

OS, Operating System, Revision 3, Version 4 for 600XL / 800XL / 1450XLD

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
Title of document:	OS, Operating System, Revision 3, Version 4 for 600XL / 800XL / 1450XLD
Author(s):	Multiple authors. Last update by Mike Barall (Michael Barall)
Original file date:	1984-06-21
Type of document:	Software, commented source code
Target audience:	Internal
Status:	Ready
Reference (Atari):	(unknown)
Reference (Laurent Delsarte):	For any discussion, this PDF has been given the reference BKUP-1984-06-21-SOFT-0008B-D which should be quoted in any communication.
Tags:	#Atari #8bit #6502 #400 #800 #1200XL #600XL #800XL #1450XLD #OS

Comments:

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

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

The most important part of my job was on presentation, to make it easier to read and understand. Comments are colored in grey. The source code presentation blocks are coloured in blue, and framed.

Revision B of this .PDF version:

Additional formatting:

- Short blocks of comments are coloured in green, and framed.
- Important 6502 instructions are highlighted in fluo.
JMP JSR RTS BEQ BNE BMI BPL BCS BCC BVS BVC
- Additional entries in the navigation for the main routines with entry points.
- Table of contents & Various index at the end.

This page intentionally left blank

OS, Operating System, Revision 3, Version 4 for 600XL / 800XL / 1450XLD

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 source is currently being refined.

Areas which need work are indicated by question marks ("???").

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

Revision 3 (600XL/800XL/1450XLD)

Fix MAXDEV, problems resulting from CRASS65 version, initial address for RAM sizing, "Boot Error" message, initial address for cartridge equivalence checksum, mishandling of SIO NAK, initializing of CHKSUM, and initialization of PORTB.

R. K. Nordin – 1984-03-27

Revision 3, Version 2 (600XL/800XL/1450XLD)

Dedicate PDVI (\$D1FF) to external parallel device IRQ status

Dedicate IPDVI (\$D1CF) to internal parallel device IRQ status

Using PDIMSK (\$0249) for external parallel device IRQ selection mask

Using IPDIMK (\$0254) for internal parallel device IRQ selection mask

After masking (PDVI, PDIMSK) & (IPDVI, IPDIMK), OR the result together, prior to processing parallel device IRQ

On cold start, initialize PDVI = 0, to avoid potential checksum error.

Y. T. Jang, V. Wu – 1984-02-22

Revision 3, Version 3 (600XL/800XL/1450XLD)

Dedicate the 11 bytes at ACMVAR (\$3ED-\$3F7) for use as a RESET routine area.

On warmstart, the OS will JSR to ACMVAR immediately after initializing hardware.

Mike Barall – 1984-06-08

Revision 3, Version 4 (600XL/800XL/1450XLD)

Make CIO accept device number 0 (like Rev. B did).

Mike Barall – 1984-06-21

History

??? Insert here a history similar to the portion in Scheiman's Usage Guidelines.

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 = 0
TRUE  = !N FALSE

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

** Identification Equates

```
IDREV = $03 ;identification revision number
IDDAY = $21 ;identification day
IDMON = $06 ;identification month
IDYEAR = $84 ;identification year
IDCPU = $02 ;identification CPU series
IDPN1 = 'B' ;identification part number field 1
IDPN2 = 'B' ;identification part number field 2
IDPN3 = $00 ;identification part number field 3
IDPN4 = $00 ;identification part number field 4
IDPN5 = $02 ;identification part number field 5
```

** Configuration Equates

```
MAXDEV = 30 ;offset to last possible entry of HATABS
IOCBSZ = 16 ;length of IOCB

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

DSCTSZ = 128 ;disk sector size

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

INIML = $0700 ;initial MEMLO

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

** IOCB Command Code Equates

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

** Special Entry Command Equates

```
; Screen Commands

DRAWLN = $11 ;draw line
FILLIN = $12 ;draw line with right fill
```

**** ICAX1 Auxiliary Byte 1 Equates**

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

**** Device Code Equates**

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

**** Character and Key Code Equates**

CLS	=	\$7D	;clear screen
EOL	=	\$9B	;end of line (RETURN)
HELP	=	\$11	;key code for HELP
CNTLF1	=	\$83	;key code for CTRL-F1
CNTLF2	=	\$84	;key code for CTRL-F2
CNTLF3	=	\$93	;key code for CTRL-F3
CNTLF4	=	\$94	;key code for CTRL-F4
CNTL1	=	\$9F	;key code for CTRL-1

**** Status Code Equates**

SUCCESS	=	1	;successful operation
BRKABT	=	128	;BREAK key abort
PRVOPN	=	129	;IOCB already open error
NONDEV	=	130	;nonexistent device error
WRONLY	=	131	;IOCB opened for write only error
INVALID	=	132	;invalid command error
NOTOPN	=	133	;device/file not open error
BADIOC	=	134	;invalid IOCB index error
RONLY	=	135	;IOCB opened for read only error
EOFERR	=	136	;end of file error
TRNRCD	=	137	;truncated record error
TIMOUT	=	138	;peripheral device timeout error
DNACK	=	139	;device does not acknowledge command error
FRMERR	=	140	;serial bus framing error
CRSROR	=	141	;cursor overrange error
OVRRUN	=	142	;serial bus data overrun error
CHKERR	=	143	;serial bus checksum error
DERROR	=	144	;device done (operation incomplete) error
BADMOD	=	145	;bad screen mode number error
FNCNOT	=	146	;function not implemented in handler error
SCRMEM	=	147	;insufficient memory for screen mode error

**** DCB Device Bus ID Equates**

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

** Bus Command Equates

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

** Command Auxiliary Byte Equates

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

** Bus Response Equates

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

** Floating Point Package Miscellaneous Equates

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

** Cassette Record Type Equates

HDR	=	\$FB	;header
DTA	=	\$FC	;data record
DT1	=	\$FA	;last data record
EOT	=	\$FE	;end of tape (file)
TONE1	=	2	;record
TONE2	=	1	;playback

** Cassette Timing Equates
--

WLEADN	=	1152	;NTSC 19.2 second WRITE file leader
RLEADN	=	576	;NTSC 9.6 second READ file leader
WIRGLN	=	180	;NTSC 3.0 second WRITE IRG
RIRGLN	=	120	;NTSC 2.0 second READ IRG
WSIRGN	=	15	;NTSC 0.25 second WRITE short IRG
RSIRGN	=	10	;NTSC 0.16 second READ short IRG
BEEPNN	=	30	;NTSC 0.5 second beep duration
BEEPFN	=	10	;NTSC 0.16 second beep separation
WLEADP	=	960	;PAL 19.2 second WRITE file leader
RLEADP	=	480	;PAL 9.6 second READ file leader
WIRGLP	=	150	;PAL 3.0 second WRITE IRG
RIRGLP	=	100	;PAL 2.0 second READ IRG
WSIRGP	=	13	;PAL 0.25 second WRITE short IRG
RSIRGP	=	8	;PAL 0.16 second READ short IRG
BEEPNP	=	25	;PAL 0.5 second beep duration
BEEPPF	=	8	;PAL 0.16 second beep separation
WIRGHI	=	0	;high WRITE IRG
RIRGHI	=	0	;high READ IRG

** Power-up Validation Byte Value Equates
--

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

** Relocating Loader Miscellaneous Equates

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

** Miscellaneous Equates

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

System Address Equates

** Page Zero Address Equates

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

;      Not Cleared

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

;      Cleared upon Coldstart Only

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

;      Cleared upon Coldstart or Warmstart

INTZBS  =      $0010   ;first page zero location to clear

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

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

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

STATUS  =      $0030   ;1-byte SIO operation status
CHKSUM  =      $0031   ;1-byte checksum (single byte sum with carry)
BUFRLO  =      $0032   ;1-byte low data buffer address
BUFRHI  =      $0033   ;1-byte high data buffer address
BFENLO  =      $0034   ;1-byte low data buffer end address
BFENHI  =      $0035   ;1-byte high data buffer end address
LTEMP  =      $0036   ;2-byte relocating loader temporary
BUFRFL  =      $0038   ;1-byte data buffer full flag (0 = not full)
RECVDN  =      $0039   ;1-byte receive-frame done flag (0 = not done)
XMTDON  =      $003A   ;1-byte transmit-frame done flag (0 = not done)
CHKSNT  =      $003B   ;1-byte checksum sent flag (0 = not sent)
NOCKSM  =      $003C   ;1-byte no checksum follows data flag (0 = does)
BPTR    =      $003D   ;1-byte cassette buffer pointer
FTYPE   =      $003E   ;1-byte cassette IRG type (neg. = continuous)
FEOF    =      $003F   ;1-byte cassette EOF flag (0 = no EOF)
FREQ    =      $0040   ;1-byte cassette beep counter
SOUNDR  =      $0041   ;1-byte noisy I/O flag (0 = quiet)

CRITIC  =      $0042   ;1-byte critical section flag (0 = not critical)

FMSZPG  =      $0043   ;7-byte reserved for DOS File Management System

ZCHAIN  =      $004A   ;2-byte handler linkage chain pointer
DSTAT   =      $004C   ;1-byte display status
ATTRACT =      $004D   ;1-byte attract-mode timer and flag
DRKMSK  =      $004E   ;1-byte attract-mode dark (luminance) mask
```

COLORSH	=	\$004F	;1-byte attract-mode color shift
TMPCHR	=	\$0050	;1-byte temporary character
HOLD1	=	\$0051	;1-byte temporary
LMARGN	=	\$0052	;1-byte text column left margin
RMARGN	=	\$0053	;1-byte text column right margin
ROWCRS	=	\$0054	;1-byte cursor row
COLCRS	=	\$0055	;2-byte cursor column
DINDEX	=	\$0057	;1-byte display mode
SAVMSC	=	\$0058	;2-byte saved memory scan counter
OLDROW	=	\$005A	;1-byte prior row
OLDCOL	=	\$005B	;2-byte prior column
OLDCHR	=	\$005D	;1-byte saved character under cursor
OLDADR	=	\$005E	;2-byte saved cursor memory address
FKDEF	=	\$0060	;2-byte function key definition table address
PALNTS	=	\$0062	;1-byte PAL/NTSC indicator (0 = NTSC, PAL = 1)
LOGCOL	=	\$0063	;1-byte logical line cursor column
ADRESS	=	\$0064	;2-byte temporary address
MLTTMP	=	\$0066	;1-byte temporary
OPNTMP	=	\$0066	;1-byte open temporary
TOADR	=	\$0066	;2-byte destination address
SAVADR	=	\$0068	;2-byte saved address
FRMADR	=	\$0068	;2-byte source address
RAMTOP	=	\$006A	;1-byte RAM size
BUFCNT	=	\$006B	;1-byte buffer count (logical line size)
BUFSTR	=	\$006C	;2-byte buffer start pointer
BITMSK	=	\$006E	;1-byte bit mask for bit map operation
SHFAMT	=	\$006F	;1-byte shift amount for pixel justification
ROWAC	=	\$0070	;2-byte draw working row
COLAC	=	\$0072	;2-byte draw working column
ENDPT	=	\$0074	;2-byte end point
DELTAR	=	\$0076	;1-byte row difference
DELTAC	=	\$0077	;2-byte column difference
KEYDEF	=	\$0079	;2-byte key definition table address
SWPFLG	=	\$007B	;1-byte split screen swap flag (0 = not swapped)
HOLDCH	=	\$007C	;1-byte temporary character
INSDAT	=	\$007D	;1-byte temporary
COUNTR	=	\$007E	;2-byte draw iteration count
;			Reserved for Application and Floating Point Package
;	=	\$0080	;128 bytes reserved for application and FPP

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

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

```

ZTEMP1 = $00F5 ;2-byte temporary
ZTEMP4 = $00F7 ;2-byte temporary
ZTEMP3 = $00F9 ;2-byte temporary

FLPTR = $00FC ;2-byte floating point number pointer
FPTR2 = $00FE ;2-byte floating point number pointer

```

** Page One (Stack) Address Equates
--

```

; = $0100 ;256-byte stack

```

** Page Two Address Equates

```

INTABS = $0200 ;42-byte interrupt handler table

VDSLST = $0200 ;2-byte display list NMI vector
VPRCED = $0202 ;2-byte serial I/O proceed line IRQ vector
VINTER = $0204 ;2-byte serial I/O interrupt line IRQ vector
VBREAK = $0206 ;2-byte BRK instruction IRQ vector
VKEYBD = $0208 ;2-byte keyboard IRQ vector
VSERIN = $020A ;2-byte serial input ready IRQ vector
VSEORR = $020C ;2-byte serial output ready IRQ vector
VSEORC = $020E ;2-byte serial output complete IRQ vector
VTIMR1 = $0210 ;2-byte POKEY timer 1 IRQ vector
VTIMR2 = $0212 ;2-byte POKEY timer 2 IRQ vector
VTIMR4 = $0214 ;2-byte POKEY timer 4 IRQ vector
VIMIRQ = $0216 ;2-byte immediate IRQ vector
CDTMV1 = $0218 ;2-byte countdown timer 1 value
CDTMV2 = $021A ;2-byte countdown timer 2 value
CDTMV3 = $021C ;2-byte countdown timer 3 value
CDTMV4 = $021E ;2-byte countdown timer 4 value
CDTMV5 = $0220 ;2-byte countdown timer 5 value
VVBLLKI = $0222 ;2-byte immediate VBLANK NMI vector
VVBLLKD = $0224 ;2-byte deferred VBLANK NMI vector
CDTMA1 = $0226 ;2-byte countdown timer 1 vector
CDTMA2 = $0228 ;2-byte countdown timer 2 vector

CDTMF3 = $022A ;1-byte countdown timer 3 flag (0 = expired)
SRTIMR = $022B ;1-byte software key repeat timer
CDTMF4 = $022C ;1-byte countdown timer 4 flag (0 = expired)
INTEMP = $022D ;1-byte temporary
CDTMF5 = $022E ;1-byte countdown timer 5 flag (0 = expired)
SDMCTL = $022F ;1-byte DMACTL shadow
SDLSTL = $0230 ;1-byte DLISTL shadow
SDLSTH = $0231 ;1-byte DLISTH shadow
SSKCTL = $0232 ;1-byte SKCTL shadow
LCOUNT = $0233 ;1-byte relocating loader record length counter
LPENH = $0234 ;1-byte light pen horizontal value
LPENV = $0235 ;1-byte light pen vertical value
BRKKY = $0236 ;2-byte BREAK key vector
VPIRQ = $0238 ;2-byte parallel device IRQ vector
CDEVIC = $023A ;1-byte command frame device ID
CCOMND = $023B ;1-byte command frame command
CAUX1 = $023C ;1-byte command auxiliary 1
CAUX2 = $023D ;1-byte command auxiliary 2

TEMP = $023E ;1-byte temporary

ASSERT low TEMP<>$FF ;may not be the last word on a page

ERRFLG = $023F ;1-byte I/O error flag (0 = no error)

ASSERT low ERRFLG<>$FF ;may not be the last word on a page

DFlags = $0240 ;1-byte disk flags from sector 1
DBSECT = $0241 ;1-byte disk boot sector count
BOOTAD = $0242 ;2-byte disk boot memory address
COLDST = $0244 ;1-byte coldstart flag (0 = complete)
RECLen = $0245 ;1-byte relocating loader record length
DSKTIM = $0246 ;1-byte disk format timeout
PDVMSK = $0247 ;1-byte parallel device selection mask
SHPDVS = $0248 ;1-byte parallel device selection shadow
PDIMSK = $0249 ;1-byte external parallel device IRQ selection mask
RELADR = $024A ;2-byte relocating loader relative address
PPTMPA = $024C ;1-byte parallel device handler temporary
PPTMPX = $024D ;1-byte parallel device handler temporary
DSKCNT = $024E ;1-byte parallel disk device count
TOPSLT = $024F ;1-byte highest parallel disk device slot

```

SPBCTL	=	\$0250	;1-byte PBICTL shadow
NRNGS	=	\$0251	;1-byte parallel modem ring detect counter
VPCTR	=	\$0252	;1-byte parallel voice phoneme counter
VMCTR	=	\$0253	;1-byte parallel voice marker counter
IPDIMK	EQU	\$0254	;1-byte internal parallel device IRQ selection mask
;			
	=	\$0255	;22 bytes reserved for Atari
CHSALT	=	\$026B	;1-byte character set alternate
VSFLAG	=	\$026C	;1-byte fine vertical scroll count
KEYDIS	=	\$026D	;1-byte keyboard disable
FINE	=	\$026E	;1-byte fine scrolling mode
GPRIOR	=	\$026F	;1-byte PRIOR shadow
PADDL0	=	\$0270	;1-byte potentiometer 0
PADDL1	=	\$0271	;1-byte potentiometer 1
PADDL2	=	\$0272	;1-byte potentiometer 2
PADDL3	=	\$0273	;1-byte potentiometer 3
PADDL4	=	\$0274	;1-byte potentiometer 4
PADDL5	=	\$0275	;1-byte potentiometer 5
PADDL6	=	\$0276	;1-byte potentiometer 6
PADDL7	=	\$0277	;1-byte potentiometer 7
STICK0	=	\$0278	;1-byte joystick 0
STICK1	=	\$0279	;1-byte joystick 1
STICK2	=	\$027A	;1-byte joystick 2
STICK3	=	\$027B	;1-byte joystick 3
PTRIG0	=	\$027C	;1-byte paddle trigger 0
PTRIG1	=	\$027D	;1-byte paddle trigger 1
PTRIG2	=	\$027E	;1-byte paddle trigger 2
PTRIG3	=	\$027F	;1-byte paddle trigger 3
PTRIG4	=	\$0280	;1-byte paddle trigger 4
PTRIG5	=	\$0281	;1-byte paddle trigger 5
PTRIG6	=	\$0282	;1-byte paddle trigger 6
PTRIG7	=	\$0283	;1-byte paddle trigger 7
STRIG0	=	\$0284	;1-byte joystick trigger 0
STRIG1	=	\$0285	;1-byte joystick trigger 1
STRIG2	=	\$0286	;1-byte joystick trigger 2
STRIG3	=	\$0287	;1-byte joystick trigger 3
HIBYTE	=	\$0288	;1-byte relocating loader high byte value
WMODE	=	\$0289	;1-byte cassette WRITE mode (\$80 = writing)
BLIM	=	\$028A	;1-byte cassette buffer limit
IMASK	=	\$028B	;1-byte (not used)
JVECK	=	\$028C	;2-byte jump vector or temporary
NEWADR	=	\$028E	;2-byte relocating address
TXTR0W	=	\$0290	;1-byte split screen text cursor row
TXTCOL	=	\$0291	;2-byte split screen text cursor column
TINDEX	=	\$0293	;1-byte split screen text mode
TXTMSC	=	\$0294	;2-byte split screen memory scan counter
TXTOLD	=	\$0296	;6-byte OLDROW, OLDCOL, OLDCHR, OLDADR for text
CRETRY	=	\$029C	;1-byte number of command frame retries
HOLD3	=	\$029D	;1-byte temporary
SUBTMP	=	\$029E	;1-byte temporary
HOLD2	=	\$029F	;1-byte (not used)
DMASK	=	\$02A0	;1-byte display (pixel location) mask
TMPLBT	=	\$02A1	;1-byte (not used)
ESCFLG	=	\$02A2	;1-byte escape flag (\$80 = ESC detected)
TABMAP	=	\$02A3	;15-byte (120-bit) tab stop bit map
LOGMAP	=	\$02B2	;8-byte (32-bit) logical line bit map
INVFLG	=	\$02B6	;1-byte inverse video flag (\$80 = inverse)
FILFLG	=	\$02B7	;1-byte right fill flag (0 = no fill)
TMPROW	=	\$02B8	;1-byte temporary row
TMPCOL	=	\$02B9	;2-byte temporary column
SCRFLG	=	\$02BB	;1-byte scroll occurrence flag (0 = not occurred)
HOLD4	=	\$02BC	;1-byte temporary
DRETRY	=	\$02BD	;1-byte number of device retries
SHFLOK	=	\$02BE	;1-byte shift/control lock flags
BOTSCR	=	\$02BF	;1-byte screen bottom (24 = normal, 4 = split)
PCOLR0	=	\$02C0	;1-byte player-missile 0 color/luminance
PCOLR1	=	\$02C1	;1-byte player-missile 1 color/luminance
PCOLR2	=	\$02C2	;1-byte player-missile 2 color/luminance
PCOLR3	=	\$02C3	;1-byte player-missile 3 color/luminance
COLOR0	=	\$02C4	;1-byte playfield 0 color/luminance
COLOR1	=	\$02C5	;1-byte playfield 1 color/luminance
COLOR2	=	\$02C6	;1-byte playfield 2 color/luminance
COLOR3	=	\$02C7	;1-byte playfield 3 color/luminance
COLOR4	=	\$02C8	;1-byte background color/luminance

