Exits via RTI
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

IRIR    =      *      ;entry

;      Initialize.

```

```

TYA
PHA          ;save Y
LDA  SKSTAT
STA  SKRES  ;reset status register

; Check for frame error.
BMI  IRI1   ;if no frame error

; Process frame error.
LDY  #FRMERR ;frame error
STY  STATUS ;indicate frame error

; Check for overrun error.
IRI1 AND  #$20
     BNE  IRI2 ;if no overrun error

; Process overrun error.
LDY  #OVRRUN ;overrun error
STY  STATUS ;indicate overrun error

; Check for buffer full.
IRI2 LDA  BUFRFL
     BEQ  IRI5 ;if buffer not yet full

; Process buffer full.
LDA  SERIN ;checksum from device
CMP  CHKSUM ;computed checksum
BEQ  IRI3  ;if checksums match

; Process checksum error.
LDY  #CHKERR ;checksum error
STY  STATUS ;indicate checksum error

; Indicate receive-frame done.
IRI3 LDA  #$FF ;receive-frame done indicator
     STA  RECDN ;indicate receive-frame done

; Exit.
IRI4 PLA
     TAY      ;restore Y
     PLA      ;restore A
     RTI      ;return

; Process buffer not full.
IRI5 LDA  SERIN ;serial input register
     LDY  #0
     STA  (BUFRLO),Y ;byte of buffer
     CLC
     ADC  CHKSUM ;add byte to checksum
     ADC  #0
     STA  CHKSUM ;update checksum
     INC  BUFRLO ;increment low buffer pointer
     BNE  IRI6  ;if low buffer pointer non-zero

     INC  BUFRHI ;increment high buffer pointer

```

```

;      Check end of buffer.

IRI6   LDA    BUFRLO      ;buffer address
      CMP    BFENLO      ;buffer end address
      LDA    BUFRHI
      SBC    BFENHI
      BCC    IRI4         ;if not past end of buffer

;      Process end of buffer.

      LDA    NOCKSM       ;no checksum follows flag
      BEQ    IRI7         ;if checksum will follow

;      Process no checksum will follow.

      LDA    #0
      STA    NOCKSM       ;clear no checksum follows flag
      BEQ    IRI3         ;indicate receive-frame done

;      Process checksum will follow.

IRI7   LDA    #$FF
      STA    BUFRFL       ;indicate buffer full
      BNE    IRI4         ;exit

```

```

**      SBP - Set Buffer Pointers
*
*      ENTRY JSR    SBP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SBP    =      *          ;entry
      CLC
      LDA    DBUFLO
      STA    BUFRLO ;low buffer address
      ADC    DBYTLO
      STA    BFENLO ;low buffer end address
      LDA    DBUFHI
      STA    BUFRHI ;high buffer address
      ADC    DBYTHI
      STA    BFENHI ;high buffer end address
      RTS          ;return

```

```

**      PCI - Process Cassette I/O
*
*      ENTRY JSR    PCI
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PCI    =      *          ;entry

;      Check command type.

      LDA    DSTATS       ;command type
      BPL    PCI3         ;if READ

;      Write a record.

```

```

LDA    #low B00600
STA    AUDF3        ;set 600 baud
LDA    #high B00600
STA    AUDF4
JSR    ESS          ;enable SIO SEND
LDX    PALNTS       ;PAL/NTSC offset
LDY    WSIRGX,X     ;low short WRITE IRG time
LDA    DAUX2        ;IRG type
BMI    PCI1         ;if short IRG is desired

LDY    WIRGLX,X     ;low long WRITE IRG time

PCI1   LDX    #WIRGHI ;high IRG time
      JSR    SSV      ;set SIO VBLANK parameters
      LDA    #MOTRGO
      STA    PACTL    ;turn on motor

PCI2   LDA    TIMFLG  ;timeout flag
      BNE    PCI2     ;if no timeout

      JSR    SBP      ;set buffer pointers
      JSR    SEN      ;send
      JMP    PCI6     ;exit

;      Read a record.

PCI3   LDA    #$FF    ;cassette I/O indicator
      STA    CASFLG   ;cassette I/O flag

      LDX    PALNTS   ;PAL/NTSC offset
      LDY    RSIRGX,X ;low short READ IRG time
      LDA    DAUX2    ;IRG type
      BMI    PCI4     ;if short IRG desired

      LDY    RIRGLX,X ;low long READ IRG time

PCI4   LDX    #RIRGHI ;high READ IRG time
      JSR    SSV      ;set SIO VBLANK parameters
      LDA    #MOTRGO
      STA    PACTL    ;turn on motor

PCI5   LDA    TIMFLG  ;timeout flag
      BNE    PCI5     ;if no timeout

      JSR    SBP      ;set buffer pointers
      JSR    GT0      ;get device timeout
      JSR    SSV      ;set SIO VBLANK parameters
      JSR    SBR      ;set initial baud rate
      JSR    REC      ;receive

;      Exit.

PCI6   LDA    DAUX2    ;IRG type
      BMI    PCI7     ;if doing short IRG

      LDA    #MOTRST
      STA    PACTL    ;turn off motor

PCI7   JMP    CS0     ;complete SIO operation, return

```

```

**      PTE - Process Timer Expiration
*
*      ENTRY JSR      PTE
*
*      MODS
*
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PTE      =      *      ;entry
          LDA      #0      ;timeout indicator
          STA      TIMFLG ;timeout flag
          RTS

```

```

**      ESS - Enable SIO SEND
*
*      ENTRY JSR      ESS
*
*      MODS
*
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

ESS      =      *      ;entry

;      Initialize.

          LDA      #$07      ;mask off previous serial bus control bits
          AND      SSKCTL
          ORA      #$20      ;set SEND mode

;      Check device type.

          LDY      DDEVIC
          CPY      #CASET
          BNE      ESS1      ;if not cassette

;      Process cassette.

          ORA      #$08      ;set FSK output
          LDY      #LOTONE      ;set FSK tone frequencies
          STY      AUDF2
          LDY      #HITONE
          STY      AUDF1

;      Set serial bus control.

ESS1     STA      SSKCTL ;SKCTL shadow
          STA      SKCTL
          LDA      #$C7      ;mask off previous serial bus interrupt bits
          AND      POKMSK ;and with POKEY IRQ enable
          ORA      #$10      ;enable output data needed interrupt
          JMP      SSR      ;set for SEND, return

```

```

**      ESR - Enable SIO RECEIVE
*
*      ENTRY JSR      ESR
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

ESR      =      *      ;entry
          LDA      #$07 ;mask off previous serial bus control bits
          AND      SSKCTL
          ORA      #$10 ;set receive mode asynchronous
          STA      SSKCTL ;SKCTL shadow
          STA      SKCTL
          STA      SKRES
          LDA      #$C7 ;mask off previous serial bus interrupt bits
          AND      POKMSK ;and with POKEY IRQ enable
          ORA      #$20 ;enable RECEIVE interrupt
;      BNE      SSR      ;set for RECEIVE, return

```

```

**      SSR - Set for SEND or RECEIVE
*
*      ENTRY JSR      SSR
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SSR      =      *      ;entry
;      Initialize.

          STA      POKMSK ;update POKEY IRQ enable
          STA      IRQEN ;IRQ enable
          LDA      #$28 ;clock ch. 3 with 1.79 MHz, ch. 4 with ch. 3
          STA      AUDCTL ;set audio control

;      Set voice controls.

          LDX      #6      ;offset to last voice control
          LDA      #$A8 ;pure tone, half volume
          LDY      SOUNDNR ;noisy I/O flag
          BNE      SSR1 ;if noisy I/O desired

          LDA      #$A0 ;pure tone, no volume
SSR1      STA      AUDC1,X      ;set tone and volume
          DEX
          DEX
          BPL      SSR1 ;if not done

;      Turn off certain voices.

          LDA      #$A0 ;pure tone, no volume
          STA      AUDC3 ;turn off sound on voice 3
          LDY      DDEVIC ;device bus ID
          CPY      #CASET ;cassette device ID
          BEQ      SSR2 ;if cassette device

          STA      AUDC1 ;turn off sound on voice 1
          STA      AUDC2 ;turn off sound on voice 2

```


SSR2 **RTS** ;return

```
**      DSR - Disable SEND and RECEIVE
*
*      ENTRY JSR    DSR
*
*      NOTES
*          Problem: NOP may not be necessary.
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
DSR      =      *      ;entry

;      Disable serial bus interrupts.

      NOP
      LDA    #$C7    ;mask to clear serial bus interrupts
      AND    POKMSK ;and with POKEY IRQ enable
      STA    POKMSK ;update POKEY IRQ enable
      STA    IRQEN   ;IRQ enable

;      Turn off audio volume.

      LDX    #6      ;offset to last voice control
      LDA    #$00    ;no volume

DSR1     STA    AUDC1,X      ;turn off voice
      DEX
      DEX
      BPL     DSR1      ;if not done
      RTS           ;return
```

```
**      GT0 - Get Device Timeout
*
*      ENTRY JSR    GT0
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
GT0      =      *      ;entry
      LDA    DTIMLO ;device timeout
      ROR    A
      ROR    A
      TAY           ;rotated timeout
      AND    #$3F   ;lower 6 bits
      TAX           ;high timeout
      TYA           ;rotated timeout
      ROR    A
      AND    #$C0   ;upper 2 bits
      TAY           ;low timeout
      RTS           ;return
```

```
**      TSIH - Table of SIO Interrupt Handlers
```

```
*      NOTES
```

```
*      Problem: not used.
```

```
TSIH  DW    IRIR    ;serial input ready IRQ
      DW    ORIR    ;serial output ready IRQ
      DW    OCIR    ;serial output complete IRQ
```

```
**      SID - Send to Intelligent Device
```

```
*      ENTRY JSR    SID
```

```
*      NOTES
```

```
*      Problem: bytes wasted by outer delay loop.
```

```
*      MODS
```

```
*      Original Author Unknown
```

```
*      1. Bring closer to Coding Standard (object unchanged).
```

```
*      R. K. Nordin      1983-11-01
```

```
SID    =      *      ;entry
;      Delay.
      LDX    #1
SID1   LDY    #255
SID2   DEY
      BNE    SID2      ;if inner loop not done
      DEX
      BNE    SID1      ;if outer loop not done
;      Send data frame.
      JSR    SEN      ;send
;      Set timer and wait.
      LDY    #low CTIM ;frame acknowledge timeout
      LDX    #high CTIM
;      JNE    STW      ;set timer and wait, return
```

```
**      STW - Set Timer and Wait
```

```
*      ENTRY JSR    STW
```

```
*      MODS
```

```
*      Original Author Unknown
```

```
*      1. Bring closer to Coding Standard (object unchanged).
```

```
*      R. K. Nordin      1983-11-01
```

```
STW    =      *      ;entry
      JSR    SSV      ;set SIO VBLANK parameters
      JSR    WCA      ;wait for completion or ACK
      TYA          ;wait termination status
      RTS          ;return
```

```

**      CBR - Compute Baud Rate
*
*      CBR computes value for POKEY frequency for the baud rate as
*      measured by an interval of the VCOUNT timer.
*
*      ENTRY JSR      CBR
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

CBR      =      *      ;entry
          STA     TIMER2      ;save final timer value
          STY     TIMER2+1
          JSR     AVV          ;adjust VCOUNT value
          STA     TIMER2      ;save adjusted timer 2 value
          LDA     TIMER1
          JSR     AVV          ;adjust VCOUNT value
          STA     TIMER1      ;save adjusted timer 1 value
          LDA     TIMER2
          SEC
          SBC     TIMER1
          STA     TEMP1        ;save difference
          LDA     TIMER2+1
          SEC
          SBC     TIMER1+1
          TAY
          LDX     PALNTS
          LDA     #0
          SEC
          SBC     CONS1X,X

CBR1      CLC
          ADC     CONS1X,X      ;accumulate product
          DEY
          BPL     CBR1          ;if not done

          CLC
          ADC     TEMP1        ;add to get total VCOUNT difference
          TAY                ;total VCOUNT difference
          LSR     A
          LSR     A
          LSR     A
          ASL     A            ;interval divided by 4
          SEC
          SBC     #22          ;adjust offset
          TAX                ;offset
          TYA                ;total VCOUNT difference
          AND     #7           ;extract lower 3 bits of interval
          TAY                ;lower 3 bits of interval
          LDA     #-11

CBR2      CLC
          ADC     #11          ;accumulate interpolation constant
          DEY
          BPL     CBR2          ;if done

          LDY     #0           ;assume no addition correction
          SEC
          SBC     #7           ;adjust interpolation constant
          BPL     CBR3

          DEY                ;indicate addition correction

```

```

CBR3  CLC
      ADC    TPFV,X      ;add constant to table value
      STA    CBAUDL      ;low POKEY frequency value
      TYA
      ADC    TPFV+1,X
      STA    CBAUDH      ;high POKEY frequency value
      RTS               ;return

```

```

**    AVV - Adjust VCOUNT Value
*
*    ENTRY JSR    AVV
*
*    MODS
*
*        Original Author Unknown
*        1. Bring closer to Coding Standard (object unchanged).
*        R. K. Nordin      1983-11-01

```

```

AVV    =    *           ;entry
      CMP    #$7C
      BMI    AVV1      ;if >= $7C

      SEC
      SBC    #$7C
      RTS               ;return

```

```

AVV1   CLC
      LDX    PALNTS
      ADC    CONS2X,X
      RTS               ;return

```

```

**    SBR - Set Initial Baud Rate
*
*    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.
*
*    ENTRY JSR    SBR
*
*    NOTES
*
*        Problem: bytes wasted by branch around branch (SBR3).
*
*    MODS
*
*        Original Author Unknown
*        1. Bring closer to Coding Standard (object unchanged).
*        R. K. Nordin      1983-11-01

```

```

SBR    =    *           ;entry

SBR1   LDA    BRKKEY
      BNE    SBR2      ;if BREAK key not pressed
      JMP    PBK        ;process BREAK key, return

SBR2   SEI
      LDA    TIMFLG      ;timeout flag
      BNE    SBR3      ;if no timeout
      BEQ    SBR5      ;process timeout

SBR3   LDA    SKSTAT
      AND    #$10        ;extract start bit
      BNE    SBR1      ;if start bit

```

```

STA SAVIO          ;save serial data in
LDX VCOUNT        ;vertical line counter
LDY RTCLOCK+2      ;low byte of VBLANK clock
STX TIMER1
STY TIMER1+1       ;save initial timer value
LDX #1
STX TEMP3          ;set mode flag
LDY #10            ;10 bits

SBR4 LDA BRKKEY
    BEQ PBK        ;if BREAK key pressed, process, return

    LDA TIMFLG     ;timeout flag
    BNE SBR6       ;if no timeout

SBR5 CLI
    JMP ITO        ;indicate timeout, return

SBR6 LDA SKSTAT
    AND #$10       ;extract
    CMP SAVIO      ;previous serial data in
    BEQ SBR4       ;if data in not changed

    STA SAVIO      ;save serial data in
    DEY            ;decrement bit counter
    BNE SBR4       ;if not done

    DEC TEMP3      ;decrement mode
    BMI SBR7       ;if done with both modes

    LDA VCOUNT
    LDY RTCLOCK+2
    JSR CBR        ;compute baud rate
    LDY #9         ;9 bits
    BNE SBR4       ;set bit counter

SBR7 LDA CBAUDL
    STA AUDF3
    LDA CBAUDH
    STA AUDF4      ;set POKEY baud rate
    LDA #0
    STA SKSTAT
    LDA SSKCTL
    STA SKSTAT     ;initialize POKEY serial port
    LDA #$55
    STA (BUFRLO),Y ;first byte of buffer
    INY
    STA (BUFRLO),Y ;second byte of buffer
    LDA #$AA
    STA CHKSUM     ;checksum
    CLC
    LDA BUFRLO
    ADC #2         ;add 2
    STA BUFRLO     ;update low buffer pointer
    LDA BUFRHI
    ADC #0
    STA BUFRHI     ;update high buffer pointer
    CLI
    RTS            ;return

```

```

**      PBK - Process BREAK Key
*
*      ENTRY JSR    PBK
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

PBK      =      *      ;entry
          JSR      DSR      ;disable SEND and RECEIVE
          LDA      #MOTRST
          STA      PACTL ;turn off motor
          LDA      #NCOMHI
          STA      PBCTL ;raise NOT COMMAND line
          LDA      #BRKABT      ;BREAK abort error
          STA      STATUS ;status
          LDX      STACKP ;saved stack pointer
          TXS      ;restore stack pointer
          DEC      BRKKEY ;indicate BREAK
          CLI
          JMP      CS0      ;complete SIO operation, return to caller of SIO

```

```

**      SSV - Set SIO VBLANK Parameters
*
*      ENTRY JSR    SSV
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

SSV      =      *      ;entry
          LDA      #low PTE      ;timer expiration routine address
          STA      CDTMA1
          LDA      #high PTE
          STA      CDTMA1+1
          LDA      #1      ;timer 1
          SEI
          JSR      SETVBV      ;set VBLANK parameters
          LDA      #1      ;no timeout indicator
          STA      TIMFLG      ;timeout flag
          CLI
          RTS      ;return

```

```

**      TPFV - Table of POKEY Frequency Values
*
*      TPFV translates VCOUNT interval timer measurements to POKEY
*      frequency register values.
*
*      Table entries are AUDF+7.
*
*      Frequency-out is Frequency-in divided by 2*(AUDF+M), where
*      Frequency-in = 1.78979 MHz and M = 7.
*
*       $AUDF+7 = (11.365167) * T\text{-out}$ , where T-out is the number of counts
*      (127 used cd solution) of VCOUNT for one character
*      time (10 bit times).

```

```

;      DW      636      ;baud rate 1407, VCOUNT interval 56
;      DW      727      ;baud rate 1231, VCOUNT interval 64
;      DW      818      ;baud rate 1094, VCOUNT interval 72
;      DW      909      ;baud rate 985, VCOUNT interval 80

TPFV   DW      1000     ;baud rate 895, VCOUNT interval 88
        DW      1091     ;baud rate 820, VCOUNT interval 96
        DW      1182     ;baud rate 757, VCOUNT interval 104
        DW      1273     ;baud rate 703, VCOUNT interval 112
        DW      1364     ;baud rate 656, VCOUNT interval 120
        DW      1455     ;baud rate 615, VCOUNT interval 128
        DW      1546     ;baud rate 579, VCOUNT interval 136
        DW      1637     ;baud rate 547, VCOUNT interval 144
        DW      1728     ;baud rate 518, VCOUNT interval 152
        DW      1818     ;baud rate 492, VCOUNT interval 160
        DW      1909     ;baud rate 469, VCOUNT interval 168
        DW      2000     ;baud rate 447, VCOUNT interval 176

;      DW      2091     ;baud rate 428, VCOUNT interval 184
;      DW      2182     ;baud rate 410, VCOUNT interval 192
;      DW      2273     ;baud rate 394, VCOUNT interval 200
;      DW      2364     ;baud rate 379, VCOUNT interval 208
;      DW      2455     ;baud rate 365, VCOUNT interval 216
;      DW      2546     ;baud rate 352, VCOUNT interval 224
;      DW      2637     ;baud rate 339, VCOUNT interval 232
;      DW      2728     ;baud rate 328, VCOUNT interval 240
;      DW      2819     ;baud rate 318, VCOUNT interval 248

```

** NTSC/PAL Constant Tables

```

WIRGLX DB      low WIRGLN      ;NTSC low long write IRG
        DB      low WIRGLP     ;PAL low long write IRG

RIRGLX DB      low RIRGLN      ;NTSC low long read IRG
        DB      low RIRGLP     ;PAL low long read IRG

WSIRGX DB      low WSIRGN      ;NTSC low short write IRG
        DB      low WSIRGP     ;PAL low short write IRG

RSIRGX DB      low RSIRGN      ;NTSC low short read IRG
        DB      low RSIRGP     ;PAL low short read IRG

CONS1X DB      131             ;NTSC
        DB      156             ;PAL

CONS2X DB      7               ;NTSC
        DB      32              ;PAL

```

Keyboard, Editor and Screen Handler, Part 1

```
**      TSMA - Table of Screen Memory Allocation
*
*      Entry n is the number of $40-byte blocks to allocate for
*      graphics mode n.
*
*      NOTES
*      Problem: For readability, this, and other tables in
*      this area, could be moved closer to the other parts of
*      the Keyboard, Editor and Screen Handler (just before
*      the EF6B patch).
```

```
TSMA  DB    24    ;0
      DB    16    ;1
      DB    10    ;2
      DB    10    ;3
      DB    16    ;4
      DB    28    ;5
      DB    52    ;6
      DB   100    ;7
      DB   196    ;8
      DB   196    ;9
      DB   196   ;10
      DB   196   ;11
      DB    28   ;12
      DB    16   ;13
      DB   100   ;14
      DB   196   ;15
```

```
**      TDLE - Table of Display List Entry Counts
*
*      Each entry is 2 bytes.
```

```
TDLE  DB   23,23 ;0
      DB   11,23 ;1
      DB   47,47 ;2
      DB   95,95 ;3
      DB   97,97 ;4
      DB   97,97 ;5
      DB   23,11 ;6
      DB  191,97 ;7
      DB   19,19 ;8
      DB    9,19 ;9
      DB   39,39 ;10
      DB   79,79 ;11
      DB   65,65 ;12
      DB   65,65 ;13
      DB   19,9  ;14
      DB  159,65 ;15
```

```
**      TAGM - Table of ANTIC Graphics Modes
*
*      Entry n is the ANTIC graphics mode corresponding to internal
*      graphics mode n.
```

```
TAGM  DB   $02    ;internal 0 - 40x2x8 characters
      DB   $06    ;internal 1 - 20x5x8 characters
      DB   $07    ;internal 2 - 20x5x16 characters
      DB   $08    ;internal 3 - 40x4x8 graphics
      DB   $09    ;internal 4 - 80x2x4 graphics
      DB   $0A    ;internal 5 - 80x4x4 graphics
```


DB	\$0B	;internal 6 - 160x2x2 graphics
DB	\$0D	;internal 7 - 160x4x2 graphics
DB	\$0F	;internal 8 - 320x2x1 graphics
DB	\$0F	;internal 9 - 320x2x1 GTIA "lum" mode
DB	\$0F	;internal 10 - 320x2x1 GTIA "color/lum" mode
DB	\$0F	;internal 11 - 320x2x1 GTIA "color" mode
DB	\$04	;internal 12 - 40x5x8 characters
DB	\$05	;internal 13 - 40x5x16 characters
DB	\$0C	;internal 14 - 160x2x1 graphics
DB	\$0E	;internal 15 - 160x4x1 graphics

**	TDLV - Table of Display List Vulnerability
*	
*	Entry n is non-zero if the display list for mode n cannot
*	cross a page boundary.

TDLV	DB	0	;0
	DB	0	;1
	DB	0	;2
	DB	0	;3
	DB	0	;4
	DB	0	;5
	DB	0	;6
	DB	1	;7
	DB	1	;8
	DB	1	;9
	DB	1	;10
	DB	1	;11
	DB	0	;12
	DB	0	;13
	DB	1	;14
	DB	1	;15

**	TLSC - Table of Left Shift Counts
*	
*	Entry n is the NUMBER OF LEFT SHIFTS NEEDED TO MULTIPLY
*	COLCRS BY # BYTES/ROW ((ROWCRS*5)/(2**TLSC)) for mode n.

TLSC	DB	3	;0
	DB	2	;1
	DB	2	;2
	DB	1	;3
	DB	1	;4
	DB	2	;5
	DB	2	;6
	DB	3	;7
	DB	3	;8
	DB	3	;9
	DB	3	;10
	DB	3	;11
	DB	3	;12
	DB	3	;13
	DB	2	;14
	DB	3	;15

**	TMCC - Table of Mode Column Counts
*	
*	Entry n is the low column count for mode n.

TMCC	DB	low 40	;0
	DB	low 20	;1
	DB	low 20	;2
	DB	low 40	;3

DB	low 80	;4
DB	low 80	;5
DB	low 160	;6
DB	low 160	;7
DB	low 320	;8
DB	low 80	;9
DB	low 80	;10
DB	low 80	;11
DB	low 40	;12
DB	low 40	;13
DB	low 160	;14
DB	low 160	;15

```

**      TMRC - Table of Mode Row Counts
*
*      Entry n is the row count for mode n.

```

TMRC	DB	24	;0
	DB	24	;1
	DB	12	;2
	DB	24	;3
	DB	48	;4
	DB	48	;5
	DB	96	;6
	DB	96	;7
	DB	192	;8
	DB	192	;9
	DB	192	;10
	DB	192	;11
	DB	24	;12
	DB	12	;13
	DB	192	;14
	DB	192	;15

```

**      TRSC - Table of Right Shift Counts
*
*      Entry n is HOW MANY RIGHT SHIFTS FOR HCRSR FOR PARTIAL
*      BYTE MODES for mode n.

```

TRSC	DB	0	;0
	DB	0	;1
	DB	0	;2
	DB	2	;3
	DB	3	;4
	DB	2	;5
	DB	3	;6
	DB	2	;7
	DB	3	;8
	DB	1	;9
	DB	1	;10
	DB	1	;11
	DB	0	;12
	DB	0	;13
	DB	3	;14
	DB	2	;15

**	TDSM - Table of Display Masks
*	
*	NOTES
*	Includes TBTM - Table of Bit Masks.

TDSM	DB	\$FF	;1
	DB	\$F0	;2
	DB	\$0F	;3
	DB	\$C0	;4
	DB	\$30	;5
	DB	\$0C	;6
	DB	\$03	;7
TBTM	DB	\$80	;8 (0)
	DB	\$40	;9 (1)
	DB	\$20	;10 (2)
	DB	\$10	;11 (3)
	DB	\$08	;12 (4)
	DB	\$04	;13 (5)
	DB	\$02	;14 (6)
	DB	\$01	;15 (7)

Peripheral Handler Loading Facility, Part 5

```
**      PHE - Perform Peripheral Handler Entry
*
*      PHE attempts to enter a peripheral handler in the handler table.
*
*      ENTRY JSR    PHE
*              X = device code
*              A = high linkage table address
*              Y = low linkage table address
*
*      EXIT
*              Success:
*              C clear
*              Handler table entry made
*
*              Failure due to entry previously made:
*              C set
*              N clear
*              X = offset to second byte of duplicate entry
*              A, Y unchanged
*
*              Failure due to handler table full:
*              C set
*              N set
*
*      CHANGES
*              A X Y
*
*      CALLS
*              -none-
*
*      MODS
*              R. S. Scheiman      1982-04-01
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin        1983-11-01
```

```
PHE      =      *      ;entry
;      Initialize.

      PHA          ;save high linkage table address
      TYA
      PHA          ;save low linkage table address
;
;      Search for device code in handler table.

      TXA          ;device code
      LDX    #0      ;offset to first entry of table
PHE1     CMP    HATABS,X      ;device code from table
      BEQ    PHE3      ;if device code found

      INX
      INX
      INX
      CPX    #MAXDEV+1      ;offset+1 of last possible entry
      BMI    PHE1      ;if not done
;
;      Search for empty entry in handler table.

      LDX    #0      ;offset to first entry of table
      TAY          ;save device code
      LDA    #0
```

```

PHE2  CMP    HATABS,X      ;device code from table
      BEQ    PHE4          ;if empty entry found

      INX
      INX
      INX
      CPX    #MAXDEV+1     ;offset+1 of last possible entry
      BMI    PHE2          ;if not done

;      Return table full condition.

      PLA          ;clean stack
      PLA
      LDY    #$FF         ;indicate table full (set N)
      SEC          ;indicate failure
      RTS          ;return

;      Return device code found condition.

PHE3  PLA          ;saved Y
      TAY          ;restore Y
      PLA          ;restore A
      INX          ;indicate device code found (clear N)
      SEC          ;indicate failure
      RTS          ;return

;      Enter handler in table.

PHE4  TYA          ;device code
      STA    HATABS,X      ;enter device code
      PLA          ;saved low linkage table address
      STA    HATABS+1,X    ;low address
      PLA          ;saved high linkage table address
      STA    HATABS+2,X    ;high address

;      Return success condition.

      CLC          ;indicate success
      RTS          ;return

```

```

**      PH0 - Perform Peripheral Handler Poll at OPEN
*
*      Subroutine to perform Type 4 Poll at OPEN time, and
*      "provisionally" open IOCB if peripheral answers.
*
*      Input parameters:
*      ICIDNO identifies calling IOCB;
*      From zero-page IOCB:
*          ICBALZ,ICBAHZ (buffer pointer)
*          ICDNOZ (device number from caller's filespec.);
*      From caller's buffer: device name (in filespec.).
*
*      Output parameters:
*      "No device" error returned if Poll not answered.
*      If poll is answered, the calling IOCB is "Provisionally"
*          opened (and successful status is returned)--
*          ICHIDZ set to mark provisional open
*          ICPTLZ,ICPTHZ points to PTL (special PUT-BYTE)
*          ICSPR in calling IOCB set to device name (from filespec)
*          ICSPR+1 in calling IOCB set to device serial load addr.
*
*      Modified:
*      Registers not saved.
*
*      Subroutines called:
*      PHP performs poll.

```

```

*
* ENTRY JSR PH0
*
* NOTES
* Problem: in the CRASS65 version, ICIDNO was not declared
* zero-page.
*
* MODS
* R. S. Scheiman 1982-04-01
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01

```

```

PH0 = * ;entry
LDY #0 ;Call for Type 4 Poll with
LDA (ICBALZ),Y ;device name from user
LDY ICDNOZ ;OPEN
JSR PHP
BPL PH01 ;if poll answered

LDY #NONDEV ;Return "no device" error
RTS ;return

PH01 LDA #$7F ;"Provisionally" OPEN the IOCB...
STA ICHIDZ ;(Mark "provisional")
LDA #low [PTL-1]
STA ICPTLZ ;(Special put byte routine addr.)
LDA #high [PTL-1]
STA ICPTHZ
LDA DVSTAT+2 ;(Peripheral address for load)
; LDX ICIDNO
VFD 8\AE,8\low ICIDNO,8\high ICIDNO
STA ICSPR+1,X
LDY #0
LDA (ICBALZ),Y ;(Device name from user)
STA ICSPR,X
LDY #SUCCESS ;indicate success
RTS ;return

```

```

** PTL - Perform PUT-BYTE for Provisionally Open IOCB
*
* Put byte entry for provisionally opened IOCB's.
* This routine performs load, relocation, initialization,
* and finishes OPEN, then calls handler's put byte entry.
*
* Input parameters:
* A Byte to output;
* X IOCB index (IOCB number times 16);
* Y "Function not supported" error code $92.
* AUX1 and AUX2 in zero-page IOCB are copied from the calling
* IOCB prior to the call to PTL.
*
* Output parameters:
* Various errors may be returned if loading fails (either user
* did not allow loading by setting HNDLOD flag, or there
* was a loading error or calling error);
* If no loading error, this routine returns nothing--anything
* returned is returned by the loaded PUT-BYTE entry which
* is called by this routine after the handler is loaded,
* initialized, and opened.
*
* Modified:
* ICIDNO (a CIO variable);
* all of the zero-page IOCB is copied from the calling IOCB;
* normal CIO open-operation variables are affected;
* after opening, the zero-page IOCB is copied to the calling IOCB;

```

```

*   Registers not saved if error return; if handler is loaded
*       and opened properly, the caller's A and X registers are
*       passed to the loaded handler's PUT-BYTE routine (and
*       Y is passed to that routine as $92)--then register state
*       on return is up to handler PUT-BYTE since it returns
*       directly to caller.
*
*   Subroutines called:
*   PHL (does loading, initializing and opening--calls CIO);
*   loaded handler's INIT, OPEN, and PUT-BYTE entries all called.
*   The PUT-BYTE entry returns directly to the PTL caller.
*
*   ENTRY JSR    PTL
*
*   NOTES
*       Problem: in the CRASS65 version, ICIDNO was not declared
*       zero-page.
*
*   MODS
*       R. S. Scheiman      1982-04-01
*       1. Bring closer to Coding Standard (object unchanged).
*       R. K. Nordin        1983-11-01

```

```

PTL   =      *      ;entry
      PHA          ;save byte to output
      TXA          ;IOCB index
      PHA          ;save IOCB index
      AND    #$0F   ;IOCB index modulo 16
      BNE     PTL2   ;if IOCB not divisible by 16, error

      CPX    #MAXIOC
      BPL     PTL2   ;if IOCB index invalid

      LDA     HNDL0D
      BNE     PTL3   ;if user wants loading

      LDY     #NONDEV      ;indicate nonexistent device error

;      Return error.

PTL1  PLA          ;clean stack
      PLA
      CPY     #0       ;indicate failure (set N)
      RTS          ;return

PTL2  LDY     #BADIOC   ;indicate bad IOCB number error
      BMI     PTL1     ;return error

;      Simulate beginning of CIO, since CIO bypassed.

PTL3  ;
      STX     ICIDNO ;IOCB index
      VFD     8,$8E,8\low ICIDNO,8\high ICIDNO
      LDY     #0       ;offset to first byte of page zero IOCB

;      Copy IOCB to page zero IOCB.

PTL4  LDA     IOCB,X ;byte of IOCB
      STA     ZIOCB,Y ;byte of page zero IOCB
      INX
      INY
      CPY     #12
      BMI     PTL4     ;if not done

      JSR     PHL      ;load and initialize peripheral handler
      BMI     PTL1     ;if error

```

```

PLA          ;Re-do the put byte call,
TAX          ;this time calling real handler...
PLA
TAY
LDA    ICPTHZ
PHA          ;(Put byte entry address minus one)
LDA    ICPTLZ
PHA
TYA
LDY    #FNCNOT
RTS        ;invoke handler (address on stack)

```

\$EF6B Patch

FIX \$EF6B

<pre>** EF6B - \$EF6B Patch * * For compatibility with OS Revision B, initiate cassette READ.</pre>

JMP ICR ;initiate cassette READ, return

Keyboard, Editor and Screen Handler, Part 2

```
**      SIN - Initialize Screen
*
*      ENTRY JSR      SIN
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
SIN      =      *          ;entry

          LDA      #$FF          ;clear code indicator
          STA      CH            ;key code

          LDA      RAMSIZ        ;size of RAM
          STA      RAMTOP        ;RAM size

          LDA      #$40          ;CAPS lock indicator
          STA      SHFLOK        ;shift/control lock flags

          LDA      #low TCKD      ;table of character key definitions
          STA      KEYDEF         ;key definition table address
          LDA      #high TCKD
          STA      KEYDEF+1

          LDA      #low TFKD      ;table of function key definitions
          STA      FKDEF          ;function key definition table address
          LDA      #high TFKD
          STA      FKDEF+1

          RTS                    ;return
```

```
**      SOP - Perform Screen OPEN
*
*      ENTRY JSR      SOP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
SOP      =      *          ;entry

;      Check mode.

          LDA      ICAX2Z
          AND      #$0F
          BNE      COC          ;if not mode 0, complete OPEN command, return

;      Process mode 0.

;      JMP      EOP            ;perform editor OPEN, return
```

```

**      EOP - Perform Editor OPEN
*
*      ENTRY JSR      EOP
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

EOP      =      *      ;entry
          LDA      ICAX1Z
          AND      #$0F
          STA      ICAX1Z
          LDA      #0
;      JMP      COC      ;complete OPEN command, return

```

```

**      COC - Complete OPEN Command
*
*      ENTRY JSR      COC
*      A = mode
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

COC      =      *      ;entry
;      Check mode.

          STA      DINDEX ;save mode
          CMP      #16
          BCC      COC1    ;if mode within range

;      Process invalid mode.

          LDA      #BADMOD
          JMP      COC17

;      Initialize for OPEN.

COC1     LDA      #high DCSORG ;high domestic character set origin
          STA      CHBAS      ;character set base
          LDA      #high ICSORG ;high international character set origin
          STA      CHSALT     ;alternate character set base
          LDA      #2
          STA      CHACT
          STA      SDMCTL      ;turn off DMA
          LDA      #SUCCES
          STA      DSTAT      ;clear status
          LDA      #$C0      ;enable IRQ
          ORA      POKMSK
          STA      POKMSK
          STA      IRQEN

;      Set DLI status.

          LDA      #$40      ;disable DLI
          STA      NMIEN
          BIT      FINE
          BPL      COC2      ;if not fine scrolling (VBLANK only)

          LDA      #low FDL
          STA      VDSLST     ;DLI vector

```

```

        LDA    #high FDL
        STA    VDSLST+1
        LDA    #C0
COC2   STA    NMIEEN
;       Clear control.

        LDA    #0
        STA    TINDEX        ;clear text index (must always be 0)
        STA    ADRESS
        STA    SWPFLG
        STA    CRSINH

;       Set initial tab stops.

        LDY    #14            ;offset to last byte of bit map
        LDA    #C01          ;tab stop every 8 characters
COC3   STA    TABMAP,Y        ;set tab stop
        DEY
        BPL    COC3          ;if not done

;       Load initialize color register shadows.