PARMBL	=	\$02C9	;6-byte relocating loader parameter block
RUNADR	=	\$02C9	;2-byte run address
HIUSED	=	\$02CB	;2-byte highest non-zero page address
ZHIUSE	=	\$02CD	;2-byte highest zero page address
OLDPAR	=	\$02CF	;6-byte relocating loader parameter block
GBYTEA	=	\$02CF	;2-byte GET-BYTE routine address
LOADAD	=	\$02D1	;2-byte non-zero page load address
ZLOADA	=	\$02D3	;2-byte zero page load address
DSCTLN	=	\$02D5	;2-byte disk sector length
ACMISR	=	\$02D7	;2-byte ACMI interrupt service routine address
KRPDEL	=	\$02D9	;1-byte auto-repeat delay
KEYREP	=	\$02DA	;1-byte auto-repeat rate
NOCLIK	=	\$02DB	;1-byte key click disable
HELPPG	=	\$02DC	;1-byte HELP key flag (0 = no HELP)
DMASAV	=	\$02DD	;1-byte SDMCTL save/restore
PBPNT	=	\$02DE	;1-byte printer buffer pointer
PBUFSZ	=	\$02DF	;1-byte printer buffer size
;	=	\$02E0	;4-byte reserved for DOS
RAMSIZ	=	\$02E4	;1-byte high RAM size
MEMTOP	=	\$02E5	;2-byte top of available user memory
MEMLO	=	\$02E7	;2-byte bottom of available user memory
HNDLOD	=	\$02E9	;1-byte user load flag (0 = no handler load)
DVSTAT	=	\$02EA	;4-byte device status buffer
CBAUDL	=	\$02EE	;1-byte low cassette baud rate
CBAUDH	=	\$02EF	;1-byte high cassette baud rate
CRSINH	=	\$02F0	;1-byte cursor inhibit (0 = cursor on)
KEYDEL	=	\$02F1	;1-byte key debounce delay timer
CH1	=	\$02F2	;1-byte prior keyboard character
CHACT	=	\$02F3	;1-byte CHACTL shadow
CHBAS	=	\$02F4	;1-byte CHBASE shadow
NEWROW	=	\$02F5	;1-byte draw destination row
NEWCOL	=	\$02F6	;2-byte draw destination column
ROWINC	=	\$02F8	;1-byte draw row increment
COLINC	=	\$02F9	;1-byte draw column increment
CHAR	=	\$02FA	;1-byte internal character
ATACHR	=	\$02FB	;1-byte ATASCII character or plot point
CH	=	\$02FC	;1-byte keyboard code (buffer)
FILDAT	=	\$02FD	;1-byte right fill data
DSPFLG	=	\$02FE	;1-byte control character display flag (0 = no)
SSFLAG	=	\$02FF	;1-byte start/stop flag (0 = not stopped)

** Page Three Address Equates

DCB	=	\$0300	;12-byte device control block
DDEVIC	=	\$0300	;1-byte unit 1 bus ID
DUNIT	=	\$0301	;1-byte unit number
DCOMND	=	\$0302	;1-byte bus command
DSTATS	=	\$0303	;1-byte command type/status return
DBUFLO	=	\$0304	;1-byte low data buffer address
DBUFHI	=	\$0305	;1-byte high data buffer address
DTIMLO	=	\$0306	;1-byte timeout (seconds)
DUNUSE	=	\$0307	;1-byte (not used)
DBYTLO	=	\$0308	;1-byte low number of bytes to transfer
DBYTHI	=	\$0309	;1-byte high number of bytes to transfer
DAUX1	=	\$030A	;1-byte first command auxiliary
DAUX2	=	\$030B	;1-byte second command auxiliary
TIMER1	=	\$030C	;2-byte initial baud rate timer value
JMPERS	=	\$030E	;1-byte jumper options
CASFLG	=	\$030F	;1-byte cassette I/O flag (0 = not cassette I/O)
TIMER2	=	\$0310	;2-byte final baud rate timer value
TEMP1	=	\$0312	;2-byte temporary
TEMP2	=	\$0313	;1-byte temporary
PTIMOT	=	\$0314	;1-byte printer timeout
TEMP3	=	\$0315	;1-byte temporary
SAVIO	=	\$0316	;1-byte saved serial data input indicator
TIMFLG	=	\$0317	;1-byte timeout flag (0 = timeout)
STACKP	=	\$0318	;1-byte SIO saved stack pointer
TSTAT	=	\$0319	;1-byte temporary status
HATABS	=	\$031A	;35-byte handler address table

```

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

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

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

; Remainder of Page Three Not Cleared upon Reset

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

```

** Page Four Address Equates

```

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

; Reserved for Application

USAREA = $0480 ;128 bytes reserved for application

```

** Page Five Address Equates

```

; Reserved for Application and Floating Point Package

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

```

** Floating Point Package Address Equates [1]

```

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

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

```

** Page Six Address Equates

```

; Reserved for Application

; = $0600 ;256 bytes reserved for application

```

** LNBUG Address Equates			
LNBUG	IF	LNBUG	
LNORG	=	\$6000	;LNBUG origin
LNIRQ	=	\$6033	;LNBUG IRQ entry
LNNMI	=	\$8351	;LNBUG NMI vector
LNBUG	ENDIF		

** Cartridge Address Equates			
CARTCS	=	\$BFFA	;2-byte cartridge coldstart address
CART	=	\$BFFC	;1-byte cartridge present indicator
CARTFG	=	\$BFFD	;1-byte cartridge flags
CARTAD	=	\$BFFE	;2-byte cartridge start vector

** CTIA/GTIA Address Equates			
CTIA	=	\$D000	;CTIA/GTIA area
; Read/Write Addresses			
CONSOL	=	\$D01F	;console switches and speaker control
; Read Addresses			
M0PF	=	\$D000	;missile 0 and playfield collision
M1PF	=	\$D001	;missile 1 and playfield collision
M2PF	=	\$D002	;missile 2 and playfield collision
M3PF	=	\$D003	;missile 3 and playfield collision
P0PF	=	\$D004	;player 0 and playfield collision
P1PF	=	\$D005	;player 1 and playfield collision
P2PF	=	\$D006	;player 2 and playfield collision
P3PF	=	\$D007	;player 3 and playfield collision
M0PL	=	\$D008	;missile 0 and player collision
M1PL	=	\$D009	;missile 1 and player collision
M2PL	=	\$D00A	;missile 2 and player collision
M3PL	=	\$D00B	;missile 3 and player collision
P0PL	=	\$D00C	;player 0 and player collision
P1PL	=	\$D00D	;player 1 and player collision
P2PL	=	\$D00E	;player 2 and player collision
P3PL	=	\$D00F	;player 3 and player collision
TRIG0	=	\$D010	;joystick trigger 0
TRIG1	=	\$D011	;joystick trigger 1
TRIG2	=	\$D012	;cartridge interlock
TRIG3	=	\$D013	;ACMI module interlock
PAL	=	\$D014	;PAL/NTSC indicator
; Write Addresses			
HPOSP0	=	\$D000	;player 0 horizontal position
HPOSP1	=	\$D001	;player 1 horizontal position
HPOSP2	=	\$D002	;player 2 horizontal position
HPOSP3	=	\$D003	;player 3 horizontal position
HPOSM0	=	\$D004	;missile 0 horizontal position
HPOSM1	=	\$D005	;missile 1 horizontal position
HPOSM2	=	\$D006	;missile 2 horizontal position
HPOSM3	=	\$D007	;missile 3 horizontal position
SIZEP0	=	\$D008	;player 0 size
SIZEP1	=	\$D009	;player 1 size
SIZEP2	=	\$D00A	;player 2 size
SIZEP3	=	\$D00B	;player 3 size
SIZEM	=	\$D00C	;missile sizes
GRAFP0	=	\$D00D	;player 0 graphics
GRAFP1	=	\$D00E	;player 1 graphics
GRAFP2	=	\$D00F	;player 2 graphics
GRAFP3	=	\$D010	;player 3 graphics
GRAFM	=	\$D011	;missile graphics


```

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

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

COLBK = $D01A ;background color/luminance

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

```

**** PBI Address Equates**

```

PBI = $D100 ;parallel bus interface area

; Read/Write Addresses

VSTB = $D104 ;parallel voice strobe
PBICL = $D108 ;parallel bus control
PMDATA = $D10C ;parallel modem transmit/receive
PMCMD = $D10E ;parallel modem command
PMCTL = $D10F ;parallel modem control
DISKWR = $D110 ;parallel disk write
DISKRD = $D114 ;parallel disk read

; Read Addresses

PMSTAT = $D10D ;parallel modem status
PDVI = $D1FF ;external parallel device IRQ status
IPDVI EQU $D1CF ;internal parallel device IRQ status

; Write Addresses

VOICE1 = $D100 ;parallel voice data/interrupt control
DMST = $D108 ;parallel disk/modem status
PDVS = $D1FF ;parallel device select

```

**** POKEY Address Equates**

```

POKEY = $D200 ;POKEY area

; Read Addresses

POT0 = $D200 ;potentiometer 0
POT1 = $D201 ;potentiometer 1
POT2 = $D202 ;potentiometer 2
POT3 = $D203 ;potentiometer 3
POT4 = $D204 ;potentiometer 4
POT5 = $D205 ;potentiometer 5
POT6 = $D206 ;potentiometer 6
POT7 = $D207 ;potentiometer 7

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

; Write Addresses

AUDF1 = $D200 ;channel 1 audio frequency
AUCD1 = $D201 ;channel 1 audio control

AUDF2 = $D202 ;channel 2 audio frequency
AUCD2 = $D203 ;channel 2 audio control

AUDF3 = $D204 ;channel 3 audio frequency
AUCD3 = $D205 ;channel 3 audio control

AUDF4 = $D206 ;channel 4 audio frequency
AUCD4 = $D207 ;channel 4 audio control

```

```

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

```

** PIA Address Equates

```

PIA = $D300 ;PIA area

; Read/Write Addresses

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

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

```

** ANTIC Address Equates

```

ANTIC = $D400 ;ANTIC area

; Read Addresses

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

; Write Addresses

DMACTL = $D400 ;DMA control
CHACTL = $D401 ;character control
DLISTL = $D402 ;low display list address
DLISTH = $D403 ;high display list address
HSCROL = $D404 ;horizontal scroll
VSCROL = $D405 ;vertical scroll
PMBASE = $D407 ;player-missile base address
CHBASE = $D409 ;character base address
WSYNC = $D40A ;wait for HBLANK synchronization
NMIEEN = $D40E ;NMI enable
NMIRES = $D40F ;NMI interrupt status reset

```

** PBI RAM Address Equates

```

PBIRAM = $D600 ;parallel bus interface RAM area

```

** ACMI Address Equates

```

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

```

** Floating Point Package Address Equates [2]

```

AFP = $D800 ;convert ASCII to floating point
FASC = $D8E6 ;convert floating point to ASCII
IFP = $D9AA ;convert integer to floating point
FPI = $D9D2 ;convert floating point to integer
ZFR0 = $DA44 ;zero FR0
ZF1 = $DA46 ;zero floating point number
FSUB = $DA60 ;subtract floating point numbers
FADD = $DA66 ;add floating point numbers
FMUL = $DADB ;multiply floating point numbers
FDIV = $DB28 ;divide floating point numbers

```

PLYEVL	=	\$DD40	;evaluate floating point polynomial
FLD0R	=	\$DD89	;load floating point number
FLD0P	=	\$DD8D	;load floating point number
FLD1R	=	\$DD98	;load floating point number
FLD1P	=	\$DD9C	;load floating point number
FST0R	=	\$DDA7	;store floating point number
FST0P	=	\$DDAB	;store floating point number
FMOVE	=	\$DDB6	;move floating point number
LOG	=	\$DECD	;calculate floating point logarithm
LOG10	=	\$DED1	;calculate floating point base 10 logarithm
EXP	=	\$DDC0	;calculate floating point exponentiation
EXP10	=	\$DDCC	;calculate floating point base 10 exponentiation

** Parallel Device Address Equates

PDID1	=	\$D803	;parallel device ID 1
PDI0V	=	\$D805	;parallel device I/O vector
PDIRQV	=	\$D808	;parallel device IRQ vector
PDID2	=	\$D80B	;parallel device ID 2
PDVV	=	\$D80D	;parallel device vector table

** Device Handler Vector Table Address Equates

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

** Jump Vector Address Equates

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

** Generic Parallel Device Handler Vector Table Address Equates

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

Miscellaneous Address Equates

** Self-test Page Zero Address Equates

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

** Self-test Address Equates

ST3000	=	\$3000	;screen memory
ST3002	=	\$3002	;cleft display
ST3004	=	\$3004	;"VOICE #" text display
ST300B	=	\$300B	;voice number display
ST301C	=	\$301C	;START key display
ST301E	=	\$301E	;SELECT key display
ST3020	=	\$3020	;OPTION key display, first 8K ROM display
ST3021	=	\$3021	;keyboard character display
ST3022	=	\$3022	;keyboard text display
ST3024	=	\$3024	;second 8K ROM display
ST3028	=	\$3028	;"RAM" text display
ST3038	=	\$3038	;RAM display
ST303C	=	\$303C	;fifth note display
ST304C	=	\$304C	;"B S" text display
ST3052	=	\$3052	;tab key display
ST3062	=	\$3062	;cleft display
ST306D	=	\$306D	;return key display
ST3072	=	\$3072	;control key display
ST3092	=	\$3092	;"SH" text display
ST309E	=	\$309E	;sixth note display
ST30AB	=	\$30AB	;"SH" text display
ST30B7	=	\$30B7	;"S P A C E B A R" text display
ST30C1	=	\$30C1	;cleft display
ST30C2	=	\$30C2	;cleft display
ST30C7	=	\$30C7	;third note display
ST30CA	=	\$30CA	;fourth note display
ST30F8	=	\$30F8	;third note display
ST3100	=	\$3100	;screen memory
ST3121	=	\$3121	;cleft display
ST3122	=	\$3122	;cleft display
ST313C	=	\$313C	;fifth note display
ST3150	=	\$3150	;first line of staff display
ST3154	=	\$3154	;first note display
ST3181	=	\$3181	;cleft display
ST3182	=	\$3182	;cleft display
ST3186	=	\$3186	;second note display
ST318C	=	\$318C	;fifth note display
ST31B0	=	\$31B0	;second line of staff display
ST31C2	=	\$31C2	;cleft display
ST31CA	=	\$31CA	;fourth note display
ST31EE	=	\$31EE	;sixth note display
ST31F1	=	\$31F1	;cleft display
ST3210	=	\$3210	;third line of staff display
ST321A	=	\$321A	;fourth note display
ST3248	=	\$3248	;third note display
ST3270	=	\$3270	;fourth line of staff display
ST32D0	=	\$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
.BYTE      0
ENDM
ELSE
FIX        *0 + 255
FIX        %1
ENDIF
ELSE
ERR                ;%1 precedes current origin counter value
ENDIF
ENDIF
.AORG      *0
ENDM
```

First 8K ROM Identification and Checksum

.AORG \$C000

** First 8K ROM Identification and Checksum

.WORD	\$0000	;reserved for checksum
.BYTE	IDDAY, IDMON, IDYEAR	;date (day, month, year)
.BYTE	IDCPU	;CPU series
.BYTE	IDPN1, IDPN2, IDPN3, IDPN4, IDPN5	;part number
.BYTE	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      NMIEEN ;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 to caller
```

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      ;save X
          PHA      ;save Y
          PHA

;      Reset NMI status.

          STA      NMIREN ;reset NMI status

;      Process NMI.

LNBUG    IF      LNBUG
          LDA      LNFLAG ;LNBUG flag
          BNE      NMI2 ;if LNBUG
          JMP      (VVBLKI) ;process immediate VBLANK NMI, return
NMI2     JMP      (LNNMI) ;invoke LNBUG NMI routine, return
LNBUG    ELSE
          JMP      (VVBLKI) ;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
*                2. Y.T. Jang    1984-05-15
```

```
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      #!N $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      ;external parallel device IRQ statuses
      AND      PDIMSK      ;select desired external IRQ statuses
      STA      ABUFPT      ;temp 4 bytes reserved for parallel device IRQ area
      LDA      IPDVI      ;internal parallel device IRQ statuses
      AND      IPDIMK
      ORA      ABUFPT
      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
;      Coninue 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 from interrupt

```

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          ;saved ???
;      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      ;save ???
;
;      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      ;saved ???
;      PHA
;      JMP      (VBREAK)   ;process BRK instruction IRQ, return
;
;      Exit IRQ processing.
CIR3     LDA      JVECK      ;saved ???
;      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 from interrupt
```

AIR - Process ACMI IRQ

```
*      ENTRY   JSR      AIR
*
*      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     .BYTE   $80      ;0 - BREAK key IRQ
          .BYTE   $40      ;1 - keyboard IRQ
          .BYTE   $04      ;2 - timer 4 IRQ
          .BYTE   $02      ;3 - timer 2 IRQ
          .BYTE   $01      ;4 - timer 1 IRQ
          .BYTE   $08      ;5 - serial output complete IRQ
          .BYTE   $10      ;6 - serial output ready IRQ
          .BYTE   $20      ;7 - serial input ready IRQ
```

```
TIRQQL   =      *-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     .BYTE   BRKKY-INTABS ;0 - BREAK key IRQ
          .BYTE   VKEYBD-INTABS ;1 - keyboard IRQ
          .BYTE   VTIMR4-INTABS ;2 - timer 4 IRQ
          .BYTE   VTIMR2-INTABS ;3 - timer 2 IRQ
          .BYTE   VTIMR1-INTABS ;4 - timer 1 IRQ
          .BYTE   VSEROC-INTABS ;5 - serial output complete IRQ
          .BYTE   VSEROR-INTABS ;6 - serial output ready IRQ
          .BYTE   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   JSR      IVNM
*
*      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      RTCLOCK+2      ;increment low frame counter
;      BNE      IVN1           ;if low counter not zero
;
;      INC      ATTRACT        ;increment attract-mode counter/flag
;      INC      RTCLOCK+1      ;increment middle frame counter
;      BNE      IVN1           ;if middle counter not zero
;
;      INC      RTCLOCK        ;increment high frame counter
;
;      Set attract-mode effects.
;
IVN1     LDA     #$FE          ;select no luminance change
;      LDX     #0              ;select no color shift
;      LDY     ATTRACT         ;attract-mode timer/flag
;      BPL      IVN2           ;if not attract-mode
;
;      STA     ATTRACT         ;ensure continued attract-mode
;      LDX     RTCLOCK+1      ;select color shift
;      LDA     #$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        ;if critical section
;      BNE     IVN4
;
;      Check for IRQ enabled.
;
;      TSX
;      LDA     $0104,X         ;stack pointer
;      AND     #$04            ;stacked P
;      BEQ     IVN5            ;I (IRQ disable) bit
;                                ;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
LDA     CDTMV3-4,X      ;offset to countdown timer
ORA     CDTMV3+1-4,X    ;countdown timer
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      ;if CTRL-1
      BEQ      IVN13

      CMP      #CNTLF1     ;if CTRL-F1
      BEQ      IVN13

      CMP      #CNTLF2     ;if CTRL-F2
      BEQ      IVN13

      CMP      #CNTLF4     ;if CTRL-F4
      BEQ      IVN13

      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
      STA      STICK3      ;simulate joystick 3 reading
VGC    ENDIF
      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

;      Process deferred VBLANK NMI.

      JMP      (VVBLKD)    ;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                    ;return to caller

DCT2     LDA     #$FF       ;indicate timer did not expire
          RTS                    ;return to caller
```

SVP - Set Vertical Blank Parameters

```
*      SVP sets countdown timers and VBLANK vectors.
*
*      ENTRY    JMP      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           ;return to caller
```

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 from interrupt
```

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
*                2. Initialize PDVS to zero.
*                  Y. T. JANG & V. WU 1984-02-22
*                3. Move PDVS initialization to IHW.
*                  Mike Barall 1984-06-08
```

```
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 to EMS, if memory bad
*                  Exits via CARTCS vector or DOSVEC vector
*
*      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           ;JMP instruction
      BNE      PRS2           ;if JMP 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    ENDIF

;      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 RESET vector

        LDA    #$60            ;$60 is the opcode for RTS
        STA    ACMVAR
;        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
        STX    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      !N 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
      STA      PTRIG4,X
      DEX
      BPL      PRS16        ;if not done
VGC    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      CHACTL           ;CHACTL (character control) shadow
      LDA      #high DCSORG      ;high domestic character set origin
      STA      CHBASE           ;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
      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
      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 to caller
```

```

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

```

```

THAV    .BYTE    PRINTR    ;printer device code
        .WORD    PRINTV    ;printer handler vector table

        .BYTE    CASSET    ;cassette device code
        .WORD    CASETV    ;cassette handler vector table

        .BYTE    SCREDT    ;editor device code
        .WORD    EDITRV    ;editor handler vector table

        .BYTE    DISPLY    ;screen device code
        .WORD    SCRENV    ;screen handler vector table

        .BYTE    KBD       ;keyboard device code
        .WORD    KEYBDV    ;keyboard handler vector table

THAVL    =        *-THAV    ;length

```

```

**      BMSG - Boot Error Message
*
*      NOTES
*      Problem: should use lower case in message.