        LDX    #4            ;offset to last color register shadow
COC4   LDA    TDSC,X          ;default screen color
        STA    COLOR0,X      ;set color register shadow
        DEX
        BPL    COC4          ;if not done

;       Set up.

        LDY    RAMTOP        ;(high) RAM size
        DEY                ;decrement (high) RAM size
        STY    TXTMSC+1
        LDA    #low [$0000-160] ;low RAM size - 160
        STA    TXTMSC
        LDX    DINDEX        ;mode
        LDA    TAGM,X        ;convert to ANTIC code
        STA    HOLD1         ;ANTIC code
        LDA    RAMTOP        ;(high) RAM size
        STA    ADRESS+1

;       Allocate memory.

        LDY    TSMA,X        ;number of 40-byte blocks to allocate
COC5   LDA    #40            ;40 bytes
        JSR    DBS           ;perform double byte subtract
        DEY
        BNE    COC5          ;if not done

;       Clear GTIA modes.

        LDA    GPRIOR
        AND    #$3F          ;clear GTIA modes
        STA    OPNTMP+1
        TAY

;       Determine mode.

        CPX    #8
        BCC    COC7          ;if mode < 8

        CPX    #15

```

```

    BEQ    COC6            ;if mode 15

    CPX    #12
    BCS    COC7            ;if mode >= 12

;    Process modes 9, 10 and 11.

    TXA                    ;mode
    ROR    A
    ROR    A
    ROR    A
    AND    #$C0            ;extract 2 low bits (in 2 high bits)
    ORA    OPNTMP+1
    TAY

;    Establish line boundary at X000.

COC6    LDA    #16        ;subtract 16 for page boundary
        JSR    DBS        ;perform double byte subtract

;    Check for mode 11.

    CPX    #11
    BNE    COC7            ;if mode 11

;    Set GTIA luminance.

    LDA    #6            ;GTIA luminance value
    STA    COLOR4        ;background color

;    Set new priority.

COC7    STY    GPRIOR     ;new priority

;    Set memory scan counter.

    LDA    ADRESS        ;memory scan counter
    STA    SAVMSC        ;save memory scan counter
    LDA    ADRESS+1
    STA    SAVMSC+1

;    Wait for VBLANK.

COC8    LDA    VCOUNT
    CMP    #$7A
    BNE    COC8            ;if VBLANK has not occurred

;    Put display list under RAM.

    JSR    DSD            ;perform double byte single decrement
    LDA    TDLV,X        ;display list vulnerability
    BEQ    COC9            ;if not vulnerable

    LDA    #$FF
    STA    ADRESS
    DEC    ADRESS+1        ;drop down 1 page

COC9    JSR    DDD            ;perform double byte double decrement
    LDA    ADRESS        ;end of display list
    STA    SAVADR        ;save address
    LDA    ADRESS+1
    STA    SAVADR+1

;    Set up.

    LDA    #$41            ;ANTIC wait for VBLANK and DMF
    JSR    SDI            ;store data indirect

```

```

    STX    OPNTMP
    LDA    #24
    STA    BOTSCR

;    Check for modes 9 ,10 and 11.

    LDA    DINDEX        ;mode
    CMP    #12
    BCS    COC10          ;if mode >= 12, mixed mode OK

    CMP    #9
    BCS    COC12          ;if mode >= 9, mixed mode not allowed

;    Check for mixed mode.
COC10 LDA    ICAX1Z
    AND    #MXDMOD
    BEQ    COC12          ;if not mixed mode

;    Process mixed mode.

    LDA    #4
    STA    BOTSCR
    LDX    #2
    LDA    FINE
    BEQ    COC11          ;if not fine scrolling

    JSR    SSE            ;set scrolling display list entry

COC11 LDA    #$02
    JSR    SDF            ;store data indirect for fine scrolling
    DEX
    BPL    COC11          ;if not done

;    Reload MSC for text.

    LDY    RAMTOP        ;(high) RAM size
    DEY                ;decrement (high) RAM size
    TYA
    JSR    SDI            ;store data indirect
    LDA    #low [$0000-160] ;low RAM size - 160
    JSR    SDI            ;store data indirect
    LDA    #$42
    JSR    SDF            ;store data indirect
    CLC
    LDA    #MXDMOD
    ADC    OPNTMP
    TAY
    LDX    TDLE,Y
    BNE    COC13

;    Check mode.
COC12 LDY    OPNTMP
    LDX    TDLE,Y        ;number of display list entries
    LDA    DINDEX        ;mode
    BNE    COC13          ;if not mode 0

;    Check for fine scrolling.

    LDA    FINE            ;fine scrolling flag
    BEQ    COC13          ;if not fine scrolling

;    Process fine scrolling.

    JSR    SSE            ;set scrolling display list entry
    LDA    #$22

```

```

        STA    HOLD1

;        Continue.

COC13  LDA    HOLD1
        JSR    SDI            ;store data indirect
        DEX
        BNE    COC13         ;if not done

;        Determine mode.

        LDA    DINDEX        ;mode
        CMP    #8
        BCC    COC16         ;if mode < 8

        CMP    #15
        BEQ    COC14         ;if mode 15

        CMP    #12
        BCS    COC16         ;if mode >= 12

;        Process modes 8, 9, 10, 11 and 15.

COC14  LDX    #93             ;remaining number of DLE's
        LDA    RAMTOP        ;(high) RAM size
        SEC
        SBC    #high $1000    ;subtract 4K
        JSR    SDI            ;store data indirect
        LDA    #low $0000
        JSR    SDI            ;store data indirect
        LDA    HOLD1         ;ANTIC MSC code
        ORA    #$40
        JSR    SDI            ;store data indirect

COC15  LDA    HOLD1         ;remaining DLE's
        JSR    SDI            ;store data indirect
        DEX
        BNE    COC15         ;if DLE's remain

;        Complete display list with LMS.

COC16  LDA    SAVMSC+1        ;high saved memory scan counter
        JSR    SDI            ;store data indirect
        LDA    SAVMSC        ;low saved memory scan counter
        JSR    SDI            ;store data indirect
        LDA    HOLD1
        ORA    #$40
        JSR    SDI            ;store data indirect
        LDA    #$70          ;8 blank lines
        JSR    SDI            ;store data indirect
        LDA    #$70          ;8 blank lines
        JSR    SDI            ;store data indirect
        LDA    ADDRESS        ;display list address
        STA    SDLSTL         ;save display list address
        LDA    ADDRESS+1
        STA    SDLSTL+1
        LDA    #$70          ;8 blank lines
        JSR    SDI            ;store data indirect
        LDA    ADDRESS        ;display list address
        STA    MEMTOP         ;update top of memory
        LDA    ADDRESS+1
        STA    MEMTOP+1
        LDY    #1             ;offset
        LDA    SDLSTL         ;saved display list address
        STA    (SAVADR),Y
        INY
        LDA    SDLSTL+1

```

```

        STA    (SAVADR),Y
;      Check status.

        LDA    DSTAT      ;status
        BPL    COC18      ;if no error

;      Process error.

COC17  STA    DERRF      ;screen OPEN error flag
        JSR    EOP        ;perform editor OPEN
        LDA    DERRF      ;restore status
        LDY    #0         ;no screen OPEN error indicator
        STY    DERRF      ;screen OPEN error flag
        TAY          ;status
        RTS             ;return

;      Check clear inhibit.

COC18  LDA    ICAX1Z
        AND    #$20      ;extract clear inhibit bit
        BNE    COC19      ;if clear inhibited

;      Clear screen.

        JSR    CSC        ;clear screen
        STA    TXTROW     ;set cursor at top row
        LDA    LMARGN     ;left margin
        STA    TXTCOL     ;set cursor at left margin

;      Exit.

COC19  LDA    #$22        ;turn on DMA control
        ORA    SDMCTL
        STA    SDMCTL
        JMP    SEC        ;set exit conditions, return

```

```

**      SGB - Perform Screen GET-BYTE
*
*      ENTRY JSR    SGB
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

SGB    =    *      ;entry
        JSR    CCR    ;check cursor range
        JSR    GDC    ;get data under cursor
        JSR    CIA    ;convert internal character to ATASCII
        JSR    SZA    ;set zero data and advance cursor
        JMP    SST    ;perform screen STATUS, return

```

```

**      GDC - Get Data under Cursor
*
*      ENTRY JSR    GDC
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

GDC    =    *      ;entry
        JSR    CCA    ;convert cursor row/column to address

```



```

        LDA    (ADDRESS),Y
        AND    DMASK

GDC1   LSR     SHFAMT ;shift data down to low bits
        BCS    GDC2  ;if done

        LSR     A
        BPL    GDC1  ;continue shifting

GDC2   STA     CHAR
        CMP     #0    ;retore flags
        RTS      ;return

```

```

**      SPB - Perform Screen PUT-BYTE
*
*      ENTRY JSR    SPB
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

SPB     =      *      ;entry
        STA     ATACHR

;        JSR     ROD    ;restore old data under cursor

        CMP     #CLS
        BNE     SPB1    ;if not clear screen

        JSR     CSC     ;clear screen
        JMP     SEC     ;set exit conditions, return

SPB1    JSR     CCR     ;check cursor range
;        JMP     CEL     ;check EOL, return

```

```

**      CEL - Check End of Line
*
*      ENTRY JSR    CEL
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

CEL     =      *      ;entry
        LDA     ATACHR
        CMP     #EOL
        BNE     CEL1    ;if not EOL

        JSR     RWS     ;return with scrolling
        JMP     SEC     ;set exit conditions, return

CEL1    JSR     PLO     ;plot point
        JSR     SEA     ;set EOL data and advance cursor
        JMP     SEC     ;set exit conditions, return

```

```

**      PLO - Plot Point
*
*      ENTRY JSR    PLO
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PLO      =      *          ;entry

;      Wait for start/stop flag clear.

PLO0     LDA      SSFLAG      ;start/stop flag
          BNE      PLO0       ;if start/stop flag non-zero, loop

;      Save cursor row/column.

          LDX      #2          ;offset to last byte

PLO1     LDA      ROWCRS,X     ;byte of cursor row/column
          STA      OLDROW,X    ;save byte of cursor row/column
          DEX
          BPL      PLO1       ;if not done

;      Convert ATASCII character to internal.

          LDA      ATACHR      ;character
          TAY          ;character
          ROL      A
          ROL      A
          ROL      A
          ROL      A
          AND      #3
          TAX          ;index into TAIC
          TYA          ;character
          AND      #$9F       ;strip off column address
          ORA      TAIC,X     ;or in new column address
;          JMP      SPQ        ;display, return

```

```

**      SPQ - Display
*
*      ENTRY JSR    SPQ
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SPQ      =      *          ;entry

;      Set CHAR.

          STA      CHAR        ;character

;      Convert cursor row/column to address.

          JSR      CCA         ;convert cursor row/column to address

;      Shift up to proper position.

          LDA      CHAR        ;character

SPQ1     LSR      SHFAMT

```

```

    BCS    SPQ2            ;if done

    ASL    A
    JMP    SPQ1            ;continue shifting

;    Update data.

SPQ2    AND    DMASK
    STA    TMPCHR        ;save shifted data
    LDA    DMASK        ;display mask
    EOR    #$FF        ;complement display mask
    AND    (ADDRESS),Y    ;mask off old data
    ORA    TMPCHR        ;or in new data
    STA    (ADDRESS),Y    ;update data
    RTS                ;return

```

```

**    SEC - Set Exit Conditions
*
*    ENTRY JSR    SEC
*
*    MODS
*
*    Original Author Unknown
*    1. Bring closer to Coding Standard (object unchanged).
*    R. K. Nordin    1983-11-01

```

```

SEC    =    *            ;entry
    JSR    GDC            ;get data under cursor
    STA    OLDCHR
    LDX    DINDEX        ;mode
    BNE    SST            ;if graphics, no cursor

    LDX    CRSINH        ;cursor inhibit flag
    BNE    SST            ;if cursor inhibited

    EOR    #$80        ;complement most significant bit
    JSR    SPQ            ;display
;    JMP    SST            ;perform screen status, return

```

```

**    SST - Perform Screen STATUS
*
*    ENTRY JSR    SST
*
*    MODS
*
*    Original Author Unknown
*    1. Bring closer to Coding Standard (object unchanged).
*    R. K. Nordin    1983-11-01

```

```

SST    =    *            ;entry
    LDY    DSTAT        ;status
    JMP    SST1        ;continue

```

\$F223 Patch

FIX \$F223

<pre>** F223 - \$F223 Patch * * For compatibility with OS Revision B, perform power-up display.</pre>

```
PPD      =      *      ;entry
          JMP    SES    ;select and execute self-test
```

Keyboard, Editor and Screen Handler, Part 3

```
;      Continue.

SST1  LDA    #SUCCE$      ;indicate success
      STA    DSTAT      ;status
      LDA    ATACHR      ;data
;      JMP    ESP      ;return
```

```
**      ESP - Perform Editor SPECIAL
*
*      ESP does nothing.
*
*      ENTRY JSR    ESP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
ESP    =      *      ;entry
      RTS      ;return
```

```
**      ECL - Perform Editor CLOSE
*
*      ENTRY JSR    ECL
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
ECL    =      *      ;entry

;      Check for fine scrolling.

      BIT    FINE      ;fine scrolling flag
      BPL    SST      ;if not fine scrolling, perform STATUS, return

;      Process fine scrolling.

      LDA    #$40
      STA    NMIEN      ;disable DLI
      LDA    #0          ;clear fine scrolling flag
      STA    FINE
      LDA    #low RIR      ;return from interrupt routine
      STA    VDSLST      ;restore initial DLI vector value
      LDA    #high RIR
      STA    VDSLST+1
      JMP    EOP          ;perform editor OPEN, return
```

```
**      EGB - Perform Editor GET-BYTE
*
*      ENTRY JSR    EGB
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
EGB    =      *      ;entry
```

```

;      Initialize.

      JSR    SWA      ;swap
      JSR    CRE      ;check cursor range for editor
      LDA    BUFCNT   ;buffer count
      BNE    EGB4     ;if something in the buffer

;      Get line.

      LDA    ROWCRS    ;cursor row
      STA    BUFSTR     ;buffer start pointer
      LDA    COLCRS    ;low cursor column
      STA    BUFSTR+1   ;high buffer start pointer

EGB1  JSR    KGB      ;perform keyboard GET-BYTE
      STY    DSTAT     ;status
      LDA    ATACHR    ;ATASCII character
      CMP    #EOL
      BEQ    EGB3      ;if EOL

      JSR    PCH      ;process character
      JSR    SWA      ;swap
      LDA    LOGCOL    ;logical column
      CMP    #113     ;column near column 120
      BNE    EGB2      ;if not near column 120, no beep

      JSR    BEL      ;beep

EGB2  JMP    EGB1     ;process next character

;      Process EOL.

EGB3  JSR    ROD      ;restore old data under cursor
      JSR    CBC      ;compute buffer count
      LDA    BUFSTR     ;buffer start pointer
      STA    ROWCRS    ;cursor row
      LDA    BUFSTR+1   ;high buffer start pointer
      STA    COLCRS    ;low cursor column

;      Check buffer count.

EGB4  LDA    BUFCNT   ;buffer count
      BEQ    EGB6     ;if buffer count zero

;      Decrement and check buffer count.

EGB5  DEC    BUFCNT   ;decrement buffer count
      BEQ    EGB6     ;if buffer count zero

;      Check status.

      LDA    DSTAT     ;status
      BMI    EGB5     ;if error, continue decrementing.

;      Perform GET-BYTE.

      JSR    SGB      ;perform screen GET-BYTE
      STA    ATACHR    ;ATASCII character
      JMP    SWA      ;swap, return

;      Exit.

EGB6  JSR    RWS      ;return with scrolling
      LDA    #EOL
      STA    ATACHR    ;ATASCII character
      JSR    SEC      ;set exit conditions
      STY    DSTAT     ;status

```

JMP SWA ;swap, return

```
**      IRA - Invoke Routine Pointed to by ADRESS
*
*      ENTRY JSR    IRA
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01
```

IRA = * ;entry
JMP (ADRESS) ;execute, return

```
**      EPB - Perform Editor PUT-BYTE
*
*      ENTRY JSR    EPB
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01
```

EPB = * ;entry
STA ATACHR ;ATASCII character
JSR SWA ;swap
JSR CRE ;check cursor range for editor
LDA #0
STA SUPERF ;clear super function flag
; **JMP** PCH ;process character, return

```
**      PCH - Process Character
*
*      PCH displays the character or processes control characters and
*      super functions (shifted function keys).
*
*      ENTRY JSR    PCH
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01
```

PCH = * ;entry
JSR R0D ;restore old data under cursor
JSR CCC ;check for control character
BEQ PCH2 ;if control character

; Display character.

PCH1 **ASL** ESCFLG ;escape flag
JSR CEL ;check EOL
JMP SWA ;swap, return

; Process control character.

PCH2 **LDA** DSPFLG ;display flag
ORA ESCFLG ;escape flag
BNE PCH1 ;if display or escape, display character

; Continue.

ASL ESCFLG

```

    INX

;    Check for super function.

    LDA    SUPERF
    BEQ    PCH3          ;if not super function

;    Adjust for super function.

    TXA
    CLC
    ADC    #TSFR-TCCR-3
    TAX                    ;adjusted offset

;    Process control character or super function.

PCH3  LDA    TCCR,X        ;low routine address
      STA    ADDRESS
      LDA    TCCR+1,X      ;high routine address
      STA    ADDRESS+1
      JSR    IRA           ;invoke routine pointed to by ADDRESS
      JSR    SEC           ;set exit conditions
      JMP    SWA           ;swap, return

```

```

**    IGN - Ignore Character and Perform Keyboard GET-BYTE
*
*    ENTRY JSR    IGN
*
*    EXIT
*
*    CH = $FF
*
*    MODS
*
*    Original Author Unknown
*    1. Bring closer to Coding Standard (object unchanged).
*    R. K. Nordin    1983-11-01

```

```

IGN    =    *    ;entry
      LDA    #$FF ;clear code indicator
      STA    CH   ;key code
;      JMP    KGB ;perform keyboard GET-BYTE, return

```

```

**    KGB - Perform Keyboard GET-BYTE
*
*    ENTRY JSR    KGB
*
*    NOTES
*
*    Problem: byte wasted by unnecessary TAX near KGB3.
*
*    MODS
*
*    Original Author Unknown
*    1. Bring closer to Coding Standard (object unchanged).
*    R. K. Nordin    1983-11-01

```

```

KGB    =    *    ;entry

;    Initialize.

KGB1   LDA    #0
      STA    SUPERF ;clear super function flag

;    Check for special edit read mode.

      LDA    ICAX1Z
      LSR    A

```



```

    BCS    KGB11 ;if special edit read mode
;
; Check for BREAK abort.
    LDA    #BRKABT ;assume BREAK abort
    LDX    BRKKEY ;BREAK key flag
    BEQ    KGB10 ;if BREAK abort
;
; Check for character.
    LDA    CH ;key code
    CMP    #$FF ;clear code indicator
    BEQ    KGB1 ;if no character
;
; Process character.
    STA    HOLDCH ;save character
    LDX    #$FF ;clear code indicator
    STX    CH ;key code
;
; Sound key click if desired.
    LDX    NOCLIK ;click inhibit flag
    BNE    KGB2 ;if click inhibited
    JSR    SKC ;sound key click
;
; Set offset to key definition.
KGB2    TAY ;save character
;
; Check for CTRL and SHIFT together.
    CPY    #$C0
    BCS    IGN ;if CTRL and SHIFT together, ignore
;
; Convert to ATASCII character.
    LDA    (KEYDEF),Y ;ATASCII character
;
; Set ATASCII character.
KGB3    STA    ATACHR ;ATASCII character
    TAX
    BMI    KGB4 ;if special key
    JMP    KGB17 ;process shift/control lock
;
; Check for null character.
KGB4    CMP    #$80
    BEQ    IGN ;if null, ignore
;
; Check for inverse video key.
    CMP    #$81
    BNE    KGB5 ;if not inverse video key
;
; Process inverse video key.
    LDA    INVFLG
    EOR    #$80
    STA    INVFLG
    BCS    IGN ;ignore
;
; Check for CAPS key.

```

```

KGB5  CMP    #$82
      BNE    KGB6    ;if not CAPS key

;      Process CAPS key.

      LDA    SHFLOK ;shift/control lock flags
      BEQ    KGB7    ;if no lock, process CAPS lock

      LDA    #$00    ;no lock indicator
      STA    SHFLOK ;shift/control lock flags
      BEQ    IGN     ;ignore

;      Check for SHIFT-CAPS key.

KGB6  CMP    #$83
      BNE    KGB8    ;if not SHIFT-CAPS

;      Process SHIFT-CAPS key.

KGB7  LDA    #$40    ;CAPS lock indicator
      STA    SHFLOK ;shift/control lock flags
      BNE    IGN     ;ignore

;      Check for CTRL-CAPS key.

KGB8  CMP    #$84
      BNE    KGB9    ;if not CTRL-CAPS

;      Process CTRL-CAPS key.

      LDA    #$80    ;control lock indicator
      STA    SHFLOK ;shift/control lock flags
      JMP    IGN     ;ignore

;      Check for CTRL-3 key.

KGB9  CMP    #$85
      BNE    KGB12   ;if not CTRL-3 key.

;      Process CTRL-3 key.

      LDA    #EOFERR

;      Set status and BREAK key flag.

KGB10 STA    DSTAT ;status
      STA    BRKKEY ;BREAK key flag

;      Set EOL character.

KGB11 LDA    #EOL
      JMP    KGB19  ;set ATASCII character

;      Check for CTRL-F3 key.

KGB12 CMP    #$89
      BNE    KGB14  ;if not CTRL-F3 key

;      Process CTRL-F3 key.

      LDA    NOCLIK ;toggle keyclick status
      EOR    #$FF
      STA    NOCLIK
      BNE    KGB13  ;if click inhibited

      JSR    SKC    ;sound key click

```

```

KGB13  JMP     IGN      ;ignore
;      Check for function key.

KGB14  CMP     #$8E
      BCS     KGB16 ;if code >= $8E, not a function key

      CMP     #$8A
      BCC     KGB13 ;if code < $8A, not a function key, ignore
;      Process function key.

      SBC     #$8A      ;convert $8A - $8D TO 0 - 3
      ASL     HOLDCH    ;saved character
      BPL     KGB15     ;if no SHIFT

      ORA     #$04      ;convert 0 - 3 to 4 - 7

KGB15  TAY     ;offset to function key definition
      LDA     (FKDEF),Y ;function key
      JMP     KGB3      ;set ATASCII character
;      Check for super function.

KGB16  CMP     #$92
      BCS     KGB17 ;if code >= $92, process shift/control lock

      CMP     #$8E
      BCC     KGB13 ;if code < $8E, not super function, ignore
;      Process super function.

      SBC     #$8E-$1C  ;convert $8E - $91 TO $1C - $1F
      INC     SUPERF    ;set super function flag
      BNE     KGB19     ;set ATASCII character
;      Process shift/control lock.

KGB17  LDA     HOLDCH ;saved character
      CMP     #$40
      BCS     KGB18 ;if not lower case

      LDA     ATACHR ;ATASCII character
      CMP     #'a'
      BCC     KGB18 ;if < "a", do not process

      CMP     #'z'+1
      BCS     KGB18 ;if > "z", do not process

      LDA     SHFLOK ;shift/control lock flags
      BEQ     KGB18 ;if no lock

      ORA     HOLDCH ;modify character
      JMP     KGB2      ;reprocess character
;      Invert character, if necessary.

KGB18  JSR     CCC      ;check for control character
      BEQ     KGB20 ;if control character, do not invert

      LDA     ATACHR ;ATASCII character
      EOR     INVFLG ;invert character
;      Set ATASCII character.

KGB19  STA     ATACHR ;ATASCII character

```

```
;      Exit.

KGB20  JMP    SST    ;perform screen status, return
```

```
**      ESC - Escape
*
*      ENTRY JSR    ESC
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01
```

```
ESC    =      *      ;entry
      LDA    #$80    ;indicate escape detected
      STA    ESCFLG  ;escape flag
      RTS          ;return
```

```
**      CUP - Move Cursor Up
*
*      ENTRY JSR    CUP
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01
```

```
CUP    =      *      ;entry
      DEC    ROWCRS  ;decrement cursor row
      BPL    CUP2    ;if row positive

      LDX    BOTSCR  ;screen bottom
      DEX          ;screen bottom - 1

CUP1    STX    ROWCRS ;update cursor row

CUP2    JMP    SBS    ;set buffer start and logical column, return
```

```
**      CDN - Move Cursor Down
*
*      ENTRY JSR    CDN
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01
```

```
CDN    =      *      ;entry
      INC    ROWCRS  ;increment cursor row
      LDA    ROWCRS  ;cursor row
      CMP    BOTSCR  ;screen bottom
      BCC    CUP2    ;if at bottom, set buffer start, return

      LDX    #0
      BEQ    CUP1    ;update cursor row, return
```

```

**      CLF - Move Cursor Left
*
*      ENTRY JSR      CLF
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CLF      =      *          ;entry
DEC      COLCRS ;decrement low cursor column
LDA      COLCRS ;low cursor column
BMI      CRM     ;if negative, move cursor to margin, return

CMP      LMARGN ;left margin
BCS      SCC1    ;if at left margin, set logical column, return

;      JMP      CRM     ;move cursor to right margin, return

```

```

**      CRM - Move Cursor to Right Margin
*
*      ENTRY JSR      CRM
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CRM      =      *          ;entry
LDA      RMARGN ;right margin
;      JMP      SCC     ;set cursor column, return

```

```

**      SCC - Set Cursor Column
*
*      ENTRY JSR      SCC
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SCC      =      *          ;entry
STA      COLCRS ;set low cursor column

SCC1     JMP      SLC     ;set logical column, return

```

```

**      CRT - Move Cursor Right
*
*      ENTRY JSR      CRT
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CRT      =      *          ;entry
INC      COLCRS ;increment low cursor column
LDA      COLCRS ;low cursor column
CMP      RMARGN ;right margin
BCC      SCC1    ;if before right margin, process, return

BEQ      SCC1    ;if at right margin

```

```
;      JMP      CLM      ;move cursor to left margin, return
```

```
**      CLM - Move Cursor to Left Margin
*
*      ENTRY JSR      CLM
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
CLM      =      *      ;entry
      LDA      LMARGN ;left margin
      JMP      SCC      ;set cursor column, return
```

```
**      CSC - Clear Screen
*
*      ENTRY JSR      CSC
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
CSC      =      *      ;entry
;      Set memory scan counter address.
      JSR      SMS      ;set memory scan counter address
;      Clear address.
      LDY      ADRESS
      LDA      #0
      STA      ADRESS
CSC1     STA      (ADRESS),Y
      INY
      BNE      CSC1      ;if not done with page
      INC      ADRESS+1
      LDX      ADRESS+1
      CPX      RAMTOP      ;(high) RAM size
      BCC      CSC1      ;if not done
;      Clean up logical line bit map.
;      LDY      #0      ;offset to first byte of bit map
      LDA      #$FF
CSC2     STA      LOGMAP,Y      ;byte of logical line bit map
      INY
      CPY      #4      ;4 bytes
      BCC      CSC2      ;if not done
;      Exit.
;      JMP      CHM      ;move cursor home, return
```

```

**      CHM - Move Cursor Home
*
*      ENTRY JSR      CHM
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

CHM      =      *      ;entry
          JSR      SCL      ;set cursor at left edge
          STA      LOGCOL    ;logical column
          STA      BUFSTR+1  ;high buffer start
          LDA      #0
          STA      ROWCRS    ;cursor row
          STA      COLCRS+1  ;high cursor column
          STA      BUFSTR    ;low buffer start pointer
          RTS

```

```

**      BSP - Backspace
*
*      ENTRY JSR      BSP
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

BSP      =      *      ;entry
          LDA      LOGCOL    ;logical column
          CMP      LMARGN    ;left margin
          BEQ      BSP3      ;if at left margin

          LDA      COLCRS    ;low cursor column
          CMP      LMARGN    ;left margin
          BNE      BSP1      ;if not at left margin

          JSR      DWQ      ;see if line should be deleted

BSP1     JSR      CLF      ;move cursor left
          LDA      COLCRS    ;low cursor column
          CMP      RMARGN    ;right margin
          BNE      BSP2      ;if not at right margin

          LDA      ROWCRS    ;cursor row
          BEQ      BSP2      ;if row zero

          JSR      CUP      ;move cursor up

BSP2     LDA      #' '
          STA      ATACHR    ;ATASCII character
          JSR      PLO      ;plot point

BSP3     JMP      SLC      ;set logical column, return

```

```

**      TAB - Tab
*
*      ENTRY JSR    TAB
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

TAB      =      *      ;entry

TAB1     JSR      CRT      ;move cursor right
          LDA      COLCRS  ;low cursor column
          CMP      LMARGN  ;left margin
          BNE      TAB2    ;if not at left margin

          JSR      RET      ;return
          JSR      BLG      ;get bit from logical line bit map
          BCS      TAB3    ;if end of logical line

;        Check for tab stop.

TAB2     LDA      LOGCOL  ;logical column
          JSR      BMG      ;get bit from bit map
          BCC      TAB1    ;if not tab stop, keep looking

;        Set logical column.

TAB3     JMP      SLC      ;set logical column, return

```

```

**      STB - Set Tab
*
*      ENTRY JSR    STB
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

STB      =      *      ;entry
          LDA      LOGCOL  ;logical column
          JMP      BMS      ;set bit in bit map, return

```

```

**      CTB - Clear Tab
*
*      ENTRY JSR    CTB
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CTB      =      *      ;entry
          LDA      LOGCOL  ;logical column
          JMP      BMC      ;clear bit in bit map, return

```



```

**      ICH - Insert Character
*
*      ENTRY JSR      ICH
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

ICH      =      *      ;entry
          JSR      SRC      ;save row and column
          JSR      GDC      ;get data under cursor
          STA      INSDAT
          LDA      #0
          STA      SCRFLG

ICH1     JSR      SPQ      ;store data
          LDA      LOGCOL  ;logical column
          PHA      ;save logical column
          JSR      ACC      ;advance cursor column
          PLA      ;saved logical column
          CMP      LOGCOL  ;logical column
          BCS      ICH2    ;if saved logical column >= logical column, exit

          LDA      INSDAT
          PHA
          JSR      GDC      ;get data under cursor
          STA      INSDAT
          PLA
          JMP      ICH1    ;continue

;      Exit.

ICH2     JSR      RRC      ;restore row and column

ICH3     DEC      SCRFLG
          BMI      ICH4    ;if scroll occurred

          DEC      ROWCRS  ;decrement cursor row
          BNE      ICH3    ;continue

ICH4     JMP      SLC      ;set logical column, return

```

```

**      DCH - Delete Character
*
*      ENTRY JSR      DCH
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

DCH      =      *      ;entry

;      Save row and column.

          JSR      SRC      ;save row and column

;      Get data to the right of cursor.

DCH1     JSR      CCA      ;convert cursor row/column to address
          LDA      ADDRESS
          STA      SAVADR  ;save address
          LDA      ADDRESS+1

```

```

STA SAVADR+1
LDA LOGCOL ;logical column
PHA ;save logical column
JSR SZA ;set zero data and advance cursor
PLA ;saved logical column
CMP LOGCOL ;logical column
BCS DCH2 ;if saved logical column >= logical column, exit

LDA ROWCRS ;cursor row
CMP BOTSCR ;screen bottom
BCS DCH2 ;if row off screen, exit

JSR GDC ;get data under cursor
LDY #0
STA (SAVADR),Y ;put data in previous position
BEQ DCH1 ;continue

DCH2 LDY #0
TYA
STA (SAVADR),Y ;clear last position
JSR DQD ;try to delete a line
JSR RRC ;restore row and column
JMP SLC ;set logical column, return

```

```

** ILN - Insert Line
*
* ENTRY JSR ILN
*
* MODS
* Original Author Unknown
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01

```

```

ILN = * ;entry
; SEC
; JMP ILN1

```

```

** ILN1 - Insert Line
*
* ENTRY JSR ILN1
*
* MODS
* Original Author Unknown
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01

```

```

ILN1 = * ;entry
JSR ELL ;extend logical line
LDA LMARGN ;left margin
STA COLCRS ;low cursor column
JSR CCA ;convert cursor row/column to address
JSR MLN ;move line
JSR CLN ;clear current line
JMP SLC ;set logical column, return

```

```

**      DLN - Delete Line
*
*      ENTRY JSR      DLN
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

DLN      =      *      ;entry
          JSR      SLC      ;set logical column
          LDY      HOLD1
          STY      ROWCRS ;cursor row
          JMP      DLN1

```

```

**      DLN1 - Delete Line
*
*      ENTRY JSR      DLN1
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

DLN1      =      *      ;entry
DLN0      LDY      ROWCRS      ;cursor row
DLN2      TYA
          SEC
          JSR      BLG2      ;get next bit
          PHP
          TYA
          CLC
          ADC      #8*[LOGMAP-TABMAP] ;add offset for logical line
          PLP
          JSR      BMP      ;put bit in bit map
          INY
          CPY      #24
          BNE      DLN2      ;if not done

          LDA      LOGMAP+2
          ORA      #1      ;set least significant bit
          STA      LOGMAP+2 ;update logical line bit map
          LDA      #0      ;delete line of data
          STA      COLCRS   ;low cursor column
          JSR      CCA      ;convert cursor row/column to address
          JSR      SSD      ;scroll screen for delete

;      Check for new logical line.

          JSR      BLG      ;get bit from logical line bit map
          BCC      DLN0      ;if not new logical line

;      Move cursor to left margin.

          JMP      CLM      ;move cursor to left margin, return

```

```

**      BEL - Sound Bell
*
*      ENTRY JSR    BEL
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

BEL      =      *      ;entry
          LDY    #$20
BEL1     JSR    SKC      ;sound key click
          DEY
          BPL    BEL1    ;if not done
          RTS          ;return

```

```

**      CBT - Move Cursor to Bottom
*
*      ENTRY JSR    CBT
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CBT      =      *      ;entry
          JSR    CHM      ;move cursor home
          JMP    CUP      ;move cursor up, return

```

```

**      DDD - Perform Double Byte Double Decrement
*
*      ENTRY JSR    DDD
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DDD      =      *      ;entry
          LDA    #2      ;indicate subtracting 2
          BNE    DBS      ;perform double byte subtract, return

```

```

**      SDF - Store Data Indirect for Fine Scrolling
*
*      ENTRY JSR    SDF
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SDF      =      *      ;entry
          LDY    FINE
          BEQ    SDI      ;if not fine scrolling
          ORA    #$20      ;enable vertical scroll
          JMP    SDI      ;store data indirect, return
;

```

```

**      SDI - Store Data Indirect
*
*      ENTRY JSR    SDI
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SDI      =      *      ;entry
;
;      Check current status.
;
;      LDY      DSTAT      ;status
;      BMI      DBS3      ;if error, return
;
;      Store data.
;
;      LDY      #0
;      STA      (ADRESS),Y
;
;      Decrement.
;
;      INP      DSD      ;perform double byte single decrement, return

```

```

**      DSD - Perform Double Byte Single Decrement
*
*      ENTRY JSR    DSD
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DSD      =      *      ;entry
;      LDA      #1      ;indicate subtracting 1
;      INP      DBS      ;perform double byte subtract, return

```

```

**      DBS - Perform Double Byte Subtract
*
*      ENTRY JSR    DBS
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

DBS      =      *      ;entry
;
;      Initialize.
;
;      STA      SUBTMP
;
;      Check current status.
;
;      LDA      DSTAT      ;status
;      BMI      DBS3      ;if error
;
;      Subtract.
;
;      LDA      ADRESS
;      SEC
;      SBC      SUBTMP

```

```

    STA    ADRESS
    BCS    DBS1          ;if no borrow

    DEC    ADRESS+1      ;adjust high byte

;    Check for overwriting APPMHI.

DBS1    LDA    APPMHI+1
        CMP    ADRESS+1
        BCC    DBS3      ;if not overwriting APPMHI

        BNE    DBS2      ;if overwriting APPMHI, error

        LDA    APPMHI
        CMP    ADRESS
        BCC    DBS3      ;if not overwriting APPMHI

;    Process error.

DBS2    LDA    #SCRMEM    ;indicate insufficient memory for screen
        STA    DSTAT      ;status

;    Exit.