```

```

BMSG    .BYTE    'Boot Error',EOL

```

```

**      Screen Editor Device Specification

```

```

SEDS    .BYTE    'E:',EOL

```

```

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

```

```

TIHV    .WORD    RIR        ;VDSLST - display list NMI vector
        .WORD    XIR        ;VPRCED - proceed line IRQ vector
        .WORD    XIR        ;VINTER - interrupt line IRQ vector
        .WORD    XIR        ;VBREAK - BRK instruction IRQ vector
        .WORD    KIR        ;VKEYBD - keyboard IRQ vector
        .WORD    IRIR       ;VSERIN - serial input ready IRQ vector
        .WORD    ORIR       ;VSEROR - serial output ready IRQ vector
        .WORD    OCIR       ;VSEROC - serial output complete IRQ vector
        .WORD    XIR        ;VTIMR1 - POKEY timer 1 IRQ vector
        .WORD    XIR        ;VTIMR2 - POKEY timer 2 IRQ vector
        .WORD    XIR        ;VTIMR4 - POKEY timer 4 IRQ vector
        .WORD    IIR        ;VIMIRQ - immediate IRQ vector
        .WORD    0          ;CDTMV1 - countdown timer 1 vector
        .WORD    0          ;CDTMV2 - countdown timer 2 vector
        .WORD    0          ;CDTMV3 - countdown timer 3 vector
        .WORD    0          ;CDTMV4 - countdown timer 4 vector
        .WORD    0          ;CDTMV5 - countdown timer 5 vector
        .WORD    IVNM       ;VVBLKI - immediate VBLANK NMI vector
        .WORD    DVNM       ;VVBLKD - deferred VBLANK NMI vector

TIHVL    =        *-TIHV    ;length

```

PMI - Perform Miscellaneous Initialization

```
*      ENTRY   JSR     PMI
*
*      MODS    Original Author Unknown  ???-??-??
*              1. Bring closer to Coding Standard (object unchanged).
*                 R. K. Nordin  1983-11-01
*              2. Fix initial address for RAM sizing.
*                 R. K. Nordin  1984-03-23
```

```
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    ;enable 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 to caller
RAMSYS ELSE

PMI4 LDA #low $4000 ;initial low address
TAY ;offset to first byte of page
STA TRAMSZ-1 ;set initial low address

LDA #high $4000 ;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 to caller
RAMSYS ENDIF

```

CCE - Check Cartridge Equivalence

```

* ENTRY JSR CCE
*
* MODS Original Author Unknown ???-??-??
* 1. Bring closer to Coding Standard (object unchanged).
* R. K. Nordin 1983-11-01
* 2. Fix initial address for equivalence checksum.
* R. K. Nordin 1984-03-23
*

```

```

CCE = * ;entry

; Initialize.

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

; Checksum 256 bytes of cartridge area.

CCE1 ADC $BF00,X ;add in byte
INX
BNE CCE1 ;if not done

; Exit.

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

```

IHW - Initialize Hardware

```
*      ENTRY   JSR      IHW
*
*      MODS    Original Author Unknown  ???-??-??
*              1. Bring closer to Coding Standard (object unchanged).
*                 R. K. Nordin 1983-11-01
*              2. Change initialization of PORTB.
*                 R. K. Nordin 1984-03-27
*              3. Initialize PDVS to zero.
*                 Mike Barall 1984-06-08
```

```
IHW      =      *      ;entry

;      Initialize CTIA, ANTIC, POKEY and PIA registers.

      LDA      #0          ;initialization value
      STA      PDVS        ;Select floating-point package
      TAX
      STA      PBCTL       ;initial offset
                           ;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      #$3C        ;precondition port B outputs
      STA      PBCTL
      LDA      #$CF
      STA      PORTB       ;initialize port B
      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

;      If coldstart, return

      LDA      WARMST
      BNE      IHW3        ;if warmstart
      RTS                 ;return to caller

;      If warmstart, exit through the RESET routine

IHW3     JMP      ACMVAR
```


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      BRKKEY              ;set BREAK key IRQ routine address
      LDA      #high BIR
      STA      BRKKEY+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      CASSETV+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 to caller
```

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

;      JMP      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 to caller
```

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 to caller
```

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      ICBAH,X      ;high buffer address
LDA      #PUTREC
STA      ICCOM,X      ;command
LDA      #$FF
STA      ICBLL,X      ;buffer length

;      Perform CI0.

JMP      CIOV      ;vector to CI0, 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      RBLOKV      ;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 DI0, 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      BOOT?   ;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 software initialization address
      LDA      DOSINI+1
      STA      CASINI+1

;      Exit.

ACB2     RTS           ;return to caller

;      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 DSCTSZ ;disk sector size
      STA DSCTLN
      LDA #high DSCTSZ
      STA DSCTLN+1
      RTS ;return to caller
```


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      #FOMAT      ;if FORMAT command
;      BEQ      DIO1
;
;      LDA      #7           ;set timeout to 7 seconds
DIO1     STA      DTIMLO      ;timeout
;
;      Set SIO command.
;
;      LDX      #GETDAT      ;assume GET DATA
;
;      LDA      DCOMND      ;command
;      CMP      #PUTSEC      ;if PUT SECTOR command
;      BEQ      DIO2
;
;      CMP      #WRITE      ;if not WRITE command
;      BNE      DIO3
DIO2     LDX      #PUTDAT      ;select PUT DATA
;
;      Check command.
DIO3     CMP      #STATC      ;if not STATUS command
;      BNE      DIO4
;
;      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      DIO5        ;perform SIO
;
;      Set up other commands.
DIO4     LDY      DSCTLN      ;low byte count
;      LDA      DSCTLN+1    ;high byte count
;
;      Perform SIO.
DIO5     STX      DSTATS      ;SIO command
;      STY      DBYTLO      ;low byte count
;      STA      DBYTHI      ;high byte count
;      JSR      SIOV        ;vector to SIO
;      BPL      DIO6        ;if no error
;
;      Process error.
;      RTS                ;return to caller
;
;      Process successful operation.
DIO6     LDA      DCOMND      ;command
;      CMP      #STATC      ;if not STATUS command
;      BNE      DIO7
;
;      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      #$FE      ;initial buffer pointer
DI08   INY
      INY
      ;increment buffer pointer
      ;increment buffer pointer
DI09   LDA      (BUFADR),Y
      CMP      #$FF
      BNE      DI08      ;if low not $FF
      INY
      LDA      (BUFADR),Y
      INY
      CMP      #$FF
      BNE      DI09      ;if high not $FF
      DEY
      DEY
      STY      DBYTLO
      LDA      #0
      STA      DBYTHI
      ;low bad sector byte count
      ;high bad sector byte count
;      Exit.
DI010  LDY      DSTATS
      RTS
      ;status
      ;return to caller

```

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    =      *
      LDA      DBUFLO
      STA      BUFADR
      LDA      DBUFHI
      STA      BUFADR+1
      RTS
      ;entry
      ;buffer address
      ;return to caller

```

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-??
*              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          ;LCOUNT=RECLN then get next new record
;      CMP      LCOUNT
;      BEQ      RLR2            ;if ???
;
;      JSR      GBY              ;get next byte
;      LDY      #DATAER
;      BCS      RLR4            ;if EOF before END record
;
;      JSR      CAL              ;call record subroutine
;      INC      LCOUNT
;      BNE      RLR3            ;???
;
RLR4     RTS                    ;return to caller
```

END - Handles record type of 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-??
*              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
          STA     RUNADR+1

END2     LDY     #SUCCES    ;Y=1 successful operation

END3     RTS              ;return to caller

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

```

.....
GBY - Jump to the address pointed to by value in GBYTEA

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

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

.....
CAL - Jump to the address pointed to by value in RUNADR

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

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

TEX - Text Record Subroutine handling several types of record

```

;      Text Record Subroutine 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.
;
;      There are 4 notes about this routine
;
;      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.
;
;      ENTRY   JSR      TEX
;
;      MODS
;
;      Y. M. Chen      1982-04-??
;      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                ;if ???Y=high byte, A=low byte

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            ;if ???
;      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
      BCC      TEX8            ;MEMTOP>HIUSED, OK
      BNE      TEX7            ;if ???
      CMP      MEMTOP
      BCC      TEX8            ;if ???
TEX7  PLA
      PLA
      LDY      #MEMERR          ;MEMTOP<=HIUSED then error
                                   ;do a force return to caller
                                   ;set memory insufficient flag
TEX8  RTS                      ;return to caller

```

FTX - Relocate text into memory

```

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

```

```

FTX    =      *      ;entry
      SEC
      PHA
      LDA      LCOUNT      ;A register has object text
      SBC      #2            ;LCOUNT counts 2 bytes of relative address
      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 - Word Subroutine handling several types of record

```
; Word Subroutine 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 |(RECLN)|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-??
* 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 to caller
```

LOO - Low byte and one byte subroutine handling several types of record

```
; Low byte and one byte subroutine 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     LOO
*
* MODS    Y. M. Chen    1982-04-??
*          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 to caller
```

HIG - High byte subroutine handling several types of record

```
; High byte subroutine 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-??
* 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 to caller
; 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	.WORD	TEX	;0 - non-zero page relocatable text
	.WORD	TEX	;1 - zero page relocatable text
	.WORD	LOO	;2 - non-zero page low byte to non-zero page
	.WORD	LOO	;3 - zero page low byte to non-zero page
	.WORD	LOO	;4 - non-zero page one byte to zero page
	.WORD	LOO	;5 - zero page one byte to zero page
	.WORD	WOR	;6 - non-zero page word to non-zero page
	.WORD	WOR	;7 - zero page word to non-zero page
	.WORD	HIG	;8 - non-zero page high byte to non-zero page
	.WORD	HIG	;9 - zero page high byte to non-zero page
	.WORD	TEX	;10 - absolute text
	.WORD	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    Original Author Unknown  ???-??-??
*              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	A. 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 to caller
```

PIO - Parallel Input/Output

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

```
PIO    =      *      ;entry
;      Initialize.

      LDA      #1
      STA      CRITIC ;indicate critical section
      LDA      DUNIT  ;device unit number
      PHA
      LDA      PDVMSK ;device selection mask
      BEQ      PIO2   ;if no device to select
;      For each device, pass request to device I/O routine.

      LDX      #TPDSL ;offset to first byte beyond table
PIO1    JSR      SNP    ;select next parallel device
      BEQ      PIO2   ;if no device selected

      TXA
      PHA
      JSR      PDIOV   ;perform parallel device I/O
      PLA
      TAX
      BCC      PIO1   ;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      PIO3   ;exit
;      Perform SIO.
PIO2    JSR      SIO    ;perform SIO
;      Exit.
PIO3    PLA
      STA      DUNIT  ;restore device unit number
      LDA      #0
      STA      CRITIC ;indicate non-critical section
      STY      DSTATS
      LDY      DSTATS ;status (re-establish N)
      RTS
```

PIR - Handle Parallel Device IRQ

```
*      ENTRY    JMP      PIR
*
*      EXIT      Exits via RTI
*
*      MODS      Original Author Unknown  ???-??-??
*                  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      TPD5-1,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 from interrupt
```

GOP - Perform Generic Parallel Device OPEN

```
*      ENTRY    JSR      GOP
*
*      MODS      Original Author Unknown  ???-??-??
*                  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      Original Author Unknown  ???-??-??
*                  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     Original Author Unknown  ????-??-??
*               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     Original Author Unknown  ????-??-??
*               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     Original Author Unknown  ????-??-??
*               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     Original Author Unknown  ????-??-??
*               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     Original Author Unknown  ???-??-??
*              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      ;return to caller

;        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      ;return to caller
```

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     Original Author Unknown  ???-??-??
*              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
*			
*	MODS	Original Author Unknown ???-??-??	
*		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
      PHA
      LDA      #1      ;save critical section status
      STA      CRITIC  ;indicate critical section
;      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
      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
          LDA      PPTMPA ;restore possible data byte
          STY      PPTMPX
          LDY      PPTMPX ;status (re-establish N)
          RTS          ;return to caller
;      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      .BYTE      $80      ;0 - device 7 (lowest priority)
          .BYTE      $40      ;1 - device 6
          .BYTE      $20      ;2 - device 5
          .BYTE      $10      ;3 - device 4
          .BYTE      $08      ;4 - device 3
          .BYTE      $04      ;5 - device 2
          .BYTE      $02      ;6 - device 1
          .BYTE      $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
*
* MODS     R. S. Scheiman   1982-04-??
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin    1983-11-01
```

```
PHL      =      *      ;entry
;
; Load peripheral handler.
LDX      ICIDNO      ;IOCB index
LDA      ICSPR+1,X
JSR      LPH          ;load peripheral handler
BCS      PHL1         ;if error
;
; Initialize peripheral handler.
CLC
JSR      PHC          ;indicate zero handler size
BCS      PHL1         ;initialize peripheral handler
;
; Find device handler.
LDX      ICIDNO      ;IOCB index
LDA      ICSPR,X
JSR      FDH          ;find device handler
BCS      PHL1         ;if not found
;
; Set handler ID.
LDX      ICIDNO      ;IOCB index
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	.BYTE	TXT0-TTXT	;0 - offset to "MEMORY TEST ROM" text
	.BYTE	TXT1-TTXT	;1 - offset to "RAM" text
	.BYTE	TXT2-TTXT	;2 - offset to "KEYBOARD TEST" text
	.BYTE	TXT3-TTXT	;3 - offset to "S P A C E B A R" text
	.BYTE	TXT4-TTXT	;4 - offset to "SH" text
	.BYTE	TXT5-TTXT	;5 - offset to "SH" text
	.BYTE	TXT6-TTXT	;6 - offset to "B S" text
	.BYTE	TXT7-TTXT	;7 - offset to keyboard text
	.BYTE	TXT8-TTXT	;8 - offset to control key text
	.BYTE	TXT9-TTXT	;9 - offset to "VOICE #" text

** TTXT - Table of Text Sequences

TTXT	=	*
------	---	---

** TXT0 - "MEMORY TEST ROM" Text

TXT0	.BYTE	\$00,\$00	
	.BYTE	\$2D,\$25,\$2D,\$2F,\$32,\$39	; "MEMORY"
	.BYTE	\$00	
	.BYTE	\$34,\$25,\$33,\$34	; "TEST"
	.BYTE	\$00,\$00,\$00	
	.BYTE	\$32,\$2F,\$2D	; "ROM"
TXT0L	=	*-TXT0	;length

** TXT1 - "RAM" Text

TXT1	.BYTE	\$32,\$21,\$2D	; "RAM"
TXT1L	=	*-TXT1	;length

** TXT2 - "KEYBOARD TEST" Text

TXT2	.BYTE	\$00,\$00	
	.BYTE	\$2B,\$25,\$39,\$22,\$2F,\$21,\$32,\$24	; "KEYBOARD"
	.BYTE	\$00	
	.BYTE	\$34,\$25,\$33,\$34	; "TEST"
	.BYTE	\$00,\$00,\$00	
	.BYTE	\$B2	
TXT2L	=	*-TXT2	;length

```

TX17      =      *

;      First Row (Function Keys)

      .BYTE      $91      ; "1"
      .BYTE      $00
      .BYTE      $92      ; "2"
      .BYTE      $00
      .BYTE      $93      ; "3"
      .BYTE      $00
      .BYTE      $94      ; "4"
      .BYTE      $00
      .BYTE      $A8      ; "H"
      .BYTE      $00
      .BYTE      $A1      ; "A"
      .BYTE      $00
      .BYTE      $A2      ; "B"
      .BYTE      $00,$00,$00

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

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

```

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

.BYTE	\$FF	
.BYTE	\$FF	
.BYTE	\$00	
.BYTE	\$31	; "Q"
.BYTE	\$00	
.BYTE	\$37	; "W"
.BYTE	\$00	
.BYTE	\$25	; "E"
.BYTE	\$00	
.BYTE	\$32	; "R"
.BYTE	\$00	
.BYTE	\$34	; "T"
.BYTE	\$00	
.BYTE	\$39	; "Y"
.BYTE	\$00	
.BYTE	\$35	; "U"
.BYTE	\$00	
.BYTE	\$29	; "I"
.BYTE	\$00	
.BYTE	\$2F	; "O"
.BYTE	\$00	
.BYTE	\$30	; "P"
.BYTE	\$00	
.BYTE	\$0D	; "-"
.BYTE	\$00	
.BYTE	\$1D	; "="
.BYTE	\$00	
.BYTE	\$B2	; "R"
.BYTE	\$B4	; "T"
.BYTE	\$00,\$00,\$00	

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

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

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

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

```
; Sixth Row (Space Bar)
```

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

```
TXT7L = *-TXT7 ;length
```


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

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

TXT3L    =        *-TXT3        ;length
```

**** TXT4 - "SH" Text**

```
TXT4    .BYTE    $00
         .BYTE    $33,$28        ; "SH"
         .BYTE    $00

TXT4L    =        *-TXT4        ;length
```

**** TXT5 - "SH" Text**

```
TXT5    =        TXT4

TXT5L    =        TXT4L        ;length
```

**** TXT6 - "B S" Text**

```
TXT6    .BYTE    $22            ; "B"
         .BYTE    $00
         .BYTE    $33            ; "S"

TXT6L    =        *-TXT6        ;length
```

**** TXT8 - Control Key**

```
TXT8    .BYTE    $00
         .BYTE    $5C
         .BYTE    $00

TXT8L    =        *-TXT8        ;length
```

**** TXT9 - "VOICE #" Text**

```
TXT9    .BYTE    $36,$2F,$29,$23,$25    ; "VOICE"
         .BYTE    $00
         .BYTE    $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-??
*			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 to caller
```

International Character Set

FIX

ICSORG

** International Character Set

.BYTE	\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00	;\$00 - space
.BYTE	\$00,\$18,\$18,\$18,\$18,\$00,\$18,\$00	;\$01 - !
.BYTE	\$00,\$66,\$66,\$66,\$66,\$00,\$00,\$00	;\$02 - "
.BYTE	\$00,\$66,\$FF,\$66,\$66,\$FF,\$66,\$00	;\$03 - #
.BYTE	\$18,\$3E,\$60,\$3C,\$06,\$7C,\$18,\$00	;\$04 - \$
.BYTE	\$00,\$66,\$6C,\$18,\$30,\$66,\$46,\$00	;\$05 - %
.BYTE	\$1C,\$36,\$1C,\$38,\$6F,\$66,\$3B,\$00	;\$06 - &
.BYTE	\$00,\$18,\$18,\$18,\$00,\$00,\$00,\$00	;\$07 - '
.BYTE	\$00,\$0E,\$1C,\$18,\$18,\$1C,\$0E,\$00	;\$08 - (
.BYTE	\$00,\$70,\$38,\$18,\$18,\$38,\$70,\$00	;\$09 -)
.BYTE	\$00,\$66,\$3C,\$FF,\$3C,\$66,\$00,\$00	;\$0A - asterisk
.BYTE	\$00,\$18,\$18,\$7E,\$18,\$18,\$00,\$00	;\$0B - plus
.BYTE	\$00,\$00,\$00,\$00,\$00,\$00,\$18,\$18,\$30	;\$0C - comma
.BYTE	\$00,\$00,\$00,\$7E,\$00,\$00,\$00,\$00	;\$0D - minus
.BYTE	\$00,\$00,\$00,\$00,\$00,\$18,\$18,\$00	;\$0E - period
.BYTE	\$00,\$06,\$0C,\$18,\$30,\$60,\$40,\$00	;\$0F - /
.BYTE	\$00,\$3C,\$66,\$6E,\$76,\$66,\$3C,\$00	;\$10 - 0
.BYTE	\$00,\$18,\$38,\$18,\$18,\$18,\$7E,\$00	;\$11 - 1
.BYTE	\$00,\$3C,\$66,\$0C,\$18,\$30,\$7E,\$00	;\$12 - 2
.BYTE	\$00,\$7E,\$0C,\$18,\$0C,\$66,\$3C,\$00	;\$13 - 3
.BYTE	\$00,\$0C,\$1C,\$3C,\$6C,\$7E,\$0C,\$00	;\$14 - 4
.BYTE	\$00,\$7E,\$60,\$7C,\$06,\$66,\$3C,\$00	;\$15 - 5
.BYTE	\$00,\$3C,\$60,\$7C,\$66,\$66,\$3C,\$00	;\$16 - 6
.BYTE	\$00,\$7E,\$06,\$0C,\$18,\$30,\$30,\$00	;\$17 - 7
.BYTE	\$00,\$3C,\$66,\$3C,\$66,\$66,\$3C,\$00	;\$18 - 8
.BYTE	\$00,\$3C,\$66,\$3E,\$06,\$0C,\$38,\$00	;\$19 - 9
.BYTE	\$00,\$00,\$18,\$18,\$00,\$18,\$18,\$00	;\$1A - colon
.BYTE	\$00,\$00,\$18,\$18,\$00,\$18,\$18,\$30	;\$1B - semicolon
.BYTE	\$06,\$0C,\$18,\$30,\$18,\$0C,\$06,\$00	;\$1C - <
.BYTE	\$00,\$00,\$7E,\$00,\$00,\$7E,\$00,\$00	;\$1D - =
.BYTE	\$60,\$30,\$18,\$0C,\$18,\$30,\$60,\$00	;\$1E - >
.BYTE	\$00,\$3C,\$66,\$0C,\$18,\$00,\$18,\$00	;\$1F - ?
.BYTE	\$00,\$3C,\$66,\$6E,\$6E,\$60,\$3E,\$00	;\$20 - @
.BYTE	\$00,\$18,\$3C,\$66,\$66,\$7E,\$66,\$00	;\$21 - A
.BYTE	\$00,\$7C,\$66,\$7C,\$66,\$66,\$7C,\$00	;\$22 - B
.BYTE	\$00,\$3C,\$66,\$60,\$60,\$66,\$3C,\$00	;\$23 - C
.BYTE	\$00,\$78,\$6C,\$66,\$66,\$6C,\$78,\$00	;\$24 - D
.BYTE	\$00,\$7E,\$60,\$7C,\$60,\$60,\$7E,\$00	;\$25 - E
.BYTE	\$00,\$7E,\$60,\$7C,\$60,\$60,\$60,\$00	;\$26 - F
.BYTE	\$00,\$3E,\$60,\$60,\$6E,\$66,\$3E,\$00	;\$27 - G
.BYTE	\$00,\$66,\$66,\$7E,\$66,\$66,\$66,\$00	;\$28 - H
.BYTE	\$00,\$7E,\$18,\$18,\$18,\$18,\$7E,\$00	;\$29 - I
.BYTE	\$00,\$06,\$06,\$06,\$06,\$66,\$3C,\$00	;\$2A - J
.BYTE	\$00,\$66,\$6C,\$78,\$78,\$6C,\$66,\$00	;\$2B - K
.BYTE	\$00,\$60,\$60,\$60,\$60,\$60,\$7E,\$00	;\$2C - L
.BYTE	\$00,\$63,\$77,\$7F,\$6B,\$63,\$63,\$00	;\$2D - M
.BYTE	\$00,\$66,\$76,\$7E,\$7E,\$6E,\$66,\$00	;\$2E - N
.BYTE	\$00,\$3C,\$66,\$66,\$66,\$66,\$3C,\$00	;\$2F - O
.BYTE	\$00,\$7C,\$66,\$66,\$7C,\$60,\$60,\$00	;\$30 - P
.BYTE	\$00,\$3C,\$66,\$66,\$66,\$6C,\$36,\$00	;\$31 - Q
.BYTE	\$00,\$7C,\$66,\$66,\$7C,\$6C,\$66,\$00	;\$32 - R
.BYTE	\$00,\$3C,\$60,\$3C,\$06,\$06,\$3C,\$00	;\$33 - S
.BYTE	\$00,\$7E,\$18,\$18,\$18,\$18,\$18,\$00	;\$34 - T
.BYTE	\$00,\$66,\$66,\$66,\$66,\$66,\$7E,\$00	;\$35 - U
.BYTE	\$00,\$66,\$66,\$66,\$66,\$3C,\$18,\$00	;\$36 - V
.BYTE	\$00,\$63,\$63,\$6B,\$7F,\$77,\$63,\$00	;\$37 - W
.BYTE	\$00,\$66,\$66,\$3C,\$3C,\$66,\$66,\$00	;\$38 - X
.BYTE	\$00,\$66,\$66,\$3C,\$18,\$18,\$18,\$00	;\$39 - Y
.BYTE	\$00,\$7E,\$0C,\$18,\$30,\$60,\$7E,\$00	;\$3A - Z
.BYTE	\$00,\$1E,\$18,\$18,\$18,\$18,\$1E,\$00	;\$3B - [
.BYTE	\$00,\$40,\$60,\$30,\$18,\$0C,\$06,\$00	;\$3C - \
.BYTE	\$00,\$78,\$18,\$18,\$18,\$18,\$78,\$00	;\$3D -]
.BYTE	\$00,\$08,\$1C,\$36,\$63,\$00,\$00,\$00	;\$3E - ^
.BYTE	\$00,\$00,\$00,\$00,\$00,\$00,\$FF,\$00	;\$3F - underline