DBS3    RTS              ;return

```

```

**      SSE - Set Scrolling Display List Entry
*
*      Store extra line in display list for fine scrolling modes.
*
*      ENTRY JSR    SSE
*
*      MODS
*          H. Stewart    1982-06-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin   1983-11-01

```

```

SSE    =    *          ;entry
        LDA    #$02
        JSR    SDI      ;store data indirect
        LDA    #$A2     ;DLI on last visible line
        JSR    SDI      ;store data indirect
        DEX
        RTS              ;return

```

```

**      CCA - Convert Cursor Row/Column to Address
*
*      ENTRY JSR    CCA
*
*      MODS
*          L. Winner     1982-06-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin   1983-11-01

```

```

CCA    =    *          ;entry
        LDX    #1
        STX    MLTTMP    ;initialize
        DEX
        STX    ADRESS+1    ;clear high address
        LDA    ROWCRS      ;cursor row position
        ASL    A            ;2 times row position
        ROL    ADRESS+1
        ASL    A            ;4 time row position
        ROL    ADRESS+1

```

	ADC	ROWCRS	;add to get 5 times row position
	STA	ADRESS	
	BCC	CCA1	
	INC	ADRESS+1	
CCA1	LDY	DINDEX	;mode
	LDX	TLSC,Y	;left shift count
CCA2	ASL	ADRESS	;ADRESS = ADRESS*X
	ROL	ADRESS+1	;divide
	DEX		
	BNE	CCA2	
	LDA	COLCRS+1	;high cursor column
	LSR	A	;save least significant bit
	LDA	COLCRS	;low cursor column
	LDX	TRSC,Y	;right shift count
	BEQ	CCA4	;if no shift
CCA3	ROR	A	;roll in carry
	ASL	MLTTMP	;shift index
	DEX		
	BNE	CCA3	
CCA4	ADC	ADRESS	;add address
	BCC	CCA5	;if no carry
	INC	ADRESS+1	;adjust high address
CCA5	CLC		
	ADC	SAVMSC	;add saved memory scan counter
	STA	ADRESS	;update address
	STA	OLDADR	;save address
	LDA	ADRESS+1	
	ADC	SAVMSC+1	
	STA	ADRESS+1	
	STA	OLDADR+1	
	LDX	TRSC,Y	
	LDA	TMSK,X	
	AND	COLCRS	;and in low cursor column
	ADC	MLTTMP	
	TAY		
	LDA	TDSM-1,Y	;display mask
	STA	DMASK	;display mask
	STA	SHFAMT	
	LDY	#0	
CCA6	RTS		;return

```

**      SZA - Set Zero Data and Advance Cursor Column
*
*      ENTRY JSR      SZA
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

SZA	=	*	;entry
	LDA	#0	
	BEQ	SDA	;set data and advance cursor

```

**      SEA - Set EOL Data and Advance Cursor Column
*
*      ENTRY JSR    SEA
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

SEA      =      *      ;entry
          LDA      #EOL ;special case eliminator
          ; BNE     SDA ;set data and advance cursor, return

```

```

**      SDA - Set Data and Advance Cursor Column
*
*      ENTRY JSR    SDA
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

SDA      =      *      ;entry
          STA      INSDAT ;set data
          ; BNE     ACC ;advance cursor column, return

```

```

**      ACC - Advance Cursor Column
*
*      ENTRY JSR    ACC
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

ACC      =      *      ;entry
          INC      LOGCOL ;increment logical column
          INC      COLCRS ;increment low cursor column
          BNE      ACC1   ;if no carry

          INC      COLCRS+1 ;adjust high cursor column

ACC1     LDA      COLCRS ;low cursor column
          LDX      DINDEX ;mode
          CMP      TMCC,X
          BEQ      ACC2   ;if equal, process EOL

          CPX      #0
          BNE      CCA6   ;if not mode 0, exit

          CMP      RMARGN ;right margin
          BEQ      CCA6   ;if at right margin, exit

          BCC      CCA6   ;if before right margin, exit

ACC2     CPX      #8
          BNE      ACC3   ;if not mode 8

          LDA      COLCRS+1 ;high cursor column
          BEQ      CCA6   ;if only at 64

ACC3     LDA      DINDEX ;mode
          BNE      RET    ;if mode 0, exit

```



```

    LDA    LOGCOL        ;logical column
    CMP    #81
    BCC    ACC4          ;if < 81, definitely not line 3

    LDA    INSDAT
    BEQ    RET           ;if non-zero, do not do log line

    JSR    RWS           ;return with scrolling
    JMP    RET5          ;return

ACC4   JSR    RET         ;return
        LDA    ROWCRS     ;cursor row
        CLC
        ADC    #8*[LOGMAP-TABMAP] ;add offset for logical line
        JSR    BMG        ;get bit from bit map
        BCC    ACC5

        LDA    INSDAT
        BEQ    ACC5      ;if zero, do not extend

        CLC
        JSR    ILN1      ;insert line

ACC5   JMP    SLC        ;set logical column, return

```

```

**      RWS - Return with Scrolling
*
*      ENTRY JSR    RWS
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

RWS    =      *          ;entry
        LDA    #EOL      ;select scrolling
        STA    INSDAT
;      JMP    RET        ;return, return

```

```

**      RET - Return
*
*      ENTRY JSR    RET
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

RET    =      *          ;entry
        JSR    SCL        ;set cursor at left edge
        LDA    #0
        STA    COLCRS+1   ;high cursor column
        INC    ROWCRS     ;increment cursor row
        LDX    DINDEXT
        LDY    #24        ;assume 24 lines
        BIT    SWPFLG
        BPL    RET1       ;if normal

        LDY    #4         ;substitute 4 lines
        TYA
        BNE    RET2

RET1   LDA    TMRC,X      ;mode row count

```

```

RET2  CMP    ROWCRS ;cursor row
      BNE    RET5

      STY    HOLD3
      TXA    ;mode
      BNE    RET5 ;if mode not 0, do not scroll

      LDA    INSDAT
      BEQ    RET5 ;if zero, do not scroll

;      If EOL, roll in a 0.

      CMP    #EOL ;to extend bottom logical line
      BEQ    RET3 ;if EOL

      CLC

RET3   JSR    SCR
      INC    SCRFLG
      DEC    BUFSTR
      BPL    RET4

      INC    BUFSTR

RET4   DEC    HOLD3
      LDA    LOGMAP
      SEC    ;indicate for partial line
      BPL    RET3 ;if partial logical line

      LDA    HOLD3
      STA    ROWCRS ;cursor row

RET5   JMP    SLC ;set logical column, return

```

```

**      SEP - Subtract End Point
*
*      ENTRY JSR    SEP
*              X = 0, if row or 2, if column
*
*      MODS
*              Original Author Unknown
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin      1983-11-01

```

```

SEP    =      *      ;entry

      SEC
      LDA    ROWAC,X ;low value from which to subtract
      SBC    ENDPT
      STA    ROWAC,X ;new low value
      LDA    ROWAC+1,X ;high value from which to subtract
      SBC    ENDPT+1
      STA    ROWAC+1,X ;new high value
      RTS    ;return

```

```

**      CRE - Check Cursor Range for Editor
*
*      ENTRY JSR      CRE
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CRE      =      *      ;entry

;      Check for mixed mode.

      LDA      BOTSCR
      CMP      #4      ;mixed mode indicator
      BEQ      CCR      ;if mixed mode, check cursor range, return

;      Check for mode 0.

      LDA      DINDEX ;mode
      BEQ      CCR      ;if mode 0, check cursor range

;      Open editor.

      JSR      EOP      ;perform editor OPEN
      JMP      CCR      ;check cursor range, return

```

```

**      CCR - Check Cursor Range
*
*      ENTRY JSR      CCR
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CCR      =      *      ;entry
      LDA      #39
      CMP      RMARGN      ;right margin
      BCS      CCR1      ;if 39 >= right margin

      STA      RMARGN      ;set right margin

CCR1     LDX      DINDEX
      LDA      TMRC,X      ;mode row count
      CMP      ROWCRS      ;cursor row
      BCC      CCR5      ;if count > row position, error

      BEQ      CCR5      ;if count = row position, error

      CPX      #8
      BNE      CCR2      ;if not mode 8

      LDA      COLCRS+1      ;high cursor column
      BEQ      CCR4      ;if high cursor column zero

      CMP      #1
      BNE      CCR5      ;if >1, bad

      BEQ      CCR3      ;if 1, check low

CCR2     LDA      COLCRS+1      ;high cursor column
      BNE      CCR5      ;if high cursor column non-zero, error

```

```

CCR3  LDA  TMCC,X      ;mode column count
      CMP  COLCRS     ;low cursor column
      BCC  CCR5        ;if count > column position, error

      BEQ  CCR5        ;if count = column position, error

CCR4  LDA  #SUCCES    ;success indicator
      STA  DSTAT      ;indicate success
      LDA  #BRKABT    ;assume BREAK abort
      LDX  BRKKEY     ;BREAK key status
      STA  BRKKEY     ;clear BREAK key status
      BEQ  CCR6        ;if BREAK

      RTS             ;return

;      Process range error.

CCR5  JSR  CHM         ;move cursor home
      LDA  #CRSROR    ;indicate cursor overrange

;      Exit.

CCR6  STA  DSTAT      ;status
      PLA                ;clean stack for return to CIO
      PLA
      LDA  SWPFLG
      BPL  CCR7        ;if not swapped

      JMP  SWA         ;swap, return

CCR7  JMP  SST         ;return (to CIO)

```

```

**      ROD - Restore Old Data under Cursor
*
*      ENTRY JSR  ROD
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

ROD   =      *          ;entry
      LDY  #0
      LDA  OLDADR+1
      BEQ  ROD1         ;if page zero

      LDA  OLDCHR       ;old data
      STA  (OLDADR),Y

ROD1  RTS              ;return

```

```

**      BMI - Initialize for Bit Map Operation
*
*      BMI sets the bit mask in BITMSK and byte offset in X.
*
*      ENTRY JSR  BMI
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

BMI   =      *          ;entry
      PHA                ;save logical column

```

```

AND    #7      ;logical column modulo 8
TAX    ;offset to bit mask
LDA    TBTM,X  ;bit mask
STA    BITMSK  ;set bit mask
PLA    ;logical column
LSR    A
LSR    A
LSR    A      ;logical column divided by 8
TAX    ;offset
RTS      ;return

```

```

**      BLR - Rotate Logical Line Bit Map Left
*
*      BLR rotates the logical line bit map left, scrolling the
*      logical lines up.
*
*      ENTRY JSR    BLR
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

BLR    =      *      ;entry
ROL    LOGMAP+2
ROL    LOGMAP+1
ROL    LOGMAP
RTS      ;return

```

```

**      BMP - Put Bit in Bit Map
*
*      PUT CARRY INTO BITMAP
*
*      ENTRY JSR    BMP
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

BMP    =      *      ;entry
BCC    BMC      ;if C clear, clear bit in bit map, return
;      JNE      BMS      ;set bit in bit map, return

```

```

**      BMS - Set Bit in Bit Map
*
*      ENTRY JSR    BMS
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

BMS    =      *      ;entry
JSR    BMI      ;initialize for bit mask operation
LDA    TABMAP,X
ORA    BITMSK    ;set bit
STA    TABMAP,X  ;update bit map
RTS      ;return

```

```

**      BMC - Clear Bit in Bit Map
*
*      ENTRY JSR    BMC
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

BMC      =      *      ;entry
          JSR      BMI      ;initialize for bit mask operation
          LDA      BITMSK
          EOR      #$FF
          AND      TABMAP,X      ;clear bit
          STA      TABMAP,X      ;update bit map
          RTS

```

```

**      BLG - Get Bit from Logical Line Bit Map
*
*      ENTRY JSR    BLG
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

BLG      =      *      ;entry
          LDA      ROWCRS      ;cursor row
          ; JMP      BLG1

```

```

**      BLG1 - Get Bit from Logical Line Bit Map
*
*      ENTRY JSR    BLG1
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

BLG1     =      *      ;entry
          CLC
          ; JMP      BLG2

```

```

**      BLG2 - Get Bit from Logical Line Bit Map
*
*      ENTRY JSR    BLG2
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

BLG2     =      *      ;entry
          ADC      #8*[LOGMAP-TABMAP] ;add offset for logical line
          ; JMP      BMG      ;get bit from bit map, return

```

```

**      BMG - Get Bit from Bit Map
*
*      ENTRY JSR      BMG
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

BMG      =      *      ;entry
          JSR      BMI      ;initialize for bit mask operation
          CLC
          LDA      TABMAP,X
          AND      BITMSK
          BEQ      BMG1
          SEC

BMG1     RTS      ;return

```

```

**      CIA - Convert Internal Character to ATASCII
*
*      ENTRY JSR      CIA
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CIA      =      *      ;entry
;      Initialize.

          LDA      CHAR

;      Check mode.

          LDY      DINDEX ;mode
          CPY      #14
          BCS      CIA2      ;if mode >= 14

          CPY      #12
          BCS      CIA1      ;if mode 12 or 13

          CPY      #3
          BCS      CIA2      ;if mode >= 3

;      Convert internal character to ATASCII.

CIA1     ROL      A
          ROL      A
          ROL      A
          ROL      A
          AND      #3
          TAX
          LDA      CHAR      ;character
          AND      #$9F      ;strip off column address
          ORA      TIAC,X ;or in new column address

;      Exit.

CIA2     STA      ATACHR ;ATASCII character

CIA3     RTS      ;return

```

```

**      MLN - Move Line
*
*      ENTRY JSR      MLN
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

MLN      =      *      ;entry

;      Initialize.

      LDX      RAMTOP      ;(high) RAM size
      DEX      ;decrement (high) RAM size
      STX      FRMADR+1    ;high source address
      STX      TOADR+1     ;high destination address
      LDA      #low [$0000-80] ;low RAM size - 80
      STA      FRMADR      ;low source address
      LDA      #low [$0000-40] ;low RAM size - 40
      STA      TOADR      ;low destination address

      LDX      ROWCRS      ;cursor row

;      Check for completion.

MLN1     INX
      CPX      BOTSCR      ;screen bottom
      BEQ      CIA3        ;if done, return

;      Move line.

      LDY      #39          ;offset to last byte

MLN2     LDA      (FRMADR),Y ;byte of source
      STA      (TOADR),Y    ;byte of destination
      DEY
      BPL      MLN2        ;if not done

;      Adjust source and destination addresses.

      SEC
      LDA      FRMADR      ;source address
      STA      TOADR      ;update destination address
      SBC      #low 40     ;subtract 40
      STA      FRMADR      ;update source address
      LDA      FRMADR+1
      STA      TOADR+1
      SBC      #high 40
      STA      FRMADR+1

;      Continue.

      JMP      MLN1        ;continue

```



```

**      ELL - Extend Logical Line
*
*      ENTRY JSR      ELL
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

ELL      =      *      ;entry
          PHP      ;save bit
          LDY      #22

ELL1     TYA
          JSR      BLG1
          PHP
          TYA
          CLC
          ADC      #8*[LOGMAP-TABMAP]+1      ;add offset for logical line
          PLP
          JSR      BMP      ;put bit in bit map
          DEY
          BMI      ELL2

          CPY      ROWCRS ;cursor row
          BCS      ELL1

ELL2     LDA      ROWCRS ;cursor row
          CLC
          ADC      #8*[LOGMAP-TABMAP] ;add offset for logical line
          PLP
          JMP      BMP      ;put bit in bit map, return

```

```

**      CLN - Clear Line
*
*      ENTRY JSR      CLN
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

CLN      =      *      ;entry
          LDA      LMARGN ;left margin
          STA      COLCRS ;low cursor column
          JSR      CCA      ;convert cursor row/column to address
          SEC
          LDA      RMARGN ;right margin
          SBC      LMARGN ;subtract left margin
          TAY      ;screen width
          LDA      #0

CLN1     STA      (ADRESS),Y
          DEY
          BPL      CLN1      ;if not done

          RTS      ;return

```

```

**      SCR - Scroll
*
*      ENTRY JSR    SCR
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SCR    =    *    ;entry
;      Initialize.
      JSR    BLR    ;rotate logical line bit map left
;      Check for fine scrolling.
      LDA    FINE
      BEQ    SCR5    ;if not fine scrolling
SCR1    LDA    VSFLAG ;vertical scroll count
      BNE    SCR1    ;if prior scroll not yet done
      LDA    #8
      STA    VSFLAG ;vertical scroll count
;      Wait for scroll to complete.
SCR2    LDA    VSFLAG ;vertical scroll count
      CMP    #1      ;start of last scan
      BNE    SCR2    ;if not done waiting
SCR3    LDA    VCOUNT
      CMP    #$40
      BCS    SCR3    ;if not done waiting for safe place
      LDX    #$0D
      LDA    BOTSCR
      CMP    #4
      BNE    SCR4    ;if not split screen
      LDX    #$70
SCR4    CPX    VCOUNT
      BCS    SCR4    ;if not done waiting
;      Exit.
SCR5    JSR    SMS    ;set memory scan counter address
;      JSR    SSD    ;scroll screen for delete, return

```

```

**      SSD - Scroll Screen for Delete
*
*      ENTRY JSR    SSD
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SSD    =    *    ;entry
;      Initialize.