.BYTE	\$0C,\$18,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$40 - acute accent a
.BYTE	\$30,\$18,\$00,\$66,\$66,\$66,\$3E,\$00	;\$41 - acute accent u
.BYTE	\$36,\$6C,\$00,\$76,\$76,\$7E,\$6E,\$00	;\$42 - tilde N
.BYTE	\$0C,\$18,\$7E,\$60,\$7C,\$60,\$7E,\$00	;\$43 - acute accent E
.BYTE	\$00,\$00,\$3C,\$60,\$60,\$3C,\$18,\$30	;\$44 - cedilla c
.BYTE	\$3C,\$66,\$00,\$3C,\$66,\$66,\$3C,\$00	;\$45 - circumflex o
.BYTE	\$30,\$18,\$00,\$3C,\$66,\$66,\$3C,\$00	;\$46 - grave accent o
.BYTE	\$30,\$18,\$00,\$38,\$18,\$18,\$3C,\$00	;\$47 - grave accent i
.BYTE	\$1C,\$30,\$30,\$78,\$30,\$30,\$7E,\$00	;\$48 - U.K. currency
.BYTE	\$00,\$66,\$00,\$38,\$18,\$18,\$3C,\$00	;\$49 - diaeresis i
.BYTE	\$00,\$66,\$00,\$66,\$66,\$66,\$3E,\$00	;\$4A - umlaut u
.BYTE	\$36,\$00,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$4B - umlaut a
.BYTE	\$66,\$00,\$3C,\$66,\$66,\$66,\$3C,\$00	;\$4C - umlaut O
.BYTE	\$0C,\$18,\$00,\$66,\$66,\$66,\$3E,\$00	;\$4D - grave accent u
.BYTE	\$0C,\$18,\$00,\$3C,\$66,\$66,\$3C,\$00	;\$4E - acute accent o
.BYTE	\$00,\$66,\$00,\$3C,\$66,\$66,\$3C,\$00	;\$4F - umlaut o
.BYTE	\$66,\$00,\$66,\$66,\$66,\$66,\$7E,\$00	;\$50 - umlaut U
.BYTE	\$3C,\$66,\$1C,\$06,\$3E,\$66,\$3E,\$00	;\$51 - circumflex a
.BYTE	\$3C,\$66,\$00,\$66,\$66,\$66,\$3E,\$00	;\$52 - circumflex u
.BYTE	\$3C,\$66,\$00,\$38,\$18,\$18,\$3C,\$00	;\$53 - circumflex i
.BYTE	\$0C,\$18,\$3C,\$66,\$7E,\$60,\$3C,\$00	;\$54 - acute accent e
.BYTE	\$30,\$18,\$3C,\$66,\$7E,\$60,\$3C,\$00	;\$55 - grave accent e
.BYTE	\$36,\$6C,\$00,\$7C,\$66,\$66,\$66,\$00	;\$56 - tilde n
.BYTE	\$3C,\$C3,\$3C,\$66,\$7E,\$60,\$3C,\$00	;\$57 - circumflex e
.BYTE	\$18,\$00,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$58 - ring a
.BYTE	\$30,\$18,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$59 - grave accent a
.BYTE	\$18,\$00,\$18,\$3C,\$66,\$7E,\$66,\$00	;\$5A - ring A
.BYTE	\$78,\$60,\$78,\$60,\$7E,\$18,\$1E,\$00	;\$5B - display escape
.BYTE	\$00,\$18,\$3C,\$7E,\$18,\$18,\$18,\$00	;\$5C - up arrow
.BYTE	\$00,\$18,\$18,\$18,\$7E,\$3C,\$18,\$00	;\$5D - down arrow
.BYTE	\$00,\$18,\$30,\$7E,\$30,\$18,\$00,\$00	;\$5E - left arrow
.BYTE	\$00,\$18,\$0C,\$7E,\$0C,\$18,\$00,\$00	;\$5F - right arrow
.BYTE	\$18,\$00,\$18,\$18,\$18,\$18,\$18,\$00	;\$60 - Spanish !
.BYTE	\$00,\$00,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$61 - a
.BYTE	\$00,\$60,\$60,\$7C,\$66,\$66,\$7C,\$00	;\$62 - b
.BYTE	\$00,\$00,\$3C,\$60,\$60,\$60,\$3C,\$00	;\$63 - c
.BYTE	\$00,\$06,\$06,\$3E,\$66,\$66,\$3E,\$00	;\$64 - d
.BYTE	\$00,\$00,\$3C,\$66,\$7E,\$60,\$3C,\$00	;\$65 - e
.BYTE	\$00,\$0E,\$18,\$3E,\$18,\$18,\$18,\$00	;\$66 - f
.BYTE	\$00,\$00,\$3E,\$66,\$66,\$3E,\$06,\$7C	;\$67 - g
.BYTE	\$00,\$60,\$60,\$7C,\$66,\$66,\$66,\$00	;\$68 - h
.BYTE	\$00,\$18,\$00,\$38,\$18,\$18,\$3C,\$00	;\$69 - i
.BYTE	\$00,\$06,\$00,\$06,\$06,\$06,\$06,\$3C	;\$6A - j
.BYTE	\$00,\$60,\$60,\$6C,\$78,\$6C,\$66,\$00	;\$6B - k
.BYTE	\$00,\$38,\$18,\$18,\$18,\$18,\$3C,\$00	;\$6C - l
.BYTE	\$00,\$00,\$66,\$7F,\$7F,\$6B,\$63,\$00	;\$6D - m
.BYTE	\$00,\$00,\$7C,\$66,\$66,\$66,\$66,\$00	;\$6E - n
.BYTE	\$00,\$00,\$3C,\$66,\$66,\$66,\$3C,\$00	;\$6F - o
.BYTE	\$00,\$00,\$7C,\$66,\$66,\$7C,\$60,\$60	;\$70 - p
.BYTE	\$00,\$00,\$3E,\$66,\$66,\$3E,\$06,\$06	;\$71 - q
.BYTE	\$00,\$00,\$7C,\$66,\$60,\$60,\$60,\$00	;\$72 - r
.BYTE	\$00,\$00,\$3E,\$60,\$3C,\$06,\$7C,\$00	;\$73 - s
.BYTE	\$00,\$18,\$7E,\$18,\$18,\$18,\$0E,\$00	;\$74 - t
.BYTE	\$00,\$00,\$66,\$66,\$66,\$66,\$3E,\$00	;\$75 - u
.BYTE	\$00,\$00,\$66,\$66,\$66,\$3C,\$18,\$00	;\$76 - v
.BYTE	\$00,\$00,\$63,\$6B,\$7F,\$3E,\$36,\$00	;\$77 - w
.BYTE	\$00,\$00,\$66,\$3C,\$18,\$3C,\$66,\$00	;\$78 - x
.BYTE	\$00,\$00,\$66,\$66,\$66,\$3E,\$0C,\$78	;\$79 - y
.BYTE	\$00,\$00,\$7E,\$0C,\$18,\$30,\$7E,\$00	;\$7A - z
.BYTE	\$66,\$66,\$18,\$3C,\$66,\$7E,\$66,\$00	;\$7B - umlaut A
.BYTE	\$18,\$18,\$18,\$18,\$18,\$18,\$18,\$18	;\$7C -
.BYTE	\$00,\$7E,\$78,\$7C,\$6E,\$66,\$06,\$00	;\$7D - display clear
.BYTE	\$08,\$18,\$38,\$78,\$38,\$18,\$08,\$00	;\$7E - display backspace
.BYTE	\$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      #$40            ;disable DLI
      STA      NMIIEN          ;NMI enable
      LDX      #0              ;main screen colors
      JSR      SUC             ;set up colors
      LDX      #low DISL1      ;display list for main screen
      LDY      #high DISL1
      JSR      SDL             ;set up display list
      LDA      #low PMD        ;process main screen DLI routine
      STA      VDSLST          ;display list NMI address
      LDA      #high PMD
      STA      VDSLST+1
      LDX      #3*4            ;main screen bold lines
      LDA      #$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      NMIIEN

;      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 to caller
```

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 to caller
```

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 from interrupt

;      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

;      ???every 4th time???

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  .BYTE     $70,$70,$70,$70,$70
      .BYTE     $47
      .WORD     SMEM1
      .BYTE     $70,$70,$70
      .BYTE     $4E
      .WORD     ST3000
      .BYTE     $70
      .BYTE     $F0
      .BYTE     $C6
      .WORD     SMEM2
      .BYTE     $70,$86
      .BYTE     $70,$86
      .BYTE     $70,$06
      .BYTE     $70,$70
      .BYTE     $4E
      .WORD     ST3000
      .BYTE     $70,$70,$70
      .BYTE     $42
      .WORD     SMEM3
      .BYTE     $41
      .WORD     DISL1

```

** SMEM1 - "SELF TEST" Text

```

SMEM1  .BYTE     $00,$00,$00,$00
      .BYTE     $33,$25,$2C,$26      ;"SELF"
      .BYTE     $00
      .BYTE     $34,$25,$33,$34      ;"TEST"
      .BYTE     $00,$00,$00

```

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

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

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

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

**** DISL2 - Display List for Memory Test**

```
DISL2 .BYTE $70,$70,$70
       .BYTE $46
       .WORD ST3000
       .BYTE $70
       .BYTE $70,$06
       .BYTE $70,$08
       .BYTE $70
       .BYTE $70,$06
       .BYTE $70,$08
       .BYTE $70,$08
       .BYTE $70,$08
       .BYTE $70,$08
       .BYTE $70,$70,$70
       .BYTE $01
       .WORD DISL3
```

**** DISL3 - Display List for Exit Text**

```
DISL3 .BYTE $A0,$40
       .BYTE $42
       .WORD SMEM4
       .BYTE $01
       .WORD STJMP
```

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

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

**** DISL4 - Display List for Keyboard Test**

```
DISL4 .BYTE $70,$70,$70,$70
       .BYTE $46
       .WORD ST3000
       .BYTE $70,$70,$70
       .BYTE $70,$02
       .BYTE $70
       .BYTE $70,$02
       .BYTE $70,$02
       .BYTE $70,$02
       .BYTE $70,$02
       .BYTE $70,$02
       .BYTE $70,$70
       .BYTE $01
       .WORD DISL3
```

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

```
DISL5 .BYTE $70,$70,$70,$70
       .BYTE $46
       .WORD SMEM5
       .BYTE $70,$06
       .BYTE $70,$70
       .BYTE $4B
       .WORD ST3100
       .BYTE $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
       .BYTE $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
       .BYTE $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
       .BYTE $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
       .BYTE $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
       .BYTE $0B,$0B
       .BYTE $70
       .BYTE $46
       .WORD ST3000
       .BYTE $70
       .BYTE $01
       .WORD DISL3
```

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

```
SMEM5 .BYTE $00,$00
       .BYTE $21,$35,$24,$29,$2F ;"AUDIO"
       .BYTE $0D ;"_"
       .BYTE $36,$29,$33,$35,$21,$2C ;"VISUAL"
       .BYTE $00,$00,$00,$00
       .BYTE $00,$00,$00,$00
       .BYTE $34,$25,$33,$34 ;"TEST"
       .BYTE $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      STLM           ;LED mask
      LDA      #0
      STA      STSMP          ;initialize ???
      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      STLM      ;LED mask
        JSR      SLD       ;set LED's
        LDA      STLM      ;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      #$CC          ;color 2 for success
      JSR      DRS          ;display RAM block status
;      ???
STM17  LDA      STSMM         ;???
      BMI      STM20         ;if ???
      LDA      #$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 to caller

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      ;return to caller
```

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      ;return to caller
```

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
          JMP      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 to caller
```

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 to caller
```



```
*
*   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 ???.
;
;   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      HELPPFG     ;HELP key flag
;   BEQ      POD1        ;if HELP not pressed
;
;   Process HELP key.
;
;   LDA      #0         ;HELP key not pressed indicator
;   STA      HELPPFG     ;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 from interrupt

```

```

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

```

```

TMNT   .BYTE   high $0000    ;$0000 - $03FF, zero page and stack
      .BYTE   high $5000    ;$5000 - $53FF, self-test ROM
      .BYTE   high $5400    ;$5400 - $57FF, self-test ROM
      .BYTE   high ST3000    ;ST3000 - ST3000+$03FF, screen memory
      .BYTE   high ST3000    ;ST3000 - ST3000+$03FF, screen memory
      .BYTE   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      #$C0            ;if ???
;      BCS      STK3
;
;      LDA      CH              ;key code
;
;      Process keypress.
;
STK4     LDX      #$FF            ;clear code indicator
;      STX      CH              ;key code
;      PHA                        ;save key code
;      AND      #$80            ;if not CTRL
;      BEQ      STK5
;
;      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                ;key code
        LDA    TSMC,X      ;display character
        PHA                ;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  ;update ???

;      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
        DAW          ;delay a while
        JSR    SAS          ;silence all sounds
        LDX    #10          ;10-VBLANK delay
        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        ;if ???

        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 to caller
```

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 to caller
```

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      .BYTE      $52,$08,$0A,$2B,$28,$0D,$3D,$39,$2D      ;"Copyright"
          .BYTE      $1F,$30,$35,$18      ;"1984"
          .BYTE      $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
;      JSR      SDL              ;indicate not keyboard self-test
;                                  ;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      ST3180,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              ;display ???
;      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      ;first note ???
LDY    #high ST3154
LDA     #0*6
JSR     DVN              ;display ???

LDA     #$51             ;first note frequency
JSR     SVN              ;sound ???

;      Display and play second note.

LDX     #low ST3186      ;second note ???
LDY     #high ST3186
LDA     #0*6
JSR     DVN              ;display ???

LDA     #$5B             ;second note frequency
JSR     SVN              ;sound ???

;      Display and play third note.

LDX     #low ST30F8      ;third note ???
LDY     #high ST30F8
LDA     #12*6
JSR     DVN              ;display ???
LDX     #low ST30C7      ;third note ???
LDY     #high ST30C7
LDA     #14*6
JSR     DVN              ;display ???
LDX     #low ST3248      ;third note ???
LDY     #high ST3248
LDA     #13*6
JSR     DVN              ;display ???

LDA     #$44             ;third note frequency
JSR     SVN              ;sound ???

;      Display and play fourth note.

LDX     #low ST30CA      ;fourth note ???
LDY     #high ST30CA
LDA     #12*6
JSR     DVN              ;display ???
LDX     #low ST321A      ;fourth note ???
LDY     #high ST321A
LDA     #13*6
JSR     DVN              ;display ???
LDX     #low ST31CA      ;fourth note ???
LDY     #high ST31CA
LDA     #1*6
JSR     DVN              ;display ???

LDA     #$3C             ;fourth note frequency
JSR     SVN              ;sound ???

;      Display and play fifth note.

LDX     #low ST303C      ;fifth note ???
LDY     #high ST303C
LDA     #12*6
JSR     DVN              ;display ???
LDX     #low ST318C      ;fifth note ???
LDY     #high ST318C
LDA     #13*6
JSR     DVN              ;display ???
LDX     #low ST313C      ;fifth note ???
LDY     #high ST313C
LDA     #1*6
JSR     DVN              ;display ???

LDA     #$2D             ;fifth note frequency
JSR     SVN              ;sound ???

```



```

;      Display and play sixth note.

LDX    #low ST309E      ;sixth note ???
LDY    #high ST309E
LDA    #12*6
JSR    DVN              ;display ???
LDX    #low ST31EE      ;sixth note ???
LDY    #high ST31EE
LDA    #13*6
JSR    DVN              ;display ???

LDA    #$35             ;sixth note frequency
JSR    SVN              ;sound ???

;      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
        JMP     STK       ;self-test keyboard

```

SVN - Sound Note

*	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                    ;return to caller

```

..... 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		;offset to ???
	LDY	#0	
	LDA	#16	;???
	STA	STTMP3	;???
	LDA	#6	;???
	STA	STTMP4	;???
DVN1	LDA	TAVD,X	;byte of ???
	ORA	(STTMP2),Y	;???
	STA	(STTMP2),Y	;update ???
	JSR	AST	;add 16
	DEC	STTMP3	;???
	BNE	DVN1	;if ???
	INC	STTMP3	;???
	INX		
	DEC	STTMP4	
	BNE	DVN1	;if ???
	RTS		;return to caller

..... 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 to caller

** TNDD - Table of Note Duration Delays

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

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

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

**** TCDA - Table of Clef Display Addresses**

TCDA	.WORD	ST30C1	;0 - ???
	.WORD	ST3121	;1 - ???
	.WORD	ST3181	;2 - ???
	.WORD	ST31F1	;3 - ???
	.WORD	ST3002	;4 - ???
	.WORD	ST3062	;5 - ???
	.WORD	ST3122	;6 - ???
	.WORD	ST3182	;7 - ???
	.WORD	ST30C2	;8 - ???
	.WORD	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 to caller
```

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      TTXT,Y      ;byte of source
          STA      ST3000,X    ;byte of destination
          INY
          INX
          DEC      STADR1      ;decrement length
          BNE      SSM1        ;if not done
          RTS                  ;return to caller
```

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 to caller

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

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

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

SUCD     .BYTE     $D2          ;0 - main screen background color
          .BYTE     $A0          ;1 - memory test background color
          .BYTE     $30          ;2 - keyboard test background color
          .BYTE     $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      .BYTE      $2C      ;$00 - L key
          .BYTE      $2A      ;$01 - J key
          .BYTE      $1B      ;$02 - semicolon key
          .BYTE      $91      ;$03 - ???
          .BYTE      $92      ;$04 - ???
          .BYTE      $2B      ;$05 - K key
          .BYTE      $0B      ;$06 - plus key
          .BYTE      $0A      ;$07 - asterisk key
          .BYTE      $2F      ;$08 - 0 key
          .BYTE      $00      ;$09
          .BYTE      $30      ;$0A - P key
          .BYTE      $35      ;$0B - U key
          .BYTE      $B2      ;$0C - RETURN key
          .BYTE      $29      ;$0D - I key
          .BYTE      $0D      ;$0E - minus key
          .BYTE      $1D      ;$0F - = key

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

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

          .BYTE      $19      ;$30 - 9 key
          .BYTE      $00      ;$31
          .BYTE      $10      ;$32 - 0 key
          .BYTE      $17      ;$33 - 7 key
          .BYTE      $A2      ;$34 - backspace key
          .BYTE      $18      ;$35 - 8 key
          .BYTE      $1C      ;$36 - < key
          .BYTE      $1E      ;$37 - > key
          .BYTE      $26      ;$38 - F key
          .BYTE      $28      ;$39 - H key
          .BYTE      $24      ;$3A - D key
          .BYTE      $00      ;$3B
          .BYTE      $A3      ;$3C - CAPS key
          .BYTE      $27      ;$3D - G key
          .BYTE      $33      ;$3E - S key
          .BYTE      $21      ;$3F - A key

```

**** TARS - Table of Address Ranges to Set**

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

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

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

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

TSTD	.BYTE	ST3000-ST3000	;0 - offset to "MEMORY TEST ROM" text
	.BYTE	ST3028-ST3000	;1 - offset to "RAM" text
	.BYTE	ST3000-ST3000	;2 - offset to "KEYBOARD TEST" text
	.BYTE	ST30B7-ST3000	;3 - offset to "S P A C E B A R" text
	.BYTE	ST3092-ST3000	;4 - offset to "SH" text
	.BYTE	ST30AB-ST3000	;5 - offset to "SH" text
	.BYTE	ST304C-ST3000	;6 - offset to "B S" text
	.BYTE	ST3022-ST3000	;7 - offset to keyboard text
	.BYTE	ST3072-ST3000	;8 - offset to control key text
	.BYTE	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 1978-??-??
*
* 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      ;zero ???
      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
      LDX      FR0M      ;save digit
                        ;first byte
      BNE      AFP3      ;if not zero

;      ???

      JSR      S0L      ;shift FR0 left 1 digit
      PLA      ;saved digit
      ORA      FR0M+FMPPREC-1 ;insert into last byte
      STA      FR0M+FMPPREC-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
      STX      DIGRT      ;increment number of digits
                        ;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 to caller
;      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
      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
      LDA          EEXP      ;first character of exponent
      PHA          EEXP      ;number of digits more than 9
      STX          EEXP      ;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          EEXP      ;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
      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      #'+'      ;if "+", process next character
      BEQ      AFP14
      CMP      #'-'      ;if not "-", ???
      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                ;save adjusted exponent
      ROL      A          ;set C with sign of exponent
      PLA                ;saved adjusted exponent
      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 to caller

      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
;
;      Process E format required.
FASC3  LDA      #1       ;GET DECIMAL POSITION???
      JSR      C0A      ;convert FR0 to ASCII
      JSR      FNZ      ;find last non-zero character
      INX
      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
      BNE      FASC8    ;increment number of 10's
                        ;continue

FASC9  CLC
      ADC      #10      ;add back 10
      PHA
      TXA
      JSR      SNL      ;store number in line buffer
      PLA
      ORA      #$80     ;set high order bit
      JSR      SNL      ;store number in line buffer

;      Perform final adjustment.

FASC10 LDA      LBUFF      ;first character
      CMP      #'0'
      BNE      FASC11    ;if not "0", ???

;      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 to caller
```

```
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        ;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      SIL      ;shift integer left
;      BCS      FPI4      ;if number too big, error
;
;      LDA      ZTEMP4      ;2 times integer
;      STA      ZTEMP3      ;save 2 times integer
;      LDA      ZTEMP4+1
;      STA      ZTEMP3+1
;      JSR      SIL      ;shift integer left
;      BCS      FPI4      ;if number too big, error
;
;      JSR      SIL      ;shift integer left
;      BCS      FPI4      ;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 in ???
;      STA      ZTEMP4+1      ;update ???
;      LDA      ZTEMP4      ;???
;      ADC      #0      ;add carry
;      BCS      FPI4      ;if overflow, error
;
;      STA      ZTEMP4      ;update ???
;      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      ;update ???
      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 to caller

;      Return error.

FPI4   SEC                    ;indicate error
      RTS                    ;return to caller


      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
;      JMP      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
;      JMP      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 to caller
```

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                      ;return to caller
```

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                      ;return to caller
```

```
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
;      SWP      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      FROM,X       ;byte of augend mantissa
      SBC      FR1M,X       ;subtract byte of addend mantissa
      STA      FROM,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      FROM,X       ;update byte of result mantissa
      STA      FROM,X
      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
;      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 to caller

;      Return error.
FMUL9  SEC          ;indicate error
        RTS        ;return to caller


        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          ;???, return

```

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
      STY      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 to caller

```

SLB - Skip Leading Blanks

```
*      ENTRY   JSR      ???
*              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
```

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
```

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      #'.'      ;if ".", check next character
      BEQ      TVN4
      CMP      #'+'      ;if "+", check next character
      BEQ      TVN3
      CMP      #'-'      ;if "-", check next character
      BEQ      TVN3
;      Clean stack.
TVN1   PLA          ;clean stack
;      Return failure.
TVN2   SEC          ;indicate failure
      RTS          ;return to caller
;      Check character after "+" or "-".
TVN3   JSR      GNC      ;get next character
      BCC      TVN5     ;if numeric, return success
      CMP      #'.'      ;if not ".", return failure
      BNE      TVN1
;      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 to caller
```

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
 JNE 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 to caller

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 ;byte to shift in
 STX FRE ;normalize FR0/FRE, return
 JNE NOE

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 to caller

;      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 to caller

;      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 to caller
```

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     ;???, 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
;      JMP     SRR     ;???, 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
```

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 to caller
```

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 FROM
        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 to caller
```

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
;          AND      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 to caller
```

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
      LDA      LBUF,X ;character
;      Exit.
FNZ3     RTS            ;return to caller
```

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 to caller
```

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 to caller
```

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              ;return to caller
```

SUP - Set Up for Multiply or Divide

```
*      ENTRY   JSR      SUP
*              A = exponent
*              CC - SET BY ADD OR SUB TO GET A??????
*
*      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              ;return to caller
```

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
;      BNE    F1R          ;???, 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
;      JMP F2R ;???, return
```

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
;      JMP FARR ;???, return
```

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 to caller
```

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 to caller
```

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 to caller

          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
*
*      CHANGES  USES FPTR2, PLYCNT, PLYARG
*
*      CALLS    CALLS FST0R, FMOVE, FLD1R, FADD, FMUL
*
*      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 to caller

      FIX      FLD0R

```

FLD0R - ???

```
*      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      ;???, return
```

FIX FLD0P

FLD0P - ???

```
*      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 to caller
```

FIX FLD1R

FLD1R - ???

```
*      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
;          JMP    FLD1P      ;???, return
```

FIX FLD1P

FLD1P - ???

```
*      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
      FIX     FST0R
```

FST0R - ???

```
*      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
;      JMP     FST0P    ;???, return
      FIX     FST0P
```

FST0P - ???

```
*      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
      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 - ???

```
*      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         ;???, return

      FIX      EXP10
```


EXP10 - ???

```

*      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
      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      #NPCOEF
      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         ;???
      LDY     #low FONE
      LDY     #high FONE
      JSR     FLD0R         ;???
      JSR     FDIV          ;divide to get result

;      Exit.

EXP5   RTS                 ;return to caller

;      Return error.

EXP6   SEC                 ;indicate error
      RTS                 ;return to caller

```

** P10COF - Used in EXP10 calculation

```

P10COF .BYTE   $3D,$17,$94,$19,$00,$00 ;0.0000179419
      .BYTE   $3D,$57,$33,$05,$00,$00 ;0.0000573305
      .BYTE   $3E,$05,$54,$76,$62,$00 ;0.0005547662
      .BYTE   $3E,$32,$19,$62,$27,$00 ;0.0032176227
      .BYTE   $3F,$01,$68,$60,$30,$36 ;0.0168603036
      .BYTE   $3F,$07,$32,$03,$27,$41 ;0.0732032741
      .BYTE   $3F,$25,$43,$34,$56,$75 ;0.2543345675
      .BYTE   $3F,$66,$27,$37,$30,$50 ;0.6627373050
      .BYTE   $40,$01,$15,$12,$92,$55 ;1.15129255
      .BYTE   $3F,$99,$99,$99,$99,$99 ;0.9999999999

NPCOEF =      [*-P10COF]/FPREC

```

** LOG10E - Base 10 Logarithm of e

```

LOG10E .BYTE   $3F,$43,$42,$94,$48,$19 ;base 10 logarithm of e

```

** FONE - 1.0

```

FONE   .BYTE   $40,$01,$00,$00,$00,$00 ;1.0

```

XFORM - 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 ;???
JSR    FADD ;X+C
LDX    #low FPSCR
LDY    #high FPSCR
JSR    FST0R ;???
LDX    #low PLYARG
LDY    #high PLYARG
JSR    FLD0R ;???
LDX    FPTR2
LDY    FPTR2+1
JSR    FLD1R ;???
JSR    FSUB ;X-C
LDX    #low FPSCR
LDY    #high FPSCR
JSR    FLD1R ;???
JSR    FDIV ;divide to get result
RTS    ;return to caller

FIX    LOG
```

LOG - Indicate 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 - Indicate 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
; JNE    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
;
;      Return error.
LOGS1     SEC           ;indicate error
;      RTS            ;return to caller
```

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          ;if ???

      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      FSTOR            ;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            ;???
      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          ;if ???
      EOR      #-$01           ;complement sign???
      CLC
      ADC      #1
      STA      FR0

LOGC5   JSR      IFP            ;convert integer to floating point
      BIT      XFMFLG
      BPL      LOGC6          ;if ???

      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           ;???
      JSR      FDIV           ;result is LOG(X) divided by LOG10(e)

;      Exit.

LOGC7  CLC                  ;indicate success
      RTS                   ;return to caller

```

** SQR10 - Square Root of 10

```

SQR10  .BYTE    $40,$03,$16,$22,$77,$66    ;square root of 10

```

** FHALF - 0.5

```

FHALF  .BYTE    $3F,$50,$00,$00,$00,$00    ;0.5

```

** LGCOEF - Logarithm Coefficients

```

LGCOEF  .BYTE    $3F,$49,$15,$57,$11,$08    ;0.4915571108
        .BYTE    $BF,$51,$70,$49,$47,$08    ;-0.5170494708
        .BYTE    $3F,$39,$20,$57,$61,$95    ;0.3920576195
        .BYTE    $BF,$04,$39,$63,$03,$55    ;-0.0439630355
        .BYTE    $3F,$10,$09,$30,$12,$64    ;0.1009301264
        .BYTE    $3F,$09,$39,$08,$04,$60    ;0.0939080460
        .BYTE    $3F,$12,$42,$58,$47,$42    ;0.1242584742
        .BYTE    $3F,$17,$37,$12,$06,$08    ;0.1737120608
        .BYTE    $3F,$28,$95,$29,$71,$17    ;0.2895297117
        .BYTE    $3F,$86,$85,$88,$96,$44    ;0.8685889644

```

```

NLCOEF  =        [*-LGCOEF]/FPREC

```

** ATCOEF - Arctangent Coefficients

```

*
* NOTES  Problem: not used.

```

```

        .BYTE    $3E,$16,$05,$44,$49,$00    ;0.001605444900
        .BYTE    $BE,$95,$68,$38,$45,$00    ;-0.009568384500
        .BYTE    $3F,$02,$68,$79,$94,$16    ;0.0268799416
        .BYTE    $BF,$04,$92,$78,$90,$80    ;-0.0492789080
        .BYTE    $3F,$07,$03,$15,$20,$00    ;0.0703152000
        .BYTE    $BF,$08,$92,$29,$12,$44    ;-0.0892291244
        .BYTE    $3F,$11,$08,$40,$09,$11    ;0.1108400911
        .BYTE    $BF,$14,$28,$31,$56,$04    ;-0.1428315604
        .BYTE    $3F,$19,$99,$98,$77,$44    ;0.1999987744
        .BYTE    $BF,$33,$33,$33,$31,$13    ;-0.3333333113
        .BYTE    $3F,$99,$99,$99,$99,$99    ;0.9999999999

        .BYTE    $3F,$78,$53,$98,$16,$34    ;pi/4 = arctan 1

```

LOGQ - ???

*	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    ;???, return
```

Domestic Character Set

FIX

DCSORG

** Domestic Character Set

.BYTE	\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00	;\$00 - space
.BYTE	\$00,\$18,\$18,\$18,\$18,\$00,\$18,\$00	;\$01 - !
.BYTE	\$00,\$66,\$66,\$66,\$66,\$00,\$00,\$00	;\$02 - "
.BYTE	\$00,\$66,\$FF,\$66,\$66,\$FF,\$66,\$00	;\$03 - #
.BYTE	\$18,\$3E,\$60,\$3C,\$06,\$7C,\$18,\$00	;\$04 - \$
.BYTE	\$00,\$66,\$6C,\$18,\$30,\$66,\$46,\$00	;\$05 - %
.BYTE	\$1C,\$36,\$1C,\$38,\$6F,\$66,\$3B,\$00	;\$06 - &
.BYTE	\$00,\$18,\$18,\$18,\$00,\$00,\$00,\$00	;\$07 - '
.BYTE	\$00,\$0E,\$1C,\$18,\$18,\$1C,\$0E,\$00	;\$08 - (
.BYTE	\$00,\$70,\$38,\$18,\$18,\$38,\$70,\$00	;\$09 -)
.BYTE	\$00,\$66,\$3C,\$FF,\$3C,\$66,\$00,\$00	;\$0A - asterisk
.BYTE	\$00,\$18,\$18,\$7E,\$18,\$18,\$00,\$00	;\$0B - plus
.BYTE	\$00,\$00,\$00,\$00,\$00,\$00,\$18,\$18	;\$0C - comma
.BYTE	\$00,\$00,\$00,\$7E,\$00,\$00,\$00,\$00	;\$0D - minus
.BYTE	\$00,\$00,\$00,\$00,\$00,\$18,\$18,\$00	;\$0E - period
.BYTE	\$00,\$06,\$0C,\$18,\$30,\$60,\$40,\$00	;\$0F - /
.BYTE	\$00,\$3C,\$66,\$6E,\$76,\$66,\$3C,\$00	;\$10 - 0
.BYTE	\$00,\$18,\$38,\$18,\$18,\$18,\$7E,\$00	;\$11 - 1
.BYTE	\$00,\$3C,\$66,\$0C,\$18,\$30,\$7E,\$00	;\$12 - 2
.BYTE	\$00,\$7E,\$0C,\$18,\$0C,\$66,\$3C,\$00	;\$13 - 3
.BYTE	\$00,\$0C,\$1C,\$3C,\$6C,\$7E,\$0C,\$00	;\$14 - 4
.BYTE	\$00,\$7E,\$60,\$7C,\$06,\$66,\$3C,\$00	;\$15 - 5
.BYTE	\$00,\$3C,\$60,\$7C,\$66,\$66,\$3C,\$00	;\$16 - 6
.BYTE	\$00,\$7E,\$06,\$0C,\$18,\$30,\$30,\$00	;\$17 - 7
.BYTE	\$00,\$3C,\$66,\$3C,\$66,\$66,\$3C,\$00	;\$18 - 8
.BYTE	\$00,\$3C,\$66,\$3E,\$06,\$0C,\$38,\$00	;\$19 - 9
.BYTE	\$00,\$00,\$18,\$18,\$00,\$18,\$18,\$00	;\$1A - colon
.BYTE	\$00,\$00,\$18,\$18,\$00,\$18,\$18,\$30	;\$1B - semicolon
.BYTE	\$06,\$0C,\$18,\$30,\$18,\$0C,\$06,\$00	;\$1C - <
.BYTE	\$00,\$00,\$7E,\$00,\$00,\$7E,\$00,\$00	;\$1D - =
.BYTE	\$60,\$30,\$18,\$0C,\$18,\$30,\$60,\$00	;\$1E - >
.BYTE	\$00,\$3C,\$66,\$0C,\$18,\$00,\$18,\$00	;\$1F - ?
.BYTE	\$00,\$3C,\$66,\$6E,\$6E,\$60,\$3E,\$00	;\$20 - @
.BYTE	\$00,\$18,\$3C,\$66,\$66,\$7E,\$66,\$00	;\$21 - A
.BYTE	\$00,\$7C,\$66,\$7C,\$66,\$66,\$7C,\$00	;\$22 - B
.BYTE	\$00,\$3C,\$66,\$60,\$60,\$66,\$3C,\$00	;\$23 - C
.BYTE	\$00,\$78,\$6C,\$66,\$66,\$6C,\$78,\$00	;\$24 - D
.BYTE	\$00,\$7E,\$60,\$7C,\$60,\$60,\$7E,\$00	;\$25 - E
.BYTE	\$00,\$7E,\$60,\$7C,\$60,\$60,\$60,\$00	;\$26 - F
.BYTE	\$00,\$3E,\$60,\$60,\$6E,\$66,\$3E,\$00	;\$27 - G
.BYTE	\$00,\$66,\$66,\$7E,\$66,\$66,\$66,\$00	;\$28 - H
.BYTE	\$00,\$7E,\$18,\$18,\$18,\$18,\$7E,\$00	;\$29 - I
.BYTE	\$00,\$06,\$06,\$06,\$06,\$66,\$3C,\$00	;\$2A - J
.BYTE	\$00,\$66,\$6C,\$78,\$78,\$6C,\$66,\$00	;\$2B - K
.BYTE	\$00,\$60,\$60,\$60,\$60,\$60,\$7E,\$00	;\$2C - L
.BYTE	\$00,\$63,\$77,\$7F,\$6B,\$63,\$63,\$00	;\$2D - M
.BYTE	\$00,\$66,\$76,\$7E,\$7E,\$6E,\$66,\$00	;\$2E - N
.BYTE	\$00,\$3C,\$66,\$66,\$66,\$66,\$3C,\$00	;\$2F - O
.BYTE	\$00,\$7C,\$66,\$66,\$7C,\$60,\$60,\$00	;\$30 - P
.BYTE	\$00,\$3C,\$66,\$66,\$66,\$6C,\$36,\$00	;\$31 - Q
.BYTE	\$00,\$7C,\$66,\$66,\$7C,\$6C,\$66,\$00	;\$32 - R
.BYTE	\$00,\$3C,\$60,\$3C,\$06,\$06,\$3C,\$00	;\$33 - S
.BYTE	\$00,\$7E,\$18,\$18,\$18,\$18,\$18,\$00	;\$34 - T
.BYTE	\$00,\$66,\$66,\$66,\$66,\$66,\$7E,\$00	;\$35 - U
.BYTE	\$00,\$66,\$66,\$66,\$66,\$3C,\$18,\$00	;\$36 - V
.BYTE	\$00,\$63,\$63,\$6B,\$7F,\$77,\$63,\$00	;\$37 - W
.BYTE	\$00,\$66,\$66,\$3C,\$3C,\$66,\$66,\$00	;\$38 - X
.BYTE	\$00,\$66,\$66,\$3C,\$18,\$18,\$18,\$00	;\$39 - Y
.BYTE	\$00,\$7E,\$0C,\$18,\$30,\$60,\$7E,\$00	;\$3A - Z
.BYTE	\$00,\$1E,\$18,\$18,\$18,\$18,\$1E,\$00	;\$3B - [
.BYTE	\$00,\$40,\$60,\$30,\$18,\$0C,\$06,\$00	;\$3C - \
.BYTE	\$00,\$78,\$18,\$18,\$18,\$18,\$78,\$00	;\$3D -]
.BYTE	\$00,\$08,\$1C,\$36,\$63,\$00,\$00,\$00	;\$3E - ^
.BYTE	\$00,\$00,\$00,\$00,\$00,\$00,\$FF,\$00	;\$3F - underline

.BYTE	\$00,\$36,\$7F,\$7F,\$3E,\$1C,\$08,\$00	;\$40 - heart card
.BYTE	\$18,\$18,\$18,\$1F,\$1F,\$18,\$18,\$18	;\$41 - mid left window
.BYTE	\$03,\$03,\$03,\$03,\$03,\$03,\$03,\$03	;\$42 - right box
.BYTE	\$18,\$18,\$18,\$F8,\$F8,\$00,\$00,\$00	;\$43 - low right window
.BYTE	\$18,\$18,\$18,\$F8,\$F8,\$18,\$18,\$18	;\$44 - mid right window
.BYTE	\$00,\$00,\$00,\$F8,\$F8,\$18,\$18,\$18	;\$45 - up right window
.BYTE	\$03,\$07,\$0E,\$1C,\$38,\$70,\$E0,\$C0	;\$46 - right slant box
.BYTE	\$C0,\$E0,\$70,\$38,\$1C,\$0E,\$07,\$03	;\$47 - left slant box
.BYTE	\$01,\$03,\$07,\$0F,\$1F,\$3F,\$7F,\$FF	;\$48 - right slant solid
.BYTE	\$00,\$00,\$00,\$00,\$0F,\$0F,\$0F,\$0F	;\$49 - low right solid
.BYTE	\$80,\$C0,\$E0,\$F0,\$F8,\$FC,\$FE,\$FF	;\$4A - left slant solid
.BYTE	\$0F,\$0F,\$0F,\$0F,\$00,\$00,\$00,\$00	;\$4B - up right solid
.BYTE	\$F0,\$F0,\$F0,\$F0,\$00,\$00,\$00,\$00	;\$4C - up left solid
.BYTE	\$FF,\$FF,\$00,\$00,\$00,\$00,\$00,\$00	;\$4D - top box
.BYTE	\$00,\$00,\$00,\$00,\$00,\$00,\$FF,\$FF	;\$4E - bottom box
.BYTE	\$00,\$00,\$00,\$00,\$F0,\$F0,\$F0,\$F0	;\$4F - low left solid
.BYTE	\$00,\$1C,\$1C,\$77,\$77,\$08,\$1C,\$00	;\$50 - club card
.BYTE	\$00,\$00,\$00,\$1F,\$1F,\$18,\$18,\$18	;\$51 - up left window
.BYTE	\$00,\$00,\$00,\$FF,\$FF,\$00,\$00,\$00	;\$52 - mid box
.BYTE	\$18,\$18,\$18,\$FF,\$FF,\$18,\$18,\$18	;\$53 - mid window
.BYTE	\$00,\$00,\$3C,\$7E,\$7E,\$7E,\$3C,\$00	;\$54 - solid circle
.BYTE	\$00,\$00,\$00,\$00,\$FF,\$FF,\$FF,\$FF	;\$55 - bottom solid
.BYTE	\$C0,\$C0,\$C0,\$C0,\$C0,\$C0,\$C0,\$C0	;\$56 - left box
.BYTE	\$00,\$00,\$00,\$FF,\$FF,\$18,\$18,\$18	;\$57 - up mid window
.BYTE	\$18,\$18,\$18,\$FF,\$FF,\$00,\$00,\$00	;\$58 - low mid window
.BYTE	\$F0,\$F0,\$F0,\$F0,\$F0,\$F0,\$F0,\$F0	;\$59 - left solid
.BYTE	\$18,\$18,\$18,\$1F,\$1F,\$00,\$00,\$00	;\$5A - low left window
.BYTE	\$78,\$60,\$78,\$60,\$7E,\$18,\$1E,\$00	;\$5B - display escape
.BYTE	\$00,\$18,\$3C,\$7E,\$18,\$18,\$18,\$00	;\$5C - up arrow
.BYTE	\$00,\$18,\$18,\$18,\$7E,\$3C,\$18,\$00	;\$5D - down arrow
.BYTE	\$00,\$18,\$30,\$7E,\$30,\$18,\$00,\$00	;\$5E - left arrow
.BYTE	\$00,\$18,\$0C,\$7E,\$0C,\$18,\$00,\$00	;\$5F - right arrow
.BYTE	\$00,\$18,\$3C,\$7E,\$7E,\$3C,\$18,\$00	;\$60 - diamond card
.BYTE	\$00,\$00,\$3C,\$06,\$3E,\$66,\$3E,\$00	;\$61 - a
.BYTE	\$00,\$60,\$60,\$7C,\$66,\$66,\$7C,\$00	;\$62 - b
.BYTE	\$00,\$00,\$3C,\$60,\$60,\$60,\$3C,\$00	;\$63 - c
.BYTE	\$00,\$06,\$06,\$3E,\$66,\$66,\$3E,\$00	;\$64 - d
.BYTE	\$00,\$00,\$3C,\$66,\$7E,\$60,\$3C,\$00	;\$65 - e
.BYTE	\$00,\$0E,\$18,\$3E,\$18,\$18,\$18,\$00	;\$66 - f
.BYTE	\$00,\$00,\$3E,\$66,\$66,\$3E,\$06,\$7C	;\$67 - g
.BYTE	\$00,\$60,\$60,\$7C,\$66,\$66,\$66,\$00	;\$68 - h
.BYTE	\$00,\$18,\$00,\$38,\$18,\$18,\$3C,\$00	;\$69 - i
.BYTE	\$00,\$06,\$00,\$06,\$06,\$06,\$06,\$3C	;\$6A - j
.BYTE	\$00,\$60,\$60,\$6C,\$78,\$6C,\$66,\$00	;\$6B - k
.BYTE	\$00,\$38,\$18,\$18,\$18,\$18,\$3C,\$00	;\$6C - l
.BYTE	\$00,\$00,\$66,\$7F,\$7F,\$6B,\$63,\$00	;\$6D - m
.BYTE	\$00,\$00,\$7C,\$66,\$66,\$66,\$66,\$00	;\$6E - n
.BYTE	\$00,\$00,\$3C,\$66,\$66,\$66,\$3C,\$00	;\$6F - o
.BYTE	\$00,\$00,\$7C,\$66,\$66,\$7C,\$60,\$60	;\$70 - p
.BYTE	\$00,\$00,\$3E,\$66,\$66,\$3E,\$06,\$06	;\$71 - q
.BYTE	\$00,\$00,\$7C,\$66,\$60,\$60,\$60,\$00	;\$72 - r
.BYTE	\$00,\$00,\$3E,\$60,\$3C,\$06,\$7C,\$00	;\$73 - s
.BYTE	\$00,\$18,\$7E,\$18,\$18,\$18,\$0E,\$00	;\$74 - t
.BYTE	\$00,\$00,\$66,\$66,\$66,\$66,\$3E,\$00	;\$75 - u
.BYTE	\$00,\$00,\$66,\$66,\$66,\$3C,\$18,\$00	;\$76 - v
.BYTE	\$00,\$00,\$63,\$6B,\$7F,\$3E,\$36,\$00	;\$77 - w
.BYTE	\$00,\$00,\$66,\$3C,\$18,\$3C,\$66,\$00	;\$78 - x
.BYTE	\$00,\$00,\$66,\$66,\$66,\$3E,\$0C,\$78	;\$79 - y
.BYTE	\$00,\$00,\$7E,\$0C,\$18,\$30,\$7E,\$00	;\$7A - z
.BYTE	\$00,\$18,\$3C,\$7E,\$7E,\$18,\$3C,\$00	;\$7B - spade card
.BYTE	\$18,\$18,\$18,\$18,\$18,\$18,\$18,\$18	;\$7C -
.BYTE	\$00,\$7E,\$78,\$7C,\$6E,\$66,\$06,\$00	;\$7D - display clear
.BYTE	\$08,\$18,\$38,\$78,\$38,\$18,\$08,\$00	;\$7E - display backspace
.BYTE	\$10,\$18,\$1C,\$1E,\$1C,\$18,\$10,\$00	;\$7F - display tab

Device Handler Vector Tables

FIX EDITRV

** EDITRV - Editor Handler Vector Table