```

```

        LDA    ADDRESS      ;address
        LDX    ADDRESS+1

;      Calculate number of bytes to move.

SSD1   INX
        CPX    RAMTOP
        BEQ    SSD2          ;if at RAMTOP

        SEC
        SBC    #$10
        JMP    SSD1          ;continue

SSD2   ADC    #39             ;(CLC and ADC #40)
        BNE    SSD3          ;if byte count non-zero

        LDX    ADDRESS+1
        INX
        CPX    RAMTOP
        BEQ    SSD6          ;if at RAMTOP

        CLC
        ADC    #$10

;      Adjust address.

SSD3   TAY
        STA    COUNTR        ;number of bytes
        SEC
        LDA    ADDRESS
        SBC    COUNTR        ;subtract
        STA    ADDRESS      ;update low address
        BCS    SSD4          ;if no borrow

        DEC    ADDRESS+1     ;adjust high address

;      Move data down.

SSD4   LDA    ADDRESS
        CLC
        ADC    #40
        STA    COUNTR        ;address + 40
        LDA    ADDRESS+1
        ADC    #0
        STA    COUNTR+1

SSD5   LDA    (COUNTR),Y     ;byte to move
        STA    (ADDRESS),Y  ;move byte
        INY
        BNE    SSD5          ;if not done (256-16 times)

        LDY    #256-240
        LDA    ADDRESS
        CMP    #-40
        BEQ    SSD6          ;if all done

        CLC
        ADC    #240
        STA    ADDRESS      ;update low address
        BCC    SSD4          ;if no carry

        INC    ADDRESS+1     ;adjust high address
        BNE    SSD4          ;continue

;      Clear last line.

SSD6   LDX    RAMTOP

```

```

DEX
STX   COUNTR+1
LDX   #-40
STX   COUNTR
LDA   #0
LDY   #39

SSD7  STA   (COUNTR),Y    ;clear byte of last line
      DEY
      BPL   SSD7          ;if not done

;      JMP   SLC          ;set logical column, return

```

```

**      SLC - Set Logical Column
*
*      ENTRY JSR   SLC
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

SLC   =   *           ;entry

;      Initialize.

      LDA   #0
      STA   LOGCOL ;initialize logical column
      LDA   ROWCRS ;cursor row
      STA   HOLD1 ;working row

;      Search for beginning of line.

SLC1  LDA   HOLD1 ;add in row component
      JSR   BLG1
      BCS   SLC2 ;if beginning of line found

      LDA   LOGCOL ;logical column
      CLC
      ADC   #40    ;add number of characters per line
      STA   LOGCOL ;update logical column
      DEC   HOLD1 ;decrement working row
      JMP   SLC1  ;continue

;      Add in cursor column.

SLC2  CLC
      LDA   LOGCOL ;logical column
      ADC   COLCRS ;add low cursor column
      STA   LOGCOL ;update logical column
      RTS

```

```

**      CBC - Compute Buffer Count
*
*      CBC computes the buffer count as the number of bytes from the
*      buffer start to the end of the logical line (with trailing
*      spaces removed).
*
*      ENTRY JSR      CBC
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CBC      =      *      ;entry

;      Initialize.

      JSR      SRC          ;save row and column
      LDA      LOGCOL       ;logical column
      PHA      PHA          ;save logical column
      LDA      BUFSTR       ;start of buffer
      STA      ROWCRS       ;cursor row
      LDA      BUFSTR+1
      STA      COLCRS       ;low cursor column
      LDA      #1
      STA      BUFCNT       ;initialize buffer count

;      Determine last line on screen.

CBC1     LDX      #23        ;normal last line on screen
      LDA      SWPFLG
      BPL      CBC2         ;if not swapped

      LDX      #3          ;last line on screen

;      Check for cursor on last line of screen.

CBC2     CPX      ROWCRS     ;cursor row
      BNE      CBC3         ;if cursor on last line

      LDA      COLCRS       ;low cursor column
      CMP      RMARGN       ;right margin
      BNE      CBC3         ;if not at right margin

      INC      BUFCNT       ;fake SEA to avoid scrolling
      JMP      CBC4

CBC3     JSR      SZA        ;set zero data and advance cursor
      INC      BUFCNT
      LDA      LOGCOL       ;logical column
      CMP      LMARGN       ;left margin
      BNE      CBC1         ;if not yet at left margin

      DEC      ROWCRS       ;decrement cursor row
      JSR      CLF          ;move cursor left

CBC4     JSR      GDC        ;get data under cursor
      BNE      CBC6         ;if non-zero, quit

      DEC      BUFCNT       ;DECREMENT COUNTER
      LDA      LOGCOL       ;logical column
      CMP      LMARGN       ;left margin
      BEQ      CBC6         ;if beginning of logical line, exit

      JSR      CLF          ;move cursor left

```

```

LDA COLCRS ;low cursor column
CMP RMARGN ;right margin
BNE CBC5 ;if cursor column not right margin

DEC ROWCRS ;decrement cursor row

CBC5 LDA BUFCNT
BNE CBC4 ;if BUFCNT non-zero, continue

CBC6 PLA ;saved logical column
STA LOGCOL ;restore logical column
JMP RRC ;restore row and column, return

```

```

** SBS - Set Buffer Start and Logical Column
*
* ENTRY JSR SBS
*
* MODS
* Original Author Unknown
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01

```

```

SBS = * ;entry
JSR SLC ;set logical column
LDA HOLD1
STA BUFSTR
LDA LMARGN ;left margin
STA BUFSTR+1

SBS1 RTS ;return

```

```

** DQQ - Delete Line
*
* ENTRY JSR DQQ
*
* MODS
* Original Author Unknown
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01

```

```

DQQ = * ;entry
LDA LOGCOL ;logical column
CMP LMARGN ;left margin
BNE DQQ1 ;if not at left margin

DEC ROWCRS ;decrement cursor row

DQQ1 JSR SLC ;set logical column
JMP DWQ

```

```

** DWQ - Delete Line
*
* ENTRY JSR DWQ
*
* MODS
* Original Author Unknown
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01

```

```

DWQ = * ;entry
; Check for left margin.

```

```

LDA LOGCOL ;logical column
CMP LMARGN ;left margin
BEQ SBS1 ;if at left margin, return

JSR CCA ;convert cursor row/column to address
LDA RMARGN ;right margin
SEC
SBC LMARGN ;subtract left margin
TAY ;offset to last byte

DWQ1 LDA (ADRESS),Y
BNE SBS1

DEY
BPL DWQ1 ;if not done

JMP DLN1 ;delete line, return

```

```

**      CCC - Check for Control Character
*
*      ENTRY JSR    CCC
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

CCC      =      *      ;entry

LDX      #TCCRL-3      ;offset to last entry

CCC1     LDA      TCCR,X      ;control character
CMP      ATACHR      ;ATASCII character
BEQ      CCC2      ;if character found, exit

DEX
DEX
DEX
BPL      CCC1      ;if not done, continue search

CCC2     RTS      ;return

```

```

**      SRC - Save Row and Column
*
*      ENTRY JSR    SRC
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

SRC      =      *      ;entry
LDX      #2      ;offset to last byte

SRC1     LDA      ROWCRS,X      ;byte of cursor row/column
STA      TMPROW,X      ;save byte of cursor row/column
DEX
BPL      SRC1      ;if not done

RTS      ;return

```

```

**      RRC - Restore Row and Column
*
*      ENTRY JSR    RRC
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

RRC      =      *          ;entry
          LDX      #2          ;offset to last byte

RRC1     LDA      TMPROW,X      ;byte of saved cursor row/column
          STA      ROWCRS,X      ;byte of cursor row/column
          DEX
          BPL      RRC1          ;if not done

          RTS                    ;return

```

```

**      SWA - Swap Cursor Position with Regular Cursor Position
*
*      ENTRY JSR    SWA
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SWA      =      *          ;entry
;      Check for split screen.

          LDA      BOTSCR        ;screen bottom
          CMP      #24          ;normal indicator
          BEQ      SWA2          ;if not split screen

;      Swap cursor parameters.

          LDX      #11          ;offset to last byte

SWA1     LDA      ROWCRS,X      ;destination cursor parameter
          PHA                    ;save cursor parameter
          LDA      TXTR0W,X      ;source cursor parameter
          STA      ROWCRS,X      ;update destination cursor parameter
          PLA                    ;saved cursor parameter
          STA      TXTR0W,X      ;update source cursor parameter
          DEX
          BPL      SWA1          ;if not done

;      Complement swap flag.

          LDA      SWPFLG        ;swap flag
          EOR      #$FF          ;complement swap flag
          STA      SWPFLG        ;update swap flag

;      Exit.

SWA2     JMP      SST          ;perform screen STATUS, return

```



```

**      SKC - Sound Key Click
*
*      ENTRY JSR      SKC
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SKC      =      *      ;entry

;      Initialize.

      LDX      #2*63 ;2 times trip count
      PHA                      ;save A

;      Turn loudspeaker on.

SKC1     STX      CONSOL ;turn loudspeaker on

;      Wait for VBLANK (loudspeaker off).

      LDA      VCOUNT ;vertical line counter

SKC2     CMP      VCOUNT ;current vertical line counter
      BEQ      SKC2 ;if vertical line not changed

;      Decrement and check trip count.

      DEX
      DEX
      BPL      SKC1 ;if not done

;      Exit.

      PLA                      ;restore A
      RTS                      ;return

```

```

**      SCL - Set Cursor at Left Edge
*
*      ENTRY JSR      SCL
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SCL      =      *      ;entry

      LDA      #0      ;assume 0

      LDX      SWPFLG ;swap flag
      BNE      SCL1 ;if not swapped

      LDX      DINDEK ;mode
      BNE      SCL2 ;if not mode 0

SCL1     LDA      LMARGN ;use left margin instead of 0

SCL2     STA      COLCRS ;set low cursor column
      RTS                      ;return

```

```

**      SMS - Set Memory Scan Counter Address
*
*      ENTRY JSR      SMS
*
*      MODS
*
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

SMS      =      *      ;entry
          LDA      SAVMSC      ;saved low memory scan counter
          STA      ADDRESS      ;set low address
          LDA      SAVMSC+1      ;saved high memory scan counter
          STA      ADDRESS+1      ;set high address
          RTS      ;return

```

```

**      SSP - Perform Screen SPECIAL
*
*      SSP draws a line from OLDROW/OLDCOL to NEWROW/NEWCOL.
*
*      ENTRY JSR      SSP
*
*      MODS
*
*      A. Miller
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

SSP      =      *      ;entry
;      Determine command.

          LDX      #0      ;assume no fill
          LDA      ICCOMZ      ;command
          CMP      #$11      ;DRAW command
          BEQ      SSP2      ;if DRAW command

          CMP      #$12      ;FILL command
          BEQ      SSP1      ;if FILL command

          LDY      #NVALID      ;invalid command error
          RTS      ;return

SSP1     INX      ;indicate fill

SSP2     STX      FILFLG      ;right fill flag

;      Set destination row/column.

          LDA      ROWCRS      ;cursor row
          STA      NEWROW
          LDA      COLCRS      ;cursor column
          STA      NEWCOL
          LDA      COLCRS+1
          STA      NEWCOL+1

;      Compute row increment and difference.

          LDA      #1      ;assume increment +1
          STA      ROWINC      ;row increment
          STA      COLINC      ;column increment
          SEC
          LDA      NEWROW      ;destination row
          SBC      OLDROW      ;subtract source row
          STA      DELTA      ;row difference

```

```

    BCS    SSP3            ;if difference positive
;    Set row increment to -1 and complement row difference.

    LDA    #$FF            ;increment -1
    STA    ROWINC          ;update row increment
    LDA    DELTAR          ;row difference
    EOR    #$FF
    CLC
    ADC    #1              ;add 1 for 2's complement
    STA    DELTAR          ;update row difference

;    Compute column increment and difference.
SSP3    SEC
    LDA    NEWCOL          ;destination column
    SBC    OLDCOL          ;source column
    STA    DELTAC          ;column difference
    LDA    NEWCOL+1
    SBC    OLDCOL+1
    STA    DELTAC+1
    BCS    SSP4            ;if difference positive
;    Set column increment to -1 and complement column difference.

    LDA    #$FF            ;increment -1
    STA    COLINC          ;update column increment
    LDA    DELTAC          ;column difference
    EOR    #$FF            ;absolute value of column difference
    STA    DELTAC          ;update column difference
    LDA    DELTAC+1
    EOR    #$FF
    STA    DELTAC+1
    INC    DELTAC          ;add 1 for 2's complement
    BNE    SSP4            ;if no carry

    INC    DELTAC+1        ;adjust for 2's complement
;    Set up working row/column and cursor row/column.
SSP4    LDX    #2            ;offset to last byte
    LDY    #0
    STY    COLAC+1        ;zero high working column

SSP5    TYA
    STA    ROWAC,X         ;zero byte of working row/column
    LDA    OLDROW,X        ;byte of source row/column
    STA    ROWCRS,X        ;byte of cursor row/column
    DEX
    BPL    SSP5            ;if not done
;    Determine difference.

    LDA    DELTAC          ;low column difference
    INX
    TAY
    LDA    DELTAC+1        ;high column difference
    STA    COUNTR+1        ;initialize high iteration counter
    STA    ENDPT+1         ;initialize high end point
    BNE    SSP6            ;if high column difference > 0

    LDA    DELTAC          ;low column difference
    CMP    DELTAR          ;row difference
    BCS    SSP6            ;if column difference > row difference

    LDA    DELTAR          ;row difference
    LDX    #2              ;offset to working column

```

```

TAY                                ;row difference

SSP6 TYA                            ;low maximum difference
STA  COUNTR                        ;low iteration counter
STA  ENDPT                        ;low end point
PHA                                ;save low end point
LDA  ENDPT+1                      ;high end point
LSR  A                            ;C = LSB of high end point
PLA                                ;saved low end point
ROR  A
STA  ROWAC,X                      ;low working row or column

;    Check for iteration counter zero.

SSP7 LDA  COUNTR                  ;low iteration counter
ORA  COUNTR+1                    ;or in high iteration counter
BNE  SSP8                        ;if iteration counter is not zero

JMP  SSP19                      ;exit

;    Update working row.

SSP8 CLC
LDA  ROWAC                      ;working row
ADC  DELTAR                     ;row difference
STA  ROWAC                      ;update working row
BCC  SSP9                      ;if no carry

INC  ROWAC+1                    ;adjust high working row

SSP9 LDA  ROWAC+1                ;high working row
CMP  ENDPT+1                    ;high end point
BCC  SSP11                      ;if high working row < high end point

BNE  SSP10                      ;if high working row > high end point

LDA  ROWAC                      ;low working row
CMP  ENDPT                      ;low end point
BCC  SSP11                      ;if low working row < low end point

SSP10 CLC
LDA  ROWCRS                     ;cursor row
ADC  ROWINC                     ;add row increment
STA  ROWCRS                     ;update cursor row
LDX  #0                         ;indicate subtract from working row
JSR  SEP                        ;subtract end pointer

SSP11 CLC
LDA  COLAC                      ;low working column
ADC  DELTAC                     ;add column difference
STA  COLAC                      ;update working column
LDA  COLAC+1
ADC  DELTAC+1
STA  COLAC+1
CMP  ENDPT+1                    ;high end point
BCC  SSP15                      ;if high working column < high end point

BNE  SSP12                      ;if high working column > high end point

LDA  COLAC                      ;low working column
CMP  ENDPT                      ;low end point
BCC  SSP15                      ;if low working column < low end point

SSP12 BIT  COLINC                ;column increment
BPL  SSP13                      ;if column increment positive

DEC  COLCRS                     ;decrement low cursor column

```

```

        LDA    COLCRS        ;low cursor column
        CMP    #$FF
        BNE    SSP14

        LDA    COLCRS+1      ;high cursor column
        BEQ    SSP14          ;if zero, do not decrement

        DEC    COLCRS+1      ;decrement high cursor column
        BPL    SSP14

SSP13   INC    COLCRS        ;increment low cursor column
        BNE    SSP14          ;if no carry

        INC    COLCRS+1      ;adjust high cursor column

SSP14   LDX    #2             ;indicate subtract from working column
        JSR    SEP           ;subtract end pointer

;       Plot point.

SSP15   JSR    CCR           ;check cursor range
        JSR    PLO           ;plot point

;       Check for right fill.

        LDA    FILFLG        ;right fill flag
        BEQ    SSP18          ;if no right fill

;       Process right fill.

        JSR    SRC           ;save row and column
        LDA    ATACHR         ;plot point
        STA    HOLD4         ;save plot point

SSP16   LDA    ROWCRS        ;cursor row
        PHA                    ;save cursor row
        JSR    ACC           ;advance cursor column
        PLA                    ;saved cursor row
        STA    ROWCRS        ;restore cursor row
        JSR    CCR           ;check cursor range
        JSR    GDC           ;get data under cursor
        BNE    SSP17          ;if non-zero data encountered

        LDA    FILDAT        ;fill data
        STA    ATACHR         ;plot point
        JSR    PLO           ;plot point
        JMP    SSP16          ;continue

SSP17   LDA    HOLD4         ;saved plot point
        STA    ATACHR         ;restore plot point
        JSR    RRC           ;restore row and column

;       Subtract 1 from iteration counter.

SSP18   SEC
        LDA    COUNTR        ;iteration counter
        SBC    #1             ;subtract 1
        STA    COUNTR        ;update iteration counter
        LDA    COUNTR+1
        SBC    #0
        STA    COUNTR+1

;       Check for completion.

        BMI    SSP19          ;if iteration counter negative, exit

        JMP    SSP7           ;continue

```

```
;      Exit.
SSP19  JMP      SST      ;perform screen STATUS, return
```

**	TMSK - Table of Bit Masks
----	---------------------------

```
TMSK  DB      $00      ;0 - mask for no bits
      DB      $01      ;1 - mask for lower 1 bit
      DB      $03      ;2 - mask for lower 2 bits
      DB      $07      ;3 - mask for lower 3 bits
```

**	TDSC - Table of Default Screen Colors
----	---------------------------------------

```
TDSC  DB      $28      ;default playfield 0 color
      DB      $CA      ;default playfield 1 color
      DB      $94      ;default playfield 2 color
      DB      $46      ;default playfield 3 color
      DB      $00      ;default background color
```

**	TCCR - Table of Control Character Routines
*	
*	Each entry is 3 bytes. The first byte is the control
*	character; the second and third bytes are the address of
*	the routine which processes the control character.

```
TCCR  DB      $1B
      DW      ESC      ;escape

      DB      $1C
      DW      CUP      ;move cursor up

      DB      $1D
      DW      CDN      ;move cursor down

      DB      $1E
      DW      CLF      ;move cursor left

      DB      $1F
      DW      CRT      ;move cursor right

      DB      $7D
      DW      CSC      ;clear screen

      DB      $7E
      DW      BSP      ;backspace

      DB      $7F
      DW      TAB      ;tab

      DB      $9B
      DW      RWS      ;return with scrolling

      DB      $9C
      DW      DLN      ;delete line

      DB      $9D
      DW      ILN      ;insert line

      DB      $9E
      DW      CTB      ;clear tab

      DB      $9F
```

```

        DW      STB      ;set tab

        DB      $FD
        DW      BEL      ;sound bell

        DB      $FE
        DW      DCH      ;delete character

        DB      $FF
        DW      ICH      ;insert character

TCCRL =      *-TCCR ;length

```

```

**      TSFR - Table of Super Function (Shifted Function Key) Routines
*
*      Each entry is 3 bytes. The first byte is the super function
*      character; the second and third bytes are the address of the
*      routine which processes the super function.

```

```

TSFR    DB      $1C
        DW      CHM      ;move cursor home

        DB      $1D
        DW      CBT      ;move cursor to bottom

        DB      $1E
        DW      CLM      ;move cursor to left margin

        DB      $1F
        DW      CRM      ;move cursor to right margin

```

```

**      TAIC - Table of ATASCII to Internal Conversion Constants

```

```

TAIC    DB      $40      ;0
        DB      $00      ;1
        DB      $20      ;2
        DB      $60      ;3

```

```

**      TIAC - Table of Internal to ATASCII Conversion Constants

```

```

TIAC    DB      $20      ;0
        DB      $40      ;1
        DB      $00      ;2
        DB      $60      ;3

```

```

**      TCKD - Table of Character Key Definitions
*
*      Entry n is the ATASCII equivalent of key code n.