```
.WORD EOP-1 ;perform editor OPEN
.WORD ECL-1 ;perform editor CLOSE
.WORD EGB-1 ;perform editor GET-BYTE
.WORD EPB-1 ;perform editor PUT-BYTE
.WORD SST-1 ;perform editor STATUS (screen STATUS)
.WORD ESP-1 ;perform editor SPECIAL
JMP SIN ;initialize editor (initialize screen)
.BYTE 0 ;reserved
```

FIX SCRENV

** SCRENV - Screen Handler Vector Table

```
.WORD SOP-1 ;perform screen OPEN
.WORD ECL-1 ;perform screen CLOSE (editor CLOSE)
.WORD SGB-1 ;perform screen GET-BYTE
.WORD SPB-1 ;perform screen PUT-BYTE
.WORD SST-1 ;perform screen STATUS
.WORD SSP-1 ;perform screen SPECIAL
JMP SIN ;initialize screen
.BYTE 0 ;reserved
```

FIX KEYBDV

** KEYBDV - Keyboard Handler Vector Table

```
.WORD SST-1 ;perform keyboard OPEN (screen STATUS)
.WORD SST-1 ;perform keyboard CLOSE (screen STATUS)
.WORD KGB-1 ;perform keyboard GET-BYTE
.WORD ESP-1 ;perform keyboard SPECIAL (editor SPECIAL)
.WORD SST-1 ;perform keyboard STATUS (screen STATUS)
.WORD ESP-1 ;perform keyboard SPECIAL (editor SPECIAL)
JMP SIN ;initialize keyboard (initialize screen)
.BYTE 0 ;reserved
```

FIX PRINTV

** PRINTV - Printer Handler Vector Table

```
.WORD POP-1 ;perform printer OPEN
.WORD PCL-1 ;perform printer CLOSE
.WORD PSP-1 ;perform printer SPECIAL
.WORD PPB-1 ;perform printer PUT-BYTE
.WORD PST-1 ;perform printer STATUS
.WORD PSP-1 ;perform printer SPECIAL
JMP PIN ;initialize printer
.BYTE 0 ;reserved
```

FIX CASETV

**** CASETV - Cassette Handler Vector Table**

.WORD	COP-1	;perform cassette OPEN
.WORD	CCL-1	;perform cassette CLOSE
.WORD	CGB-1	;perform cassette GET-BYTE
.WORD	CPB-1	;perform cassette PUT-BYTE
.WORD	CST-1	;perform cassette STATUS
.WORD	CSP-1	;perform cassette SPECIAL
JMP	CIN	;initialize cassette
.BYTE	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 JMP	PHINIV PHI	;perform peripheral handler initialization ???

Generic Parallel Device Handler Vector Table

FIX

GPDVV

**	GPDVV - Generic Parallel Device Handler Vector Table
----	--

.WORD	GOP-1	;perform generic parallel device OPEN
.WORD	GCL-1	;perform generic parallel device CLOSE
.WORD	GGB-1	;perform generic parallel device GET-BYTE
.WORD	GPB-1	;perform generic parallel device PUT-BYTE
.WORD	GST-1	;perform generic parallel device STATUS
.WORD	GSP-1	;perform generic parallel device SPECIAL
JMP	GIN	;initialize generic parallel device

\$E4C0 Patch

FIX \$E4C0

** E4C0 - \$E4C0 Patch
*
* For compatibility with OS Revision B, return.

RTS ;return to caller

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      ICPTH,X
      TXA
      CLC
      ADC      #IOCBSZ         ;add IOCB size
      TAX
      CMP      #MAXIOC         ;index of next IOCB
      BCC      ICIO1           ;index of first invalid IOCB
      ;if not done

      RTS                     ;return to caller
```

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 to caller
```

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      CIO1 ;if IOCB not multiple of 16, error

      CPX      #MAXIOC ;index of first invalid IOCB
      BCC      CIO2    ;if index within range

;      Indicate Invalid IOCB Index error.

CIO1     LDY      #BADIOC ;invalid IOCB index error
      JMP      SSC      ;set status and complete operation, return

;      Move part of IOCB to zero page IOCB.

CIO2     LDY      #0      ;offset to first byte of page zero IOCB

CIO3     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      CIO3      ;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 ???.
XOP2     LDA      HNDLOD ;???
;      BNE      PPO     ;if user wants unconditional poll
;
;      Search handler table.
;
;      JSR      SHT      ;search handler table
;      BCS      PPO     ;if not found, poll
;
;      Initialize status.
;
;      LDA      #0
;      STA      DVSTAT ;clear status
;      STA      DVSTAT+1
;
;      Initialize IOCB.
;
;      JMP      IIO      ;initialize IOCB for OPEN, return
```

IIO - Initialize IOCB for OPEN

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

```
IIO      =      *      ;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      PHO      ;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

```
*      ??? word about implicit OPEN and CLOSE.
*
*      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      CCO    ;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 - Move IOCB in zero page back to user area

```
;      Returns with status stored in ICSTAZ
;      Move IOCB in zero page back to user area
*
*      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   ;clear ???

CCO1    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      CCO1     ;if not done
;
;      Restore A, X and Y.
        LDA      CIOCHR    ;data
        LDX      ICIDNO   ;IOCB index
        LDY      ICSTAZ   ;status
        RTS               ;return to caller
```

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      TCVO-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 to caller
```

DBL - Decrement Buffer Length

```
*      Z flag = 0 on return if length = 0 after decrement???
*
*      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                ;return to caller
```

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                ;return to caller
```

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                ;return to caller
```

SFL - Set Final Buffer Length

```
*      Set buffer length = buffer length - working byte count???
*
*      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 to caller
```

EHC - Execute Handler Command

```
*      Y = status on return, N flag=1 if error on return
*
*      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 to caller
```

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
*               2. Accept device number 0 (like revision B did).
*               Mike Barall 1984-06-21
```

```
        NOP                      ;keep the length of this module the
        NOP                      ;same as in previous revisions
        NOP
        NOP

SHT      =      *              ;entry
;        Set device number.

        SEC
        LDY     #1
        LDA     (ICBALZ),Y      ;device number
        SBC     #'0'

        CMP     #10
        BCC     SHT2            ;if number in range "0" to "9"

SHT1     LDA     #1              ;substitute device number "1"

SHT2     STA     ICDNOZ          ;device number (0 through 9)

;        Find device handler.

        LDY     #0              ;offset to device code
        LDA     (ICBALZ),Y      ;device code
;        JMP     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
;      RTS
;
;      Set handler ID.
;
FDH3     TYA
;      STA      ICHIDZ            ;offset to device code in table
;      CLC
;      RTS                      ;set handler ID
;                                ;indicate no error
;                                ;return to caller
```

```
**      TCV0 - Table of Command Vector Offsets
*
*      Entry n is the vector offset for command n+3.
```

```
TCV0     .BYTE    0              ;3 - open
;      .BYTE    4              ;4 - get ???
;      .BYTE    4              ;5 - get record
;      .BYTE    4              ;6 - get ???
;      .BYTE    4              ;7 - get byte(s)
;      .BYTE    6              ;8 - put ???
;      .BYTE    6              ;9 - put record
;      .BYTE    6              ;10 - put ???
;      .BYTE    6              ;11 - put byte(s)
;      .BYTE    2              ;12 - close
;      .BYTE    8              ;13 - status
;      .BYTE    10             ;14 - special
```

Peripheral Handler Loading Facility, Part 3

PHR - Peripheral Handler Routine

```
*      * 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
*
*      CALLS   CLT, PHW, PHP, GNL, LPH, PHC.
*
*      MODS    R. S. Scheiman  1982-04-??
*              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 to caller

;      Process answered poll.
PHR5   CLC                      ;Is there room to load?
      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 ??? fits

;      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      ;??? "Poll Reset" poll
      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 - Peripheral Handler Polling subroutine

```
*      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-??
*              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     .BYTE     $4F      ;device bus ID
      .BYTE     1      ;unit number
      .BYTE     '@'      ;type 3 or 4 poll command
      .BYTE     $40      ;I/O direction
      .WORD     DVSTAT      ;buffer
      .BYTE     30      ;timeout
      .BYTE     0
      .WORD     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-??
*              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

;      Reloacte 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-??
*              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      ;next ???
      BEQ      PHG2      ;if empty, load next block
;      Retrieve next byte.
PHG1     LDX      TEMP3      ;???
      LDA      CASBUF-$80,X ;byte
      CLC      ;indicate no error
      RTS      ;return to caller
;      Load next block and retrieve next byte.
PHG2     LDA      #-128     ;offset to first byte
      STA      TEMP3      ;reset ???
      JSR      GNL      ;get next load block
      BPL      PHG1      ;if no error, retrieve next byte
;      Process error.
      SEC      ;indicate error
      RTS      ;return to caller
```

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-??
*              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     .BYTE    $00          ;dummy device bus ID
;      .BYTE    1              ;dummy unit number
;      .BYTE    '&'            ;load command
;      .BYTE    $40            ;I/O direction
;      .WORD    CASBUF         ;buffer
;      .BYTE    30             ;timeout
;      .BYTE    0
;      .WORD    128            ;buffer length
;      .BYTE    0              ;auxiliary 1
;      .BYTE    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-??
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin 1983-11-01
```

```
SHC      =      *      ;entry
;        Initialize.
        STY      TEMP1      ;???save match parameters
        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      ;low chain pointer
        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 to caller
;        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 to caller
;        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 - Handler Initialization Subroutine

```

*      * 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-??
*          1. Bring closer to Coding Standard (object unchanged).
*          R. K. Nordin 1983-11-01

```

```

PHW = * ;entry
SEC ;indicate not zeroing handler size
PHP ;save ???
BCS PHQ ;skip chaining

```

PHI - Full Initialization Entry

```
*      ENTRY   JSR     PHI
*
*      MODS    R. S. Scheiman  1982-04-??
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin    1983-11-01
```

```
PHI    =        *                ;entry
        STA     DVSTAT+3          ;high ???
        STY     DVSTAT+2          ;low ???
;      JMP     PHC                ;???, return
```

PHC - Peripheral Handler Main Entry

```
*      ENTRY   JSR     PHC
*
*      MODS    R. S. Scheiman  1982-04-??
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin    1983-11-01
```

```
PHC    =        *                ;entry
;
;      Initialize.
        PHP                      ;save ???
;
;      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 to ???
        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
;
;      ???
;      JMP     PHQ                ;???, return
```

PHQ - Handler Initialization Continued

*	ENTRY	JSR	PHQ
*			
*	MODS	R. S. Scheiman	1982-04-??
*		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 to caller
;      Check for ???.
PHQ2     PLP          ;???
      BCS      PHQ3     ;if not to zero
;      Zero handler size.
      LDA      #0
      LDY      #16      ;offset to ???
      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
      RTS         ;indicate success
           ;return to caller
```

PHX - Initialize Handler

*	ENTRY	JSR	PHX
*			
*	MODS	R. S. Scheiman	1982-04-??
*		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-??
*              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
;
;      ???
;
;      TAY      ;(Save return parm)
;      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???
;
;      not loaded at MEMLO.
;
;      JSR      CLT      ;checksum linkage table
;      BNE      PHU2     ;if checksum nonzero, bad checksum???
```



```

;      OK remove table...

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 predecessor's link.
      STA      (ZCHAIN),Y
      DEY
      TXA
      STA      (ZCHAIN),Y
      CLC      ;indicate success
      RTS      ;return to caller

;      Clean stack and process error.

PHU2  PLA      ;Error return--restore stack
      PLA

;      Process error.

PHU3  SEC      ;indicate error
      RTS      ;return to caller

```

\$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 to caller
```

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
      BNE      SI04      ;if error received

      TYA
      BNE      SI05      ;status???
                        ;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 ;status???? type????
      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

```

CSO - Complete SIO Operation

```
*      ENTRY   JSR      CSO
*
*      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 to caller
```

WCA - Wait for Completion or ACK

```
*      ENTRY   JSR      WCA
*
*      EXIT     Y = 0, if failure
*               Y = $FF, if success
*
*      MODS     Original Author Unknown  ???-??-??
*               1. Bring closer to Coding Standard (object unchanged).
*                  R. K. Nordin 1983-11-01
*               2. Correct mishandling of NAK.
*                  R. K. Nordin 1984-03-23
```

```
WCA      =      *              ;entry
;
;      Initialize.
;
;      LDA      #0              ;clear error flag
;      STA      ERRFLG
;
;      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
;
;      Indicate failure.
;
WCA3     LDY      #0            ;failure indicator
;
;      Exit.
;
WCA4     LDA      STATUS         ;status
;      STA      TSTAT          ;temporary status
;      RTS                    ;return to caller
```

SEN - Sends a Buffer over the Serial Bus

```
*      ENTRY   JSR      SEN
*
*      MODS    Original Author Unknown  ???-??-??
*              1. Bring closer to Coding Standard (object unchanged).
*                R. K. Nordin 1983-11-01
*              2. Initialize CHKSUM before transmitting first byte.
*                R. K. Nordin 1984-03-23
```

```
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      CHKSUM          ;initialize checksum
      STA      SEROUT          ;serial output register
;      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
```

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          ;restore Y
      TAY          ;restore A
      PLA
      RTI          ;return from interrupt

;      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 from interrupt
```

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      #SUCCESS      ;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 to caller
```

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 to caller
```

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.
;      THIS MAY NOT BE THE PLACE TO DO IT ????????
      BMI      IRI1 ;if no frame error
;      Process frame error.
      LDY      #FRMERR ;frame error
      STY      STATUS ;indicate frame error
;      Check for overrun error.
IRI1     AND      #$20
      BNE      IRI2 ;if no overrun error
;      Process overrun error.
      LDY      #OVRERR ;overrun error
      STY      STATUS ;indicate overrun error
;      Check for buffer full.
IRI2     LDA      BUFRFL
      BEQ      IRI5 ;if buffer not yet full
;      Process buffer full.
      LDA      SERIN ;checksum from device
      CMP      CHKSUM ;computed checksum
      BEQ      IRI3 ;if checksums match
;      Process checksum error.
      LDY      #CHKERR ;checksum error
      STY      STATUS ;indicate checksum error
;      Indicate receive-frame done.
IRI3     LDA      #$FF ;receive-frame done indicator
      STA      RECDVN ;indicate receive-frame done
;      Exit.
IRI4     PLA
      TAY          ;restore Y
      PLA          ;restore A
      RTI          ;return from interrupt
;      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 to caller

```

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                    ;return to caller
```

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
;
;      ???

        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    ;and with ???
        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
;      JMP      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      SOUNDR  ;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 to caller
```

DSR - Disable SEND and RECEIVE

```
*      ENTRY   JSR     DSR
*
*      NOTES   Problem: NOP 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 to caller
```

GTO - Get Device Timeout

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

```
GTO    =      *          ;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 to caller
```

```
**      TSIH - Table of SIO Interrupt Handlers
*
*      NOTES    Problem: not used.
```

```
TSIH   .WORD    IRIR     ;serial input ready IRQ
        .WORD    ORIR     ;serial output ready IRQ
        .WORD    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
;      JMP     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 to caller

..... 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          ;difference
          LDX     PALNTS
          LDA     #0
          SEC
          SBC     CONS1X,X    ;???