```

```

TCKD    =      *

;      Lower Case Characters

        DB      $6C      ;$00 - l
        DB      $6A      ;$01 - j
        DB      $3B      ;$02 - semicolon
        DB      $8A      ;$03 - F1
        DB      $8B      ;$04 - F2
        DB      $6B      ;$05 - k
        DB      $2B      ;$06 - +
        DB      $2A      ;$07 - *
        DB      $6F      ;$08 - o

```

DB	\$80	;\$09 - (invalid)
DB	\$70	;\$0A - p
DB	\$75	;\$0B - u
DB	\$9B	;\$0C - return
DB	\$69	;\$0D - i
DB	\$2D	;\$0E - -
DB	\$3D	;\$0F - =

DB	\$76	;\$10 - v
DB	\$80	;\$11 - (invalid)
DB	\$63	;\$12 - c
DB	\$8C	;\$13 - F3
DB	\$8D	;\$14 - F4
DB	\$62	;\$15 - b
DB	\$78	;\$16 - x
DB	\$7A	;\$17 - z
DB	\$34	;\$18 - 4
DB	\$80	;\$19 - (invalid)
DB	\$33	;\$1A - 3
DB	\$36	;\$1B - 6
DB	\$1B	;\$1C - escape
DB	\$35	;\$1D - 5
DB	\$32	;\$1E - 2
DB	\$31	;\$1F - 1

DB	\$2C	;\$20 - comma
DB	\$20	;\$21 - space
DB	\$2E	;\$22 - period
DB	\$6E	;\$23 - n
DB	\$80	;\$24 - (invalid)
DB	\$6D	;\$25 - m
DB	\$2F	;\$26 - /
DB	\$81	;\$27 - inverse
DB	\$72	;\$28 - r
DB	\$80	;\$29 - (invalid)
DB	\$65	;\$2A - e
DB	\$79	;\$2B - y
DB	\$7F	;\$2C - tab
DB	\$74	;\$2D - t
DB	\$77	;\$2E - w
DB	\$71	;\$2F - q

DB	\$39	;\$30 - 9
DB	\$80	;\$31 - (invalid)
DB	\$30	;\$32 - 0
DB	\$37	;\$33 - 7
DB	\$7E	;\$34 - backspace
DB	\$38	;\$35 - 8
DB	\$3C	;\$36 - <
DB	\$3E	;\$37 - >
DB	\$66	;\$38 - f
DB	\$68	;\$39 - h
DB	\$64	;\$3A - d
DB	\$80	;\$3B - (invalid)
DB	\$82	;\$3C - CAPS
DB	\$67	;\$3D - g
DB	\$73	;\$3E - s
DB	\$61	;\$3F - a

;
Upper Case Characters

DB	\$4C	;\$40 - L
DB	\$4A	;\$41 - J
DB	\$3A	;\$42 - colon
DB	\$8A	;\$43 - SHIFT-F1
DB	\$8B	;\$44 - SHIFT-F2
DB	\$4B	;\$45 - K

DB	\$5C	;\$46 - \
DB	\$5E	;\$47 - ^
DB	\$4F	;\$48 - 0
DB	\$80	;\$49 - (invalid)
DB	\$50	;\$4A - P
DB	\$55	;\$4B - U
DB	\$9B	;\$4C - SHIFT-return
DB	\$49	;\$4D - I
DB	\$5F	;\$4E - _
DB	\$7C	;\$4F -
DB	\$56	;\$50 - V
DB	\$80	;\$51 - (invalid)
DB	\$43	;\$52 - C
DB	\$8C	;\$53 - SHIFT-F3
DB	\$8D	;\$54 - SHIFT-F4
DB	\$42	;\$55 - B
DB	\$58	;\$56 - X
DB	\$5A	;\$57 - Z
DB	\$24	;\$58 - \$
DB	\$80	;\$59 - (invalid)
DB	\$23	;\$5A - #
DB	\$26	;\$5B - &
DB	\$1B	;\$5C - SHIFT-escape
DB	\$25	;\$5D - %
DB	\$22	;\$5E - "
DB	\$21	;\$5F - !
DB	\$5B	;\$60 - [
DB	\$20	;\$61 - SHIFT-space
DB	\$5D	;\$62 -]
DB	\$4E	;\$63 - N
DB	\$80	;\$64 - (invalid)
DB	\$4D	;\$65 - M
DB	\$3F	;\$66 - ?
DB	\$81	;\$67 - SHIFT-inverse
DB	\$52	;\$68 - R
DB	\$80	;\$69 - (invalid)
DB	\$45	;\$6A - E
DB	\$59	;\$6B - Y
DB	\$9F	;\$6C - SHIFT-tab
DB	\$54	;\$6D - T
DB	\$57	;\$6E - W
DB	\$51	;\$6F - Q
DB	\$28	;\$70 - (
DB	\$80	;\$71 - (invalid)
DB	\$29	;\$72 -)
DB	\$27	;\$73 - '
DB	\$9C	;\$74 - SHIFT-delete
DB	\$40	;\$75 - @
DB	\$7D	;\$76 - SHIFT-clear
DB	\$9D	;\$77 - SHIFT-insert
DB	\$46	;\$78 - F
DB	\$48	;\$79 - H
DB	\$44	;\$7A - D
DB	\$80	;\$7B - (invalid)
DB	\$83	;\$7C - SHIFT-CAPS
DB	\$47	;\$7D - G
DB	\$53	;\$7E - S
DB	\$41	;\$7F - A

;
Control Characters

DB	\$0C	;\$80 - CTRL-L
DB	\$0A	;\$81 - CTRL-J
DB	\$7B	;\$82 - CTRL-semicolon

DB	\$80	;\$83 - (invalid)
DB	\$80	;\$84 - (invalid)
DB	\$0B	;\$85 - CTRL-K
DB	\$1E	;\$86 - CTRL-left arrow
DB	\$1F	;\$87 - CTRL-right arrow
DB	\$0F	;\$88 - CTRL-O
DB	\$80	;\$89 - (invalid)
DB	\$10	;\$8A - CTRL-P
DB	\$15	;\$8B - CTRL-U
DB	\$9B	;\$8C - CTRL-return
DB	\$09	;\$8D - CTRL-I
DB	\$1C	;\$8E - CTRL-up arrow
DB	\$1D	;\$8F - CTRL-down arrow
DB	\$16	;\$90 - CTRL-V
DB	\$80	;\$91 - (invalid)
DB	\$03	;\$92 - CTRL-C
DB	\$89	;\$93 - CTRL-F3
DB	\$80	;\$94 - (invalid)
DB	\$02	;\$95 - CTRL-B
DB	\$18	;\$96 - CTRL-X
DB	\$1A	;\$97 - CTRL-Z
DB	\$80	;\$98 - (invalid)
DB	\$80	;\$99 - (invalid)
DB	\$85	;\$9A - CTRL-3
DB	\$80	;\$9B - (invalid)
DB	\$1B	;\$9C - CTRL-escape
DB	\$80	;\$9D - (invalid)
DB	\$FD	;\$9E - CTRL-2
DB	\$80	;\$9F - (invalid)
DB	\$00	;\$A0 - CTRL-comma
DB	\$20	;\$A1 - CTRL-space
DB	\$60	;\$A2 - CTRL-period
DB	\$0E	;\$A3 - CTRL-N
DB	\$80	;\$A4 - (invalid)
DB	\$0D	;\$A5 - CTRL-M
DB	\$80	;\$A6 - (invalid)
DB	\$81	;\$A7 - CTRL-inverse
DB	\$12	;\$A8 - CTRL-R
DB	\$80	;\$A9 - (invalid)
DB	\$05	;\$AA - CTRL-E
DB	\$19	;\$AB - CTRL-Y
DB	\$9E	;\$AC - CTRL-tab
DB	\$14	;\$AD - CTRL-T
DB	\$17	;\$AE - CTRL-W
DB	\$11	;\$AF - CTRL-Q
DB	\$80	;\$B0 - (invalid)
DB	\$80	;\$B1 - (invalid)
DB	\$80	;\$B2 - (invalid)
DB	\$80	;\$B3 - (invalid)
DB	\$FE	;\$B4 - CTRL-delete
DB	\$80	;\$B5 - (invalid)
DB	\$7D	;\$B6 - CTRL-clear
DB	\$FF	;\$B7 - CTRL-insert
DB	\$06	;\$B8 - CTRL-F
DB	\$08	;\$B9 - CTRL-H
DB	\$04	;\$BA - CTRL-D
DB	\$80	;\$BB - (invalid)
DB	\$84	;\$BC - CTRL-CAPS
DB	\$07	;\$BD - CTRL-G
DB	\$13	;\$BE - CTRL-S
DB	\$01	;\$BF - CTRL-A

```

**      TFKD - Table of Function Key Definitions
*
*      Entry n is the ATASCII equivalent of adjusted function key
*      code n.

```

```

TFKD  DB    $1C    ;0 - F1 key
      DB    $1D    ;1 - F2 key
      DB    $1E    ;2 - F3 key
      DB    $1F    ;3 - F4 key

      DB    $8E    ;4 - SHIFT-F1 key
      DB    $8F    ;5 - SHIFT-F2 key
      DB    $90    ;6 - SHIFT-F3 key
      DB    $91    ;7 - SHIFT-F4 key

```

```

**      KIR - Process Keyboard IRQ
*
*      ENTRY JMP    KIR
*
*      EXIT
*      Exits via RTI
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

KIR    =    *      ;entry

;      Initialize.

      TXA
      PHA          ;save X
      TYA
      PHA          ;save Y
      LDY    PORTB ;port B memory control
      LDA    KBCODE ;keyboard code
      CMP    CH1   ;last key code
      BNE    KIR1  ;if not last key code

      LDX    KEYDEL ;keyboard debounce delay
      BNE    KIR8  ;if delay not expired, treat as bounce

;      Check for CTRL-F1.

KIR1   LDX    KEYDIS ;save keyboard disable flag
      CMP    #CNTLF1
      BNE    KIR4   ;if not CTRL-F1

;      Process CTRL-F1.

      TXA          ;keyboard disable flag
      EOR    #$FF  ;complement keyboard disable flag
      STA    KEYDIS ;update keyboard disable flag
      BNE    KIR2  ;if keyboard disabled

      TYA          ;port B memory control
      ORA    #$04  ;turn off LED 1
      BNE    KIR3  ;update port B memory control

KIR2   TYA          ;port B memory control
      AND    #$FB  ;turn on LED 1

KIR3   TAY          ;updated port B memory control

```

```

        BCS    KIR7    ;reset keyboard controls

;      Check keyboard disable.

KIR4    TXA          ;keyboard disable flag
        BNE    KIR9    ;if keyboard disabled, exit

;      Get character.

        LDA     KBCODE ;keyboard code
        TAX          ;character

;      Check for CTRL-1.

        CMP     #CNTL1
        BNE    KIR5    ;if not CTRL-1

;      Process CTRL-1.

        LDA     SSFLAG ;start/stop flag
        EOR     #$FF   ;complement start/stop flag
        STA     SSFLAG ;update start/stop flag
        BCS    KIR7    ;make CTRL-1 invisible

;      Check character.

KIR5    AND     #$3F   ;mask off shift and control bits
        CMP     #HELP
        BNE    KIR10  ;if not HELP key

;      Process HELP.

        STX     HELPPG ;indicate HELP key pressed
        BEQ     KIR7    ;reset keyboard controls

;      Process character.

KIR6    STX     CH      ;key code
        STX     CH1    ;reset previous key code

;      Reset keyboard controls.

KIR7    LDA     #3
        STA     KEYDEL ;re-initialize for debounce
        LDA     #0
        STA     ATTRACT ;clear attract-mode timer/flag

;      Prepare to exit.

KIR8    LDA     KRPDEL ;auto-repeat delay
        STA     SRTIMR ;reset software key repeat timer
        LDA     SDMCTL ;DMA control
        BNE    KIR9    ;if DMA not disabled, exit

        LDA     DMASAV ;saved DMA control
        STA     SDMCTL ;DMA control

;      Exit.

KIR9    STY     PORTB  ;update port B memory control
        PLA          ;saved Y
        TAY          ;restore Y
        PLA          ;saved X
        TAX          ;restore X
        PLA          ;restore A
        RTI          ;return

```

```

;      Check for CTRL-F2 or CTRL-F4.
KIR10 CPX    #CNTLF2
      BEQ    KIR12 ;if CTRL-F2

      CPX    #CNTLF4
      BNE    KIR6  ;if not CTRL-F4

;      Process CTRL-F4.

      LDA    CHBAS ;character set base
      LDX    CHSALT ;character set alternate
      STA    CHSALT ;update character set alternate
      STX    CHBAS ;update character set base

      CPX    #high ICSORG ;high international character set origin
      BEQ    KIR11        ;if international character set

      TYA
      ORA    #$08 ;turn off LED 2
      TAY    ;updated port B memory control
      BNE    KIR7 ;reset keyboard controls

KIR11 TYA    ;port B memory control
      AND    #$F7 ;turn on LED 2
      TAY    ;updated port B memory control
      JMP    KIR7 ;reset keyboard controls

;      Process CTRL-F2.
KIR12 LDA    SDMCTL ;DMA control
      BEQ    KIR9  ;if disabled, exit

      STA    DMASAV ;save DMA state
      LDA    #0     ;disable DMA
      STA    SDMCTL ;DMA control
      BEQ    KIR9  ;exit

```

```

**      FDL - Process Display List Interrupt for Fine Scrolling
*
*      ENTRY  JMP      FDL
*
*      EXIT
*          Exits via RTI
*
*      NOTES
*          Problem: in the CRASS65 version, COLRSH was not declared
*          zero-page.
*          Problem: in the CRASS65 version, DRKMSK was not declared
*          zero-page.
*
*      MODS
*          H. Stewart      1982-06-01
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

FDL      =      *          ;entry
          PHA          ;save A
          LDA      COLOR2 ;playfield 2 color
          ; EOR      COLRSH ;modify with attract-mode color shift
          VFD      8\4D,8\low COLRSH,8\high COLRSH
          ; AND      DRKMSK ;modify with attract-mode luminance
          VFD      8\2D,8\low DRKMSK,8\high COLRSH
          STA      WSYNC  ;wait for HBLANK synchronization
          STA      COLPF1 ;playfield 1 color/luminance
          PLA          ;restore A
          RTI          ;return

```

\$FCD8 Patch

FIX \$FCD8

**	FCD8 - \$FCD8 Patch
*	
*	For compatibility with OS Revision B, sound key click.

JMP SKC ;sound key click, return

Cassette Handler

```
**      CIN - Initialize Cassette
*
*      ENTRY JSR    CIN
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
CIN      =      *      ;entry
          LDA      #low B00600 ;indicate 600 baud
          STA      CBAUDL      ;cassette baud rate
          LDA      #high B00600
          STA      CBAUDH
;      IRP      CSP      ;return
```

```
**      CSP - Perform Cassette SPECIAL
*
*      CSP does nothing.
*
*      ENTRY JSR    CSP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
CSP      =      *      ;entry
          RTS          ;return
```

```
**      COP - Perform Cassette OPEN
*
*      ENTRY JSR    COP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
COP      =      *      ;entry
;      Set cassette IRG type.

          LDA      ICAX2Z ;second auxiliary information
          STA      FTYPE ;cassette IRG type
;      Check OPEN mode.

          LDA      ICAX1Z ;OPEN mode
          AND      #$0C ;open for input and output bits
          CMP      #$04 ;open for input bit
          BEQ      OCI ;if open for input, process, return

          CMP      #$08 ;open for output bit
          BEQ      OCO ;if open for output, process, return
;      Exit.

          RTS          ;return
```



```

**      OCI - Open Cassette for Input
*
*      ENTRY JSR    OCI
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

OCI      =      *      ;entry

;      Process open for input.

      LDA      #0      ;indicate reading
      STA      WMODE    ;WRITE mode
      STA      EOF      ;indicate no EOF yet
      LDA      #TONE2   ;tone for pressing PLAY
      JSR      AUB      ;alert user with beep
      BMI      PBC1     ;if error

;      Initiate cassette READ.

;      JMP      ICR      ;initiate cassette READ, return

```

```

**      ICR - Initiate Cassette READ
*
*      ENTRY JSR    ICR
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

ICR      =      *      ;entry

;      Initialize.

      LDA      #MOTRGO   ;motor on
      STA      PACTL     ;port A control

;      Wait for leader read.

      LDX      PALNTS
      LDY      RLEADL,X  ;low READ leader
      LDA      RLEADH,X  ;high READ leader
      TAX
      LDA      #3
      STA      CDTMF3
      JSR      SETVBV    ;set up VBLANK timer

ICR1     LDA      CDTMF3
      BNE      ICR1      ;if not done waiting

;      Initialize.

      LDA      #128     ;buffer size
      STA      BPTR     ;initialize buffer pointer
      STA      BLIM     ;initialize buffer limit
      JMP      OC02     ;exit

```

```

**      PBC - Process BREAK for Cassette Operation
*
*      ENTRY JSR    PBC
*
*      MODS
*
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PBC      =      *      ;entry
          LDY      #BRKABT      ;BREAK abort error
          DEC      BRKKEY      ;reset BREAK key flag

```

```

PBC1     LDA      #0      ;indicate reading
          STA      WMODE      ;WRITE mode
          RTS

```

```

**      OC0 - Open Cassette for Output
*
*      ENTRY JSR    OC0
*
*      MODS
*
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

OC0      =      *      ;entry
;      Initialize.

          LDA      #$80      ;indicate writing
          STA      WMODE      ;WRITE mode
          LDA      #TONE1
          JSR      AUB      ;alert user with beep
          BMI      PBC1      ;if error

;      Set baud rate to 600.

          LDA      #low B00600      ;600 baud
          STA      AUDF3
          LDA      #high B00600
          STA      AUDF4

;      Write marks.

          LDA      #$60
          STA      DDEVIC
          JSR      SENDEV
          LDA      #MOTRGO      ;write 5 second blank tape
          STA      PACTL

;      Wait for leader written.

          LDX      PALNTS
          LDY      WLEADL,X
          LDA      WLEADH,X
          TAX
          LDA      #3
          JSR      SETVBV      ;set VBLANK parameters
          LDA      #$FF
          STA      CDTMF3

OC01     LDA      BRKKEY      ;BREAK key flag
          BEQ      PBC      ;if BREAK during write leader, process BREAK

```

```

    LDA    CDTMF3
    BNE    OC01    ;if not done waiting

;    Initialize buffer pointer.

    LDA    #0
    STA    BPTR    ;buffer pointer

;    Indicate success.

OC02    LDY    #SUCCES    ;indicate success
        RTS        ;return

```

```

**      CGB - Perform Cassette GET-BYTE
*
*      ENTRY JSR    CGB
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

CGB    =    *        ;entry

;    Check for EOF.

    LDA    EOF      ;EOF flag
    BMI    RCB3     ;if at EOF already

;    Check for end of buffer.

    LDX    BPTR      ;buffer pointer
    CPX    BLIM      ;buffer limit
    BEQ    RCB       ;if end of buffer, read block, return

;    Get next byte.

    LDA    CASBUF+3,X ;byte
    INC    BPTR      ;increment pointer
    LDY    #SUCCES   ;indicate success

CGB1    RTS        ;return

```

```

**      RCB - Read Cassette Block
*
*      ENTRY JSR    RCB
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01

```

```

RCB    =    *        ;entry

;    Perform READ.

    LDA    #'R'      ;read
    JSR    SCB        ;perform SIO on cassette buffer
    TYA
    BMI    CGB1      ;if SIO error

    LDA    #0
    STA    BPTR      ;reset pointer

```

```

        LDX    #$80    ;default number of bytes
;      Check for header.

        LDA    CASBUF+2
        CMP    #EOT
        BEQ    RCB2    ;if header, read again
;      Check for last record.

        CMP    #DT1
        BNE    RCB1      ;if not last data record

        LDX    CASBUF+130 ;number of bytes
;      Set number of bytes.

RCB1    STX     BLIM
;      Perform cassette GET-BYTE.

        JMP    CGB          ;perform cassette GET-BYTE, return
;      Set EOF flag.

RCB2    DEC     EOF          ;set EOF flag
;      Exit.

RCB3    LDY     #EOFERR      ;end of file indicator
        RTS          ;return

```

```

**      CPB - Perform Cassette PUT-BYTE
*
*      ENTRY JSR    CPB
*
*      MODS
*
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CPB    =      *          ;entry
;      Move data to buffer.

        LDX    BPTR      ;buffer pointer
        STA    CASBUF+3,X ;data
        INC    BPTR      ;increment buffer pointer
        LDY    #SUCCES   ;assume success
;      Check buffer full.

        CPX    #127      ;offset to last byte of buffer
        BEQ    CPB1      ;if buffer full

        RTS          ;return
;      Write cassette buffer.

CPB1    LDA     #DTA      ;indicate data record type
        JSR     WCB       ;write cassette buffer
        LDA     #0
        STA     BPTR     ;reset buffer pointer
        RTS          ;return

```

```

**      CST - Perform Cassette STATUS
*
*      ENTRY JSR      CST
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CST      =      *      ;entry
          LDY      #SUCCES      ;indicate success
          RTS      ;return

```

```

**      CCL - Perform Cassette CLOSE
*
*      ENTRY JSR      CCL
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CCL      =      *      ;entry

;      Check mode.

          LDA      WMODE      ;WRITE mode
          BMI      CCL2      ;if writing

;      Process reading.

          LDY      #SUCCES      ;indicate success

;      Exit.

CCL1     LDA      #MOTRST
          STA      PACTL      ;stop motor
          RTS      ;return

;      Process writing.

CCL2     LDX      BPTR      ;buffer pointer
          BEQ      CCL3      ;if no data bytes in buffer

          STX      CASBUF+130 ;number of bytes
          LDA      #DT1      ;indicate data record type
          JSR      WCB      ;write cassette buffer
          BMI      CCL1      ;if error, exit

;      Zero buffer.

CCL3     LDX      #127      ;offset to last byte in buffer
          LDA      #0

CCL4     STA      CASBUF+3,X ;zero byte
          DEX
          BPL      CCL4      ;if not done

;      Write cassette buffer.

          LDA      #EOT      ;indicate EOT record type
          JSR      WCB      ;write cassette buffer

;      Exit.

```

JMP CCL1 ;exit

```
** AUB - Alert User with Beep
*
* ON ENTRY A= FREQ
*
* ENTRY JSR AUB
*
* MODS
* Original Author Unknown
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01
```

```
AUB = * ;entry
; Initialize.
STA FREQ ;frequency
; Compute termination time of beep duration.
AUB1 LDA RTCLK+2 ;current time
CLC
LDX PALNTS
ADC BEEPNTS,X ;add constant for 1 second tone
TAX ;beep duration termination time
; Turn on speaker.
AUB2 LDA #$FF
STA CONSOL ;turn on speaker
LDA #$00
; Delay.
LDY #$F0
AUB3 DEY
BNE AUB3 ;if not done delaying
; Turn off speaker.
STA CONSOL ;turn off speaker
; Delay.
LDY #$F0
AUB4 DEY
BNE AUB4 ;if not done delaying
; Check for beep duration termination time.
CPX RTCLK+2 ;compare current time
BNE AUB2 ;if termination time not reached
DEC FREQ ;decrement frequency
BEQ AUB6 ;if all done, wait for another key
; Compute termination time of beep separation.
TXA
CLC
LDX PALNTS
```

```

ADC    BEEPFX,X      ;add constant
TAX                                ;beep separation termination time

;    Wait for termination of beep separation.

AUB5   CPX    RTCLOK+2    ;compare current time
      BNE    AUB5      ;if termination time not reached

;    Beep again.

      BEQ    AUB1      ;beep again

;    Wait for key.

AUB6   JSR    WFK      ;wait for key
      TYA                                ;status
      RTS                                ;return

```

```

**    WFK - Wait for Key
*
*    ENTRY JSR    WFK
*
*    NOTES
*        Problem: bytes wasted by not doing LDA #high[KGB-1]
*        and LDA #low[KGB-1].
*        Problem: bytes wasted by this being a subroutine.
*
*    MODS
*        Original Author Unknown
*        1. Bring closer to Coding Standard (object unchanged).
*        R. K. Nordin    1983-11-01

```

```

WFK    =    *          ;entry
      LDA    KEYBDV+5    ;keyboard GET-BYTE routine address
      PHA                                ;put address on stack
      LDA    KEYBDV+4
      PHA
      RTS                ;invoke keyboard GET-BYTE routine

```

```

**    SCB - Perform SIO on Cassette Buffer
*
*    ENTRY JSR    SCB
*
*    NOTES
*        Problem: byte wasted by JSR/RTS exit.
*
*    MODS
*        Original Author Unknown
*        1. Bring closer to Coding Standard (object unchanged).
*        R. K. Nordin    1983-11-01

```

```

SCB    =    *          ;entry
      STA    DCOMND      ;command
      LDA    #high 131
      STA    DBYTHI      ;buffer length
      LDA    #low 131
      STA    DBYTLO
      LDA    #high CASBUF
      STA    DBUFHI      ;buffer address
      LDA    #low CASBUF
      STA    DBUFLO
      LDA    #$60        ;cassette bus ID
      STA    DDEVIC
      LDA    #0

```

```

STA    DUNIT
LDA    #35          ;timeout
STA    DTIMLO
LDA    DCOMND        ;command
LDY    #GETDAT       ;assume SIO GET-DATA command
CMP    #READ
BEQ    SCB1          ;if READ command

LDY    #PUTDAT       ;SIO PUT-DATA command

SCB1   STY    DSTATS  ;SIO command
LDA    FTYPE         ;IRG type
STA    DAUX2         ;second auxiliary information
JSR    SIOV          ;vector to SIO
RTS                ;return

```

```

**      WCB - Write Cassette Buffer
*
*      ENTRY JSR    WCB
*
*      NOTES
*          Problem: byte wasted by JSR/RTS exit.
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

WCB    =    *          ;entry
STA    CASBUF+2       ;record type
LDA    #$55
STA    CASBUF+0
STA    CASBUF+1
LDA    #'W'          ;write
JSR    SCB            ;perform SIO on cassette buffer
RTS                ;return

```

** NTSC/PAL Constant Tables

```

WLEADH DB    high WLEADN ;high NTSC WRITE file leader
DB    high WLEADP ;high PAL WRITE file leader

WLEADL DB    low WLEADN  ;low NTSC WRITE file leader
DB    low WLEADP  ;low PAL WRITE file leader

RLEADH DB    high RLEADN ;high NTSC READ file leader
DB    high RLEADP ;high PAL READ file leader

RLEADL DB    low RLEADN  ;low NTSC READ file leader
DB    low RLEADP  ;low PAL READ file leader

BEEPNX DB    BEEPNN      ;NTSC beep duration
DB    BEEPNP      ;PAL beep duration

BEEPFx DB    BEEPFN      ;NTSC beep separation
DB    BEEFPF      ;PAL beep separation

```

Printer Handler

```
**      PIN - Initialize Printer
*
*      ENTRY JSR      PIN
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
PIN      =      *      ;entry
          LDA      #30      ;30 second timeout
          STA      PTIMOT ;printer timeout
          RTS
          ;return
```

```
**      Printer Handler Address Data
*
*      NOTES
*          Problem: bytes wasted by tables and code. Load
*          Immediate instructions should be used.
```

```
PSTB      DW      DVSTAT ;status buffer address
PPRB      DW      PRNBUF ;printer buffer address
```

```
**      PST - Perform Printer STATUS
*
*      ENTRY JSR      PST
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
PST      =      *      ;entry
;          Get status.

          LDA      #4      ;4 bytes for status
          STA      PBUFSZ ;buffer size
          LDX      PSTB    ;address of status buffer
          LDY      PSTB+1
          LDA      #STATC ;status command
          STA      DCOMND ;command
          STA      DAUX1
          JSR      SDP      ;set up DCB for printer
          JSR      SIOV     ;vector to SIO
          BMI      PSP      ;if error, return

;          Exit.

          JSR      STS      ;set printer timeout from status
          JMP      PSP      ;return
```

```

**      PSP - Perform Printer SPECIAL
*
*      PSP does nothing.
*
*      ENTRY JSR    PSP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PSP      =      *      ;entry
          RTS          ;return

```

```

**      POP - Perform Printer OPEN
*
*      ENTRY JSR    POP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

POP      =      *      ;entry
          JSR      PST  ;perform printer STATUS
          LDA      #0
          STA      PBPNT ;clear printer buffer pointer
          RTS          ;return

```

```

**      PPB - Perform Printer PUT-BYTE
*
*      ENTRY JSR    PPB
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PPB      =      *      ;entry

;      Initialize.

          PHA          ;save data
          LDA      ICDNO,X      ;device number
          STA      ICDNOZ      ;device number
          JSR      PPM          ;process print mode

;      Put data in buffer.

          LDX      PBPNT      ;printer buffer pointer
          PLA          ;saved data
          STA      PRNBUF,X    ;put data in buffer
          INX

;      Check for buffer full.

          CPX      PBUFSZ      ;printer buffer size
          BEQ      PPP          ;if buffer full, perform PUT, return

;      Update printer buffer pointer.

          STX      PBPNT      ;printer buffer pointer

```

```

;      Check for EOL.

      CMP    #EOL
      BEQ    PPB1          ;if EOL, space fill

;      Exit.

      LDY    #SUCCES      ;indicate success
      RTS                      ;return

;      Space fill buffer.

PPB1   LDA    #' '          ;indicate space fill
;      JMP    FPB          ;fill printer buffer, return

```

```

**      FPB - Fill Printer Buffer
*
*      ENTRY JSR    FPB
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

FPB    =      *            ;entry

;      Fill printer buffer.

FPB1   STA    PRNBUF,X      ;byte of printer buffer
      INX
      CPX    PBUFSZ        ;printer buffer size
      BNE    FPB1          ;if not done

;      Perform printer PUT.

;      JMP    PPP          ;perform printer PUT, return

```

```

**      PPP - Perform Printer PUT
*
*      ENTRY JSR    PPP
*
*      MODS
*      Original Author Unknown
*      1. Bring closer to Coding Standard (object unchanged).
*      R. K. Nordin      1983-11-01

```

```

PPP    =      *            ;entry

;      Clear printer buffer pointer.

      LDA    #0
      STA    PBPNT          ;clear printer buffer pointer

;      Set up DCB.

      LDX    PPRB           ;address of printer buffer
      LDY    PPRB+1
      JSR    SDP            ;set up DCB for printer

;      Perform PUT.

      JMP    SIOV           ;vector to SIO, return

```

```

**      PCL - Perform Printer CLOSE
*
*      ENTRY JSR    PCL
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PCL      =      *      ;entry
;      Initialize.
      JSR      PPM      ;process print mode
;      Check buffer pointer.
      LDA      #EOL      ;indicate EOL fill
      LDX      PBPNT      ;printer buffer pointer
      BNE      FPB      ;if buffer pointer non-zero, fill buffer, return
;      Exit.
      LDY      #SUCCES      ;indicate success
      RTS

```

```

**      SDP - Set Up DCB for Printer
*
*      ENTRY JSR    SDP
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

SDP      =      *      ;entry
      STX      DBUFLO      ;low buffer address
      STY      DBUFHI      ;high buffer address
      LDA      #PDEVN      ;printer device bus ID
      STA      DDEVIC      ;device bus ID
      LDA      ICDNOZ      ;device number
      STA      DUNIT      ;unit number
      LDA      #$80      ;SIO WRITE command
      LDX      DCOMND      ;I/O direction
      CPX      #STATC      ;STATUS command
      BNE      SDP1      ;if STATUS command
      LDA      #$40      ;SIO READ command
SDP1     STA      DSTATS      ;SIO command
      LDA      PBUFSZ
      STA      DBYTLO      ;low buffer size
      LDA      #0
      STA      DBYTHI      ;high buffer size
      LDA      PTIMOT
      STA      DTIMLO      ;device timeout
      RTS

```

```

**      STS - Set Printer Timeout from Status
*
*      ENTRY JSR      STS
*
*      NOTES
*          Problem: bytes wasted by this code's being a subroutine.
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

STS      =      *          ;entry
          LDA      DVSTAT+2 ;timeout
          STA      PTIMOT   ;set printer timeout
          RTS                     ;return

```

```

**      PPM - Process Print Mode
*
*      PPM sets up the DCB according to the print mode.
*
*      ENTRY JSR      PPM
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

PPM      =      *          ;entry

;      Initialize.

          LDY      #WRITE ;WRITE command
          LDA      ICAX2Z ;print mode

;      Determine buffer size.

PPM1     CMP      #NORMAL      ;NORMAL mode
          BNE      PPM2        ;if not NORMAL mode

          LDX      #NBUFSZ      ;NORMAL mode buffer size
          BNE      PPM4        ;set buffer size

PPM2     CMP      #DOUBLE      ;DOUBLE mode
          BNE      PPM3        ;if not DOUBLE mode

          LDX      #DBUFSZ      ;DOUBLE mode buffer size
          BNE      PPM4        ;set buffer size

PPM3     CMP      #SIDWAY      ;SIDEWAYS mode
          BNE      PPM5        ;if not SIDEWAYS mode, assume NORMAL

          LDX      #SBUFSZ      ;SIDEWAYS mode buffer size

;      Set buffer size.

PPM4     STX      PBUFSZ ;set printer buffer size

;      Set DCB command and mode.

          STY      DCOMND ;command
          STA      DAUX1   ;print mode
          RTS                     ;return

```

```
;      Assume NORMAL mode.  
PPM5  LDA    #NORMAL      ;NORMAL mode  
      BNE    PPM1      ;set buffer size
```

Self-test, Part 4

```
**      VFR - Verify First 8K ROM
*
*      ENTRY JSR    VFR
*
*      EXIT
*          C clear, if verified
*          set, if not verified
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
VFR      =      *      ;entry
;      Initialize.
;
;          LDX      #0      ;offset to first region to checksum
;          STX      STCHK ;initial sum is zero
;          STX      STCHK+1
;
;      Checksum ROM.
VFR1     JSR      CRR      ;checksum region of ROM
;          CPX      #12
;          BNE      VFR1    ;if not done
;
;      Compare result.
;
;          LDA      $C000 ;low checksum in ROM
;          LDX      $C001 ;high checksum in ROM
;          JVP      VCS    ;verify checksum, return
```

```
**      VCS - Verify Checksum
*
*      ENTRY JSR    VCS
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01
```

```
VCS      =      *      ;entry
;          CMP      STCHK ;low checksum
;          BNE      VCS1  ;if low checksum bad
;
;          CPX      STCHK+1 ;high checksum
;          BNE      VCS1  ;if high checksum bad
;
;          CLC
;          RTS          ;indicate verified
;                      ;return
VCS1     SEC          ;indicate not verified
;          RTS          ;return
```

```

**      VSR - Verify Second 8K ROM
*
*      ENTRY JSR    VSR
*
*      EXIT
*          C clear, if verified
*          set, if not verified
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

VSR      =      *      ;entry
LDX      #0
STX      STCHK ;initial sum is zero
STX      STCHK+1
LDX      #12    ;offset to first region to checksum
JSR      CRR    ;checksum region of ROM
JSR      CRR    ;checksum region of ROM
LDA      $FFF8  ;low checksum from ROM
LDX      $FFF9  ;high checksum from ROM
JMP      VCS    ;verify checksum, return

```

```

**      CRR - Checksum Region of ROM
*
*      ENTRY JSR    CRR
*          X = offset
*
*      MODS
*          Original Author Unknown
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin      1983-11-01

```

```

CRR      =      *      ;entry
;      Transfer range addresses.
LDY      #0
CRR1     LDA     TARV,X
          STA     STADR1,Y
          INX
          INY
          CPY     #4    ;4 bytes for 2 addresses
          BNE     CRR1  ;if not done
;      Checksum range.
LDY      #0
CRR2     CLC
          LDA     (STADR1),Y
          ADC     STCHK
          STA     STCHK
          BCC     CRR3  ;if low value non-zero
          INC     STCHK+1    ;adjust high value
CRR3     INC     STADR1 ;advance address
          BNE     CRR4  ;if low address non-zero
          INC     STADR1+1    ;adjust high address

```



```

CRR4  LDA    STADR1 ;current address
      CMP    STADR2 ;end of range
      BNE    CRR2  ;if not done

      LDA    STADR1+1
      CMP    STADR2+1
      BNE    CRR2  ;if not done

      RTS                    ;return

```

**	TARV - Table of Address Ranges to Verify
----	--

TARV	DW	\$C002,\$D000	;first 8K ROM, \$C002 - \$CFFF
	DW	\$5000,\$5800	;first 8K ROM, \$D000 - \$D7FF
	DW	\$D800,\$E000	;first 8K ROM, \$D800 - \$DFFF
	DW	\$E000,\$FFF8	;second 8K ROM, \$E000 - \$FFF7
	DW	\$FFFA,\$0000	;second 8K ROM, \$FFFA - \$FFFF

Second 8K ROM Identification and Checksum

FIX \$FFEE

** Second 8K ROM Identification and Checksum
--

DB	IDDAY, IDMON, IDYEAR	;date (day, month, year)
DB	IDCPU	;CPU series
DB	IDPN1, IDPN2, IDPN3, IDPN4, IDPN5	;part number
DB	IDREV	;revision number
DW	\$0000	;reserved for checksum

6502 Machine Vectors

FIX \$FFFA

** 6502 Machine Vectors

DW NMI ;vector to process NMI
DW RES ;vector to process RESET
DW IRQ ;vector to process IRQ

END