CBR1      CLC
          ADC     CONS1X,X    ;accumulate product
          DEY
          BPL
          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
          LDY     #0          ;assume no addition correction
          SEC
          SBC     #7          ;adjust interpolation constant
          BPL     CBR3      ;if ???
          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    ;high POKEY frequency value
          STA     CBAUDH
          RTS          ;return to caller
```

.....
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 to caller

AVV1     CLC
          LDX     PALNTS
          ADC     CONS2X,X
          RTS                      ;return to caller
```

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      IT0      ;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           ;checksum for two bytes of $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 to caller

```

.....

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 to caller

```

```

**      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-out, where T-out is the number of counts
*      (127 used cd solution???) of VCOUNT for one character time (10 bit times).
*
*      Baud rate is ????.

```

```

;      .WORD      636      ;baud rate 1407, VCOUNT interval 56
;      .WORD      727      ;baud rate 1231, VCOUNT interval 64
;      .WORD      818      ;baud rate 1094, VCOUNT interval 72
;      .WORD      909      ;baud rate 985, VCOUNT interval 80

TPFV   .WORD      1000     ;baud rate 895, VCOUNT interval 88
       .WORD      1091     ;baud rate 820, VCOUNT interval 96
       .WORD      1182     ;baud rate 757, VCOUNT interval 104
       .WORD      1273     ;baud rate 703, VCOUNT interval 112
       .WORD      1364     ;baud rate 656, VCOUNT interval 120
       .WORD      1455     ;baud rate 615, VCOUNT interval 128
       .WORD      1546     ;baud rate 579, VCOUNT interval 136
       .WORD      1637     ;baud rate 547, VCOUNT interval 144
       .WORD      1728     ;baud rate 518, VCOUNT interval 152
       .WORD      1818     ;baud rate 492, VCOUNT interval 160
       .WORD      1909     ;baud rate 469, VCOUNT interval 168
       .WORD      2000     ;baud rate 447, VCOUNT interval 176

;
;      .WORD      2091     ;baud rate 428, VCOUNT interval 184
;      .WORD      2182     ;baud rate 410, VCOUNT interval 192
;      .WORD      2273     ;baud rate 394, VCOUNT interval 200
;      .WORD      2364     ;baud rate 379, VCOUNT interval 208
;      .WORD      2455     ;baud rate 365, VCOUNT interval 216
;      .WORD      2546     ;baud rate 352, VCOUNT interval 224
;      .WORD      2637     ;baud rate 339, VCOUNT interval 232
;      .WORD      2728     ;baud rate 328, VCOUNT interval 240
;      .WORD      2819     ;baud rate 318, VCOUNT interval 248

```

** Serial Input/Output NTSC/PAL Constant Tables

```

WIRGLX .BYTE      low WIRGLN      ;NTSC low long write IRG
        .BYTE      low WIRGLP     ;PAL low long write IRG

RIRGLX .BYTE      low RIRGLN      ;NTSC low long read IRG
        .BYTE      low RIRGLP     ;PAL low long read IRG

WSIRGX .BYTE      low WSIRGN      ;NTSC low short write IRG
        .BYTE      low WSIRGP     ;PAL low short write IRG

RSIRGX .BYTE      low RSIRGN      ;NTSC low short read IRG
        .BYTE      low RSIRGP     ;PAL low short read IRG

CONS1X .BYTE      131             ;NTSC ???
        .BYTE      156           ;PAL ???

CONS2X .BYTE      7               ;NTSC ???
        .BYTE      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  .BYTE  24      ;0
      .BYTE  16      ;1
      .BYTE  10      ;2
      .BYTE  10      ;3
      .BYTE  16      ;4
      .BYTE  28      ;5
      .BYTE  52      ;6
      .BYTE  100     ;7
      .BYTE  196     ;8
      .BYTE  196     ;9
      .BYTE  196     ;10
      .BYTE  196     ;11
      .BYTE  28      ;12
      .BYTE  16      ;13
      .BYTE  100     ;14
      .BYTE  196     ;15
```

```
**      TDLE - Table of Display List Entry Counts
*
*      Each entry is 2 bytes.  The first byte is ??????
```

```
TDLE  .BYTE  23,23   ;0
      .BYTE  11,23   ;1
      .BYTE  47,47   ;2
      .BYTE  95,95   ;3
      .BYTE  97,97   ;4
      .BYTE  97,97   ;5
      .BYTE  23,11   ;6
      .BYTE  191,97  ;7
      .BYTE  19,19   ;8
      .BYTE  9,19    ;9
      .BYTE  39,39   ;10
      .BYTE  79,79   ;11
      .BYTE  65,65   ;12
      .BYTE  65,65   ;13
      .BYTE  19,9    ;14
      .BYTE  159,65  ;15
```

```
**      TAGM - Table of ANTIC Graphics Modes
*
*      Entry n is the ANTIC graphics mode corresponding to internal graphics mode n.
```

```
TAGM  .BYTE  $02      ;internal 0 - 40x2x8 characters
      .BYTE  $06      ;internal 1 - 20x5x8 characters
      .BYTE  $07      ;internal 2 - 20x5x16 characters
      .BYTE  $08      ;internal 3 - 40x4x8 graphics
      .BYTE  $09      ;internal 4 - 80x2x4 graphics
      .BYTE  $0A      ;internal 5 - 80x4x4 graphics
      .BYTE  $0B      ;internal 6 - 160x2x2 graphics
      .BYTE  $0D      ;internal 7 - 160x4x2 graphics
      .BYTE  $0F      ;internal 8 - 320x2x1 graphics
      .BYTE  $0F      ;internal 9 - 320x2x1 GTIA "lum" mode
      .BYTE  $0F      ;internal 10 - 320x2x1 GTIA "color/lum" mode
      .BYTE  $0F      ;internal 11 - 320x2x1 GTIA "color" mode
      .BYTE  $04      ;internal 12 - 40x5x8 characters
      .BYTE  $05      ;internal 13 - 40x5x16 characters
      .BYTE  $0C      ;internal 14 - 160x2x1 graphics
      .BYTE  $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  .BYTE  0      ;0
      .BYTE  0      ;1
      .BYTE  0      ;2
      .BYTE  0      ;3
      .BYTE  0      ;4
      .BYTE  0      ;5
      .BYTE  0      ;6
      .BYTE  1      ;7
      .BYTE  1      ;8
      .BYTE  1      ;9
      .BYTE  1     ;10
      .BYTE  1     ;11
      .BYTE  0     ;12
      .BYTE  0     ;13
      .BYTE  1     ;14
      .BYTE  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  .BYTE  3      ;0
      .BYTE  2      ;1
      .BYTE  2      ;2
      .BYTE  1      ;3
      .BYTE  1      ;4
      .BYTE  2      ;5
      .BYTE  2      ;6
      .BYTE  3      ;7
      .BYTE  3      ;8
      .BYTE  3      ;9
      .BYTE  3     ;10
      .BYTE  3     ;11
      .BYTE  3     ;12
      .BYTE  3     ;13
      .BYTE  2     ;14
      .BYTE  3     ;15

```

```

**      TMCC - Table of Mode Column Counts
*
*      Entry n is the low column count for mode n.

```

```

TMCC  .BYTE  low 40      ;0
      .BYTE  low 20      ;1
      .BYTE  low 20      ;2
      .BYTE  low 40      ;3
      .BYTE  low 80      ;4
      .BYTE  low 80      ;5
      .BYTE  low 160     ;6
      .BYTE  low 160     ;7
      .BYTE  low 320     ;8
      .BYTE  low 80      ;9
      .BYTE  low 80     ;10
      .BYTE  low 80     ;11
      .BYTE  low 40     ;12
      .BYTE  low 40     ;13
      .BYTE  low 160     ;14
      .BYTE  low 160     ;15

```

```

**      TMRC - Table of Mode Row Counts
*
*      Entry n is the row count for mode n.

```

```

TMRC      .BYTE    24      ;0
          .BYTE    24      ;1
          .BYTE    12      ;2
          .BYTE    24      ;3
          .BYTE    48      ;4
          .BYTE    48      ;5
          .BYTE    96      ;6
          .BYTE    96      ;7
          .BYTE   192      ;8
          .BYTE   192      ;9
          .BYTE   192     ;10
          .BYTE   192     ;11
          .BYTE    24     ;12
          .BYTE    12     ;13
          .BYTE   192     ;14
          .BYTE   192     ;15

```

```

**      TRSC - Table of Right Shift Counts
*
*      Entry n is ??? for mode n.
*      HOW MANY RIGHT SHIFTS FOR HCRSR FOR PARTIAL BYTE MODES

```

```

TRSC      .BYTE     0      ;0
          .BYTE     0      ;1
          .BYTE     0      ;2
          .BYTE     2      ;3
          .BYTE     3      ;4
          .BYTE     2      ;5
          .BYTE     3      ;6
          .BYTE     2      ;7
          .BYTE     3      ;8
          .BYTE     1      ;9
          .BYTE     1     ;10
          .BYTE     1     ;11
          .BYTE     0     ;12
          .BYTE     0     ;13
          .BYTE     3     ;14
          .BYTE     2     ;15

```

```

**      TDSM - Table of Display Masks

```

```

TDSM      .BYTE    $FF     ;1
          .BYTE    $F0     ;2
          .BYTE    $0F     ;3
          .BYTE    $C0     ;4
          .BYTE    $30     ;5
          .BYTE    $0C     ;6
          .BYTE    $03     ;7

```

```

**      TBTM - Table of Bit Masks

```

```

TBTM      .BYTE    $80     ;8 (0)
          .BYTE    $40     ;9 (1)
          .BYTE    $20    ;10 (2)
          .BYTE    $10    ;11 (3)
          .BYTE    $08    ;12 (4)
          .BYTE    $04    ;13 (5)
          .BYTE    $02    ;14 (6)
          .BYTE    $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-??
*               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 to caller
```

```

;      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 to caller

;      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 to caller

```

PHO - 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        PHO
*
*      MODS     R. S. Scheiman   1982-04-??
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin     1983-11-01
```

```
PHO      =      *      ;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 to caller

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
          STA     ICSPR+1,X
          LDY     #0
          LDA     (ICBALZ),Y   ;(Device name from user)
          STA     ICSPR,X
          LDY     #SUCCES      ;indicate success
          RTS              ;return to caller
```

PTL - 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
*
*      MODS     R. S. Scheiman   1982-04-??
*              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      HNDLOD   ;???
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 to caller

PTL2     LDY      #BADIOC ;indicate bad IOCB number error
BMI      PTL1     ;return error

;      Simulate beginning of CIO, since CIO bypassed.

PTL3     STX      ICIDNO ;IOCB index
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
      TAX
      PLA
      TAY
      LDA      ICPHYZ
      PHA
      LDA      ICPTLZ      ;(Put byte entry address minus one)
      PHA
      TYA
      LDY      #FNCNOT
      RTS              ;invoke handler (address on stack)

```

\$EF6B Patch

FIX \$EF6B

**	EF6B - \$EF6B Patch
*	
*	For compatibility with OS Revision B, initiate cassette READ.

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     SHFL0K        ;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 to caller
```

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 ;???
;      DLI???

      LDA      #$40 ;disable DLI
      STA      NMIEEN ;???
      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 ???.

      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      #$01 ;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
```

```

;      ???

LDY      RAMTOP          ;(high) RAM size
DEY      ;decrement (high) RAM size
STY      TXTMSC+1        ;high ???
LDA      #low [$0000-160] ;low RAM size - 160
STA      TXTMSC          ;low ???
LDX      DINDEX          ;mode
LDA      TAGM,X          ;convert to ANTIC code???
STA      HOLD1           ;??? ANTIC code
LDA      RAMTOP          ;(high) RAM size
STA      ADRESS+1        ;high ???

;      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

;      ????.

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
ROR      A               ;mode
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

;      ???

COC7     STY      GPRIOR   ;new priority
        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 ADDRESS
DEC ADDRESS+1 ;drop down 1 page

COC9 JSR DDD ;perform double byte double decrement
LDA ADDRESS ;end of display list
STA SAVADR ;save address
LDA ADDRESS+1
STA SAVADR+1

; ???

LDA #$41 ;??? (ANTIC) wait for VBLANK and JMP
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 ;fine scrolling???
JSR SDF ;store data indirect
CLC
LDA #MXDMOD
ADC OPNTMP
TAY
LDX TDLE,Y ;???
BNE COC13 ;???

; ???

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       ;???

;      ???

COC13  LDA      HOLD1       ;???
      JSR      SDI          ;store data indirect
      DEX
      BNE      COC13       ;if ???

;      Determine mode.  MESSY 320x1 problem???

      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
      JSR      SDI          ;low saved memory scan counter
      LDA      HOLD1       ;store data indirect
      ORA      #$40
      JSR      SDI          ;store data indirect
      LDA      #$70
      JSR      SDI          ;8 blank lines ???
      LDA      #$70
      JSR      SDI          ;store data indirect
      LDA      #$70
      JSR      SDI          ;8 blank lines ???
      LDA      SDI          ;store data indirect
      LDA      ADRESS
      STA      SDLSTL       ;display list address
      LDA      ADRESS+1
      STA      SDLSTL+1    ;save display list address
      LDA      ADRESS+1
      STA      SDLSTL+1
      LDA      #$70
      JSR      SDI          ;8 blank lines
      LDA      ADRESS
      STA      MEMTOP       ;display list address
      LDA      MEMTOP
      STA      MEMTOP+1    ;update top of memory
      LDY      #1
      LDA      SDLSTL       ;offset to ???
      STA      (SAVADR),Y   ;saved display list address
      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 to caller

;      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 to caller

```

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           ;???, return
```

SPQ - ???

```
*      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
;      ???

SPQ2   AND     DMASK
      STA     TMPCHR         ;save shifted data
      LDA     DMASK          ;display mask
      EOR     #$FF          ;complement mdisplay mask
      AND     (ADDRESS),Y    ;mask off old data
      ORA     TMPCHR         ;or in new data
      STA     (ADDRESS),Y    ;update data
      RTS                   ;return to caller
```

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     ;TOGGLE MSB
          JSR      SPQ      ;display
          JMP      SST      ;???, 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

**	F223 - \$F223 Patch
*	
*	For compatibility with OS Revision B, perform power-up display.

PPD	=	*	;entry
	JMP	SES	;select and execute self-test

Keyboard, Editor and Screen Handler, Part 3

```
;      Continue.

SST1   LDA    #SUCCES ;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 to caller
```

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     NMIEEN  ;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      ;???
      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      ;??? (undo swap of PCH)
      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.
;
;      ???

      JSR      SGB      ;perform screen GET-BYTE
      STA      ATACHR   ;ATASCII character
      JMP      SWA      ;swap ???, return
;
;      ???

EGB6     JSR      RWS      ;return with scrolling
      LDA      #EOL
      STA      ATACHR   ;ATASCII character
      JSR      SEC      ;set exit conditions
      STY      DSTAT    ;status
      JMP      SWA      ;???, return
```

IRA - Invoke Routine Pointed to by ADDRESS

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

IRA = * ;entry
 JMP (ADDRESS) ;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 ;???
 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 ROD ;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 ;???, return

; Process control character.

PCH2 LDA DSPFLG ;display flag
      ORA ESCFLG ;escape flag
      BNE PCH1 ;if display or escape, display character

; ???

      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 ;???, return
```

IGN - Ignore Character and Perform Keyboard GET-BYTE

```
* ENTRY JSR IGN
*
* EXIT CH = $FF
*
* CALLS KGB
*
* 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
;
;      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 ??? key.

KGB9   CMP      #$85
      BNE      KGB12      ;if not ??? key.

;      Process ??? 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 to caller
```

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 ???, move cursor to right 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 ???.

      LDY      ADRESS          ;low ???
      LDA      #0
      STA      ADRESS          ;clear low ???

CSC1     STA      (ADRESS),Y      ;clear ???
      INY
      BNE      CSC1            ;if not done with page

      INC      ADRESS+1          ;increment high ???
      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                      ;return to caller
```

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      #' '      ;ATASCII character
          STA      ATACHR   ;plot point
          JSR      PLO

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

;        ???

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   ;???

;      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      DQO              ;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              ;NORMAL ILN PUTS "1" INTO BIT MAP
;          JMP      ILN1          ;???, return
```

ILN1 - Insert Line Continued

```
*      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    ;???, return
```

DLN1 - Delete Line Continued

```
*      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

;      ???

          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 to caller
```


..... 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      (ADDRESS),Y

;      Decrement ???.

;      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
          SBC     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 to caller
```

SSE - Set Scrolling Display List Entry

```
*      Store extra line in display list for fine scrolling modes.
*
*      ENTRY   JSR       SSE
*
*      MODS    Original Author Unknown  ???-??-??
*              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 to caller
```

CCA - Convert Cursor Row/Column to Address

*	ENTRY	JSR	CCA
*			
*	MODS	Original Author Unknown	????-??-??
*		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	;if ???
	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	;if ???
	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	;if ???
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	;add ???
	TAY		
	LDA	TDSM-1,Y	;display mask
	STA	DMASK	;display mask
	STA	SHFAMT	;???
	LDY	#0	
CCA6	RTS		;return to caller

SDA - Set Zero Data and Advance Cursor Column

```
*      Increment cursor and detect both end of line and end of screen
*
*      ENTRY   JSR      SZA
*
*      MODS    Original Author Unknown  ????-??-??
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin 1983-11-01
```

```
SDA      =      *      ;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???
          JMP      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
          JMP      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       ;if not mode 0, exit
          BNE     CCA6

          CMP     RMARGN   ;right margin
          BEQ     CCA6     ;if at right margin, exit

          BCC     CCA6     ;if before right margin, exit

ACC2     CPX     #8       ;if not mode 8
          BNE     ACC3

          LDA     COLCRS+1 ;high cursor column
          BEQ     CCA6     ;if ONLY AT 64 SO DON'T DO IT???

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     ;if ???don't extend

          LDA     INSDAT   ;???
          BEQ     ACC5     ;if zero, do not extend???

          CLC
          JSR     ILN1     ;indicate ???
                          ;???

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      DINDEX
      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             ;if ???

      STY      HOLD3
      TXA
      BNE      RET5             ;mode
                                   ;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             ;if ???

      INC      BUFSTR

RET4    DEC      HOLD3
      LDA      LOGMAP
      SEC
      BPL      RET3             ;indicate ??? for partial line
                                   ;if partial logical line

      LDA      HOLD3           ;???
      STA      ROWCRS          ;cursor row

RET5    JMP      SLC              ;set logical column, return
```

SEP - Subtract ENDPT from ROWAC or COLAC

```
*      Subtract ENDPT from ROWAC or COLAC. (X=0 OR 2)
*
*      ENTRY   JSR      SEP
*              X = ???
*
*      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 to caller
```

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      ;right margin
          CMP      RMARGN    ;if 39 >= right margin
          BCS      CCR1

          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 to caller

;      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    ;if not swapped
          BPL      CCR7

          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 to caller
```

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 to caller
```

BLR - Rotate Logical Line Bit Map Left

```
*      BLR rotates the logical line bit map left, scrolling the logical lines up.
*
*      ENTRY   JSR      BLR
*              C = ???
*
*      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 to caller
```

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
;          JMP     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 to caller
```

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                ;return to caller
```

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
      ; JNE      BLG1      ;???, return
```

BLG1 - Get Bit from Logical Line Bit Map Continued 1

```
*      ENTRY   JSR      BLG1
*
*      MODS    Original Author Unknown  ????-??-??
*              1. Bring closer to Coding Standard (object unchanged).
*              R. K. Nordin 1983-11-01
```

```
BLG1   =      *          ;entry
      CLC
      ; JNE      BLG2      ;???, return
```

BLG2 - Get Bit from Logical Line Bit Map Continued 2

```
*      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
      ; JNE      BMG          ;get bit from bit map, return
```

BMG - Get Bit from Bit Map

```
*      ENTRY   JSR    BMG
*
*      EXIT    C = ???
*
*      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    ;if ???

          SEC

BMG1     RTS           ;return to caller
```

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 to caller
```

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	;if ???
	CPY	ROWCRS	;cursor row
	BCS	ELL1	;if ???
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	(ADDRESS),Y	
	DEY		
	BPL	CLN1	;if not done
	RTS		;return to caller

SCR - ???

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

```
SCR      =      *      ;entry
          JSR      BLR      ;rotate logical line bit map left
          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 SDAN
          BNE      SCR2      ;if not done waiting for ???

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 for ???

;      ???

SCR5     JSR      SMS          ;set memory scan counter address
          JMP      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      ADRESS          ;address
      LDX      ADRESS+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      ADRESS+1
      INX
      CPX      RAMTOP
      BEQ      SSD6            ;if at RAMTOP

      CLC
      ADC      #$10

;      Adjust address.

SSD3     TAY              ;number of bytes
      STA      COUNTR        ;???
      SEC
      LDA      ADRESS
      SBC      COUNTR        ;subtract ???
      STA      ADRESS        ;update low address
      BCS      SSD4          ;if no borrow

      DEC      ADRESS+1      ;adjust high address

;      Move data down.

SSD4     LDA      ADRESS
      CLC
      ADC      #40
      STA      COUNTR        ;address + 40
      LDA      ADRESS+1
      ADC      #0
      STA      COUNTR+1

SSD5     LDA      (COUNTR),Y    ;byte to move
      STA      (ADRESS),Y      ;move byte
      INY
      BNE      SSD5            ;if not done (256-16 times)

      LDY      #256-240
      LDA      ADRESS
      CMP      #-40
      BEQ      SSD6            ;if all done

      CLC
      ADC      #240
      STA      ADRESS        ;update low address
      BCC      SSD4          ;if no carry

      INC      ADRESS+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
*
*      EXIT    HOLD1 = ???
*
*      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              ;return to caller

```

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      ;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 to caller
```

DQQ - Deletes a Line if it is empty and an Extension

*	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        ;???, return
```

DWQ - ???

*	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       ;if ???, return

          DEY
          BPL      DWQ1       ;if not done

;      Delete line.

          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 to caller
```

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 to caller
```

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 to caller
```

SWA - If mixed mode, swap ??? cursor with regular cursors

```
*      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 ??? parameters.
;
;      LDX      #11          ;offset to last byte
SWA1     LDA      ROWCRS,X    ;??? parameter
;      PHA                      ;save ??? parameter
;      LDA      TXTRW,X      ;??? parameter
;      STA      ROWCRS,X      ;update ???
;      PLA                      ;saved ???
;      STA      TXTRW,X      ;update ???
;      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 to caller
```

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 ???
        LDX     DINDEXT   ;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 to caller
```

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     ADRESS     ;set low address
        LDA     SAVMSC+1   ;saved high memory scan counter
        STA     ADRESS+1   ;set high address
        RTS           ;return to caller
```

SSP - Perform Screen SPECIAL

```
*      SSP draws a line from OLDROW/OLDCOL to NEWROW/NEWCOL.
*
*      ENTRY   JSR      SSP
*
*      MODS     Original Author Unknown  ???-??-??
*              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      #INVALID ;invalid command error
      RTS

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      DELTAR   ;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 ????.

      LDA    DELTAC       ;low column difference
      INX
      TAY                ;offset to working row
                        ;low column difference
      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
      STA    COUNTR       ;low maximum difference
                        ;low iteration counter
      STA    ENDPT        ;low end point
      PHA
      LDA    ENDPT+1      ;save low end point
      LSR    A            ;high end point
                        ;C = LSB of high end point
      PLA
      ROR    A            ;saved low end point
                        ;???
      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

;      ????.

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 ADC STA LDX JSR	ROWCRS ROWINC ROWCRS #0 SEP	;cursor row ;add row increment ;update cursor row ;indicate subtract from working row ;subtract end pointer
SSP11	CLC LDA ADC STA LDA ADC STA CMP BCC BNE LDA CMP BCC	COLAC DELTAC COLAC COLAC+1 DELTAC+1 COLAC+1 ENDPT+1 SSP15 SSP12 COLAC ENDPT SSP15	;low working column ;add column difference ;update working column ;high end point ;if high working column < high end point ;if high working column > high end point ;low working column ;low end point ;if low working column < low end point
SSP12	BIT BPL DEC LDA CMP BNE LDA BEQ DEC BPL	COLINC SSP13 COLCRS COLCRS #\$FF SSP14 COLCRS+1 SSP14 COLCRS+1 SSP14	;column increment ;if column increment positive ;decrement low cursor column ;low cursor column ;if ??? ;high cursor column ;if zero, do not decrement ;decrement high cursor column ;???
SSP13	INC BNE INC	COLCRS SSP14 COLCRS+1	;increment low cursor column ;if no carry ;adjust high cursor column
SSP14	LDX JSR	#2 SEP	;indicate subtract from working column ;subtract end pointer
; Plot point.			
SSP15	JSR JSR	CCR PLO	;check cursor range ;plot point
; Check for right fill.			
	LDA BEQ	FILFLG SSP18	;right fill flag ;if no right fill
; Process right fill.			
	JSR LDA STA	SRC ATACHR HOLD4	;save row and column ;plot point ;save plot point
SSP16	LDA PHA JSR PLA STA JSR JSR BNE LDA STA JSR JMP	ROWCRS ACC ROWCRS CCR GDC SSP17 FILDAT ATACHR PLO SSP16	;cursor row ;save cursor row ;advance cursor column ;saved cursor row ;restore cursor row ;check cursor range ;get data under cursor ;if non-zero data encountered ;fill data ;plot point ;plot point ;continue
SSP17	LDA STA JSR	HOLD4 ATACHR RRC	;saved plot point ;restore plot point ;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 - Mask used in CCA, Convert Cursor Row/Column to Address

```

TMSK   .BYTE   $00      ;0 - mask for no bits
       .BYTE   $01      ;1 - mask for lower 1 bit
       .BYTE   $03      ;2 - mask for lower 2 bits
       .BYTE   $07      ;3 - mask for lower 3 bits

```

** TDSC - Table of Default Screen Colors

```

TDSC   .BYTE   $28      ;default playfield 0 color
       .BYTE   $CA      ;default playfield 1 color
       .BYTE   $94      ;default playfield 2 color
       .BYTE   $46      ;default playfield 3 color
       .BYTE   $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   .BYTE   $1B
       .WORD   ESC      ;escape

       .BYTE   $1C
       .WORD   CUP      ;move cursor up

       .BYTE   $1D
       .WORD   CDN      ;move cursor down

       .BYTE   $1E
       .WORD   CLF      ;move cursor left

       .BYTE   $1F
       .WORD   CRT      ;move cursor right

       .BYTE   $7D
       .WORD   CSC      ;clear screen

       .BYTE   $7E
       .WORD   BSP      ;backspace

       .BYTE   $7F
       .WORD   TAB      ;tab

       .BYTE   $9B
       .WORD   RWS      ;return with scrolling

       .BYTE   $9C
       .WORD   DLN      ;delete line

       .BYTE   $9D
       .WORD   ILN      ;insert line

       .BYTE   $9E
       .WORD   CTB      ;clear tab

```

```

        .BYTE    $9F
        .WORD    STB          ;set tab

        .BYTE    $FD
        .WORD    BEL          ;sound bell

        .BYTE    $FE
        .WORD    DCH          ;delete character

        .BYTE    $FF
        .WORD    ICH          ;insert character

TCCR1L =      *-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    .BYTE    $1C
        .WORD    CHM          ;move cursor home

        .BYTE    $1D
        .WORD    CBT          ;move cursor to bottom

        .BYTE    $1E
        .WORD    CLM          ;move cursor to left margin

        .BYTE    $1F
        .WORD    CRM          ;move cursor to right margin

```

```

**      TAIC - Table of ATASCII to Internal Conversion Constants

```

```

TAIC    .BYTE    $40          ;0 - ???
        .BYTE    $00          ;1 - ???
        .BYTE    $20          ;2 - ???
        .BYTE    $60          ;3 - ???

```

```

**      TIAC - Table of Internal to ATASCII Conversion Constants

```

```

TIAC    .BYTE    $20          ;0 - ???
        .BYTE    $40          ;1 - ???
        .BYTE    $00          ;2 - ???
        .BYTE    $60          ;3 - ???

```

```

**      TCKD - Table of Character Key Definitions
*
*      Entry n is the ATASCII equivalent of key code n.???

```

```

TCKD      =      *
;      Lower Case Characters

      .BYTE      $6C      ;$00 - l
      .BYTE      $6A      ;$01 - j
      .BYTE      $3B      ;$02 - semicolon
      .BYTE      $8A      ;$03 - F1
      .BYTE      $8B      ;$04 - F2
      .BYTE      $6B      ;$05 - k
      .BYTE      $2B      ;$06 - +
      .BYTE      $2A      ;$07 - *
      .BYTE      $6F      ;$08 - o
      .BYTE      $80      ;$09 - (invalid)
      .BYTE      $70      ;$0A - p
      .BYTE      $75      ;$0B - u
      .BYTE      $9B      ;$0C - return
      .BYTE      $69      ;$0D - i
      .BYTE      $2D      ;$0E - -
      .BYTE      $3D      ;$0F - =

      .BYTE      $76      ;$10 - v
      .BYTE      $80      ;$11 - (invalid)
      .BYTE      $63      ;$12 - c
      .BYTE      $8C      ;$13 - F3
      .BYTE      $8D      ;$14 - F4
      .BYTE      $62      ;$15 - b
      .BYTE      $78      ;$16 - x
      .BYTE      $7A      ;$17 - z
      .BYTE      $34      ;$18 - 4
      .BYTE      $80      ;$19 - (invalid)
      .BYTE      $33      ;$1A - 3
      .BYTE      $36      ;$1B - 6
      .BYTE      $1B      ;$1C - escape
      .BYTE      $35      ;$1D - 5
      .BYTE      $32      ;$1E - 2
      .BYTE      $31      ;$1F - 1

      .BYTE      $2C      ;$20 - comma
      .BYTE      $20      ;$21 - space
      .BYTE      $2E      ;$22 - period
      .BYTE      $6E      ;$23 - n
      .BYTE      $80      ;$24 - (invalid)
      .BYTE      $6D      ;$25 - m
      .BYTE      $2F      ;$26 - /
      .BYTE      $81      ;$27 - inverse
      .BYTE      $72      ;$28 - r
      .BYTE      $80      ;$29 - (invalid)
      .BYTE      $65      ;$2A - e
      .BYTE      $79      ;$2B - y
      .BYTE      $7F      ;$2C - tab
      .BYTE      $74      ;$2D - t
      .BYTE      $77      ;$2E - w
      .BYTE      $71      ;$2F - q

      .BYTE      $39      ;$30 - 9
      .BYTE      $80      ;$31 - (invalid)
      .BYTE      $30      ;$32 - 0
      .BYTE      $37      ;$33 - 7
      .BYTE      $7E      ;$34 - backspace
      .BYTE      $38      ;$35 - 8
      .BYTE      $3C      ;$36 - <
      .BYTE      $3E      ;$37 - >
      .BYTE      $66      ;$38 - f
      .BYTE      $68      ;$39 - h
      .BYTE      $64      ;$3A - d
      .BYTE      $80      ;$3B - (invalid)
      .BYTE      $82      ;$3C - CAPS
      .BYTE      $67      ;$3D - g
      .BYTE      $73      ;$3E - s
      .BYTE      $61      ;$3F - a

```

```

;      Upper Case Characters

.BYTE  $4C      ;$40 - L
.BYTE  $4A      ;$41 - J
.BYTE  $3A      ;$42 - colon
.BYTE  $8A      ;$43 - SHIFT-F1
.BYTE  $8B      ;$44 - SHIFT-F2
.BYTE  $4B      ;$45 - K
.BYTE  $5C      ;$46 - \
.BYTE  $5E      ;$47 - ^
.BYTE  $4F      ;$48 - O
.BYTE  $80      ;$49 - (invalid)
.BYTE  $50      ;$4A - P
.BYTE  $55      ;$4B - U
.BYTE  $9B      ;$4C - SHIFT-return
.BYTE  $49      ;$4D - I
.BYTE  $5F      ;$4E - _
.BYTE  $7C      ;$4F - |

.BYTE  $56      ;$50 - V
.BYTE  $80      ;$51 - (invalid)
.BYTE  $43      ;$52 - C
.BYTE  $8C      ;$53 - SHIFT-F3
.BYTE  $8D      ;$54 - SHIFT-F4
.BYTE  $42      ;$55 - B
.BYTE  $58      ;$56 - X
.BYTE  $5A      ;$57 - Z
.BYTE  $24      ;$58 - $
.BYTE  $80      ;$59 - (invalid)
.BYTE  $23      ;$5A - #
.BYTE  $26      ;$5B - &
.BYTE  $1B      ;$5C - SHIFT-escape
.BYTE  $25      ;$5D - %
.BYTE  $22      ;$5E - "
.BYTE  $21      ;$5F - !

.BYTE  $5B      ;$60 - [
.BYTE  $20      ;$61 - SHIFT-space
.BYTE  $5D      ;$62 - ]
.BYTE  $4E      ;$63 - N
.BYTE  $80      ;$64 - (invalid)
.BYTE  $4D      ;$65 - M
.BYTE  $3F      ;$66 - ?
.BYTE  $81      ;$67 - SHIFT-inverse
.BYTE  $52      ;$68 - R
.BYTE  $80      ;$69 - (invalid)
.BYTE  $45      ;$6A - E
.BYTE  $59      ;$6B - Y
.BYTE  $9F      ;$6C - SHIFT-tab
.BYTE  $54      ;$6D - T
.BYTE  $57      ;$6E - W
.BYTE  $51      ;$6F - Q

.BYTE  $28      ;$70 - (
.BYTE  $80      ;$71 - (invalid)
.BYTE  $29      ;$72 - )
.BYTE  $27      ;$73 - '
.BYTE  $9C      ;$74 - SHIFT-delete
.BYTE  $40      ;$75 - @
.BYTE  $7D      ;$76 - SHIFT-clear
.BYTE  $9D      ;$77 - SHIFT-insert
.BYTE  $46      ;$78 - F
.BYTE  $48      ;$79 - H
.BYTE  $44      ;$7A - D
.BYTE  $80      ;$7B - (invalid)
.BYTE  $83      ;$7C - SHIFT-CAPS
.BYTE  $47      ;$7D - G
.BYTE  $53      ;$7E - S
.BYTE  $41      ;$7F - A

```

;

Control Characters

.BYTE	\$0C	;\$80 - CTRL-L
.BYTE	\$0A	;\$81 - CTRL-J
.BYTE	\$7B	;\$82 - CTRL-semicolon
.BYTE	\$80	;\$83 - (invalid)
.BYTE	\$80	;\$84 - (invalid)
.BYTE	\$0B	;\$85 - CTRL-K
.BYTE	\$1E	;\$86 - CTRL-left arrow
.BYTE	\$1F	;\$87 - CTRL-right arrow
.BYTE	\$0F	;\$88 - CTRL-O
.BYTE	\$80	;\$89 - (invalid)
.BYTE	\$10	;\$8A - CTRL-P
.BYTE	\$15	;\$8B - CTRL-U
.BYTE	\$9B	;\$8C - CTRL-return
.BYTE	\$09	;\$8D - CTRL-I
.BYTE	\$1C	;\$8E - CTRL-up arrow
.BYTE	\$1D	;\$8F - CTRL-down arrow
.BYTE	\$16	;\$90 - CTRL-V
.BYTE	\$80	;\$91 - (invalid)
.BYTE	\$03	;\$92 - CTRL-C
.BYTE	\$89	;\$93 - CTRL-F3
.BYTE	\$80	;\$94 - (invalid)
.BYTE	\$02	;\$95 - CTRL-B
.BYTE	\$18	;\$96 - CTRL-X
.BYTE	\$1A	;\$97 - CTRL-Z
.BYTE	\$80	;\$98 - (invalid)
.BYTE	\$80	;\$99 - (invalid)
.BYTE	\$85	;\$9A - ???
.BYTE	\$80	;\$9B - (invalid)
.BYTE	\$1B	;\$9C - CTRL-escape
.BYTE	\$80	;\$9D - (invalid)
.BYTE	\$FD	;\$9E - CTRL-2
.BYTE	\$80	;\$9F - (invalid)
.BYTE	\$00	;\$A0 - CTRL-comma
.BYTE	\$20	;\$A1 - CTRL-space
.BYTE	\$60	;\$A2 - CTRL-period
.BYTE	\$0E	;\$A3 - CTRL-N
.BYTE	\$80	;\$A4 - (invalid)
.BYTE	\$0D	;\$A5 - CTRL-M
.BYTE	\$80	;\$A6 - (invalid)
.BYTE	\$81	;\$A7 - CTRL-inverse
.BYTE	\$12	;\$A8 - CTRL-R
.BYTE	\$80	;\$A9 - (invalid)
.BYTE	\$05	;\$AA - CTRL-E
.BYTE	\$19	;\$AB - CTRL-Y
.BYTE	\$9E	;\$AC - CTRL-tab
.BYTE	\$14	;\$AD - CTRL-T
.BYTE	\$17	;\$AE - CTRL-W
.BYTE	\$11	;\$AF - CTRL-Q
.BYTE	\$80	;\$B0 - (invalid)
.BYTE	\$80	;\$B1 - (invalid)
.BYTE	\$80	;\$B2 - (invalid)
.BYTE	\$80	;\$B3 - (invalid)
.BYTE	\$FE	;\$B4 - CTRL-delete
.BYTE	\$80	;\$B5 - (invalid)
.BYTE	\$7D	;\$B6 - CTRL-clear
.BYTE	\$FF	;\$B7 - CTRL-insert
.BYTE	\$06	;\$B8 - CTRL-F
.BYTE	\$08	;\$B9 - CTRL-H
.BYTE	\$04	;\$BA - CTRL-D
.BYTE	\$80	;\$BB - (invalid)
.BYTE	\$84	;\$BC - CTRL-CAPS
.BYTE	\$07	;\$BD - CTRL-G
.BYTE	\$13	;\$BE - CTRL-S
.BYTE	\$01	;\$BF - CTRL-A

```
**      TFKD - Table of Function Key Definitions
*
*      Entry n is the ATASCII equivalent of adjusted function key code n.
```

```
TFKD  .BYTE  $1C      ;0 - F1 key
      .BYTE  $1D      ;1 - F2 key
      .BYTE  $1E      ;2 - F3 key
      .BYTE  $1F      ;3 - F4 key

      .BYTE  $8E      ;4 - SHIFT-F1 key
      .BYTE  $8F      ;5 - SHIFT-F2 key
      .BYTE  $90      ;6 - SHIFT-F3 key
      .BYTE  $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      HELPPFG ;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      ;re-initialize for debounce
      STA      KEYDEL
      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      DMSAV   ;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 from interrupt

;      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      ;port B memory control
      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      DMSAV   ;save DMA state
      LDA      #0      ;disable DMA
      STA      SDMCTL  ;DMA control
      BEQ      KIR9    ;exit

```

FDL - Process Display List Interrupt for Fine Scrolling

```
*      ENTRY   ???JSR   FDL
*
*      EXIT     Exits via RTI
*
*      MODS     Original Author Unknown   ???-??-??
*               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
          AND      DRKMSK ;modify with attract-mode luminance
          STA      WSYNC  ;wait for HBLANK synchronization
          STA      COLPF1 ;playfield 1 color/luminance
          PLA      ;restore A
          RTI      ;return from interrupt
```

\$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
;      JMP      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 to caller
```

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 to caller
```

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      #MOTRG0   ;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              ;return to caller
```

OCO - Open Cassette for Output

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

```
OCO    =      *      ;entry

;      Initialize.

      LDA      #$80      ;indicate writing
      STA      WMODE      ;WRITE mode
      LDA      #TONE1      ;tone for ???
      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

;      ???

      LDA      #$60
      STA      DDEVIC
      JSR      SENDEV      ;TELL POKEY TO WRITE MARKS
      LDA      #MOTRGO      ;WRITE 5 SEC 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

OCO1    LDA      BRKKEY      ;BREAK key flag
      BEQ      PBC      ;if BREAK during write leader, process BREAK

      LDA      CDTMF3
      BNE      OCO1      ;if not done waiting

;      Initialize buffer pointer.

      LDA      #0
      STA      BPTR      ;buffer pointer

;      Indicate success.

OCO2    LDY      #SUCCESS      ;indicate success
      RTS      ;return to caller
```

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      FE0F      ;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 to caller
```

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
;
;      ???
;
RCB2     DEC      FE0F      ;set EOF flag
;
;      Exit.
;
RCB3     LDY      #EOFERR   ;end of file indicator
;      RTS      ;return to caller
```

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 to caller

;        Write cassette buffer.

CPB1     LDA     #DTA          ;indicate data record type
        JSR     WCB           ;write cassette buffer
        LDA     #0
        STA     BPTR          ;reset buffer pointer
        RTS                  ;return to caller
```

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 to caller
```

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
;      Cheeck 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 to caller
;      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
;      ???
      LDA      #EOT      ;indicate EOT record type
      JSR      WCB      ;write cassette buffer
      JMP      CCL1      ;exit
```

AUB - Alert User with Beep

```
*      On entry A=frequency
*
*      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      RTCLOK+2      ;current time
;      CLC
;      LDX      PALNTS
;      ADC      BEEPNX,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      RTCLOK+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      BEEFPX,X      ;add constant for ???
;      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 to caller
```

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 to caller
```

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 to caller
```

**** Cassette Handler NTSC/PAL Constant Tables**

WLEADH	.BYTE	high WLEADN	;high NTSC WRITE file leader
	.BYTE	high WLEADP	;high PAL WRITE file leader
WLEADL	.BYTE	low WLEADN	;low NTSC WRITE file leader
	.BYTE	low WLEADP	;low PAL WRITE file leader
RLEADH	.BYTE	high RLEADN	;high NTSC READ file leader
	.BYTE	high RLEADP	;high PAL READ file leader
RLEADL	.BYTE	low RLEADN	;low NTSC READ file leader
	.BYTE	low RLEADP	;low PAL READ file leader
BEEPNX	.BYTE	BEEPNN	;NTSC beep duration
	.BYTE	BEEPNP	;PAL beep duration
BEEPFX	.BYTE	BEEPFN	;NTSC beep separation
	.BYTE	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 to caller
```

```
**      Printer Handler Address Data
*
*      NOTES    Problem: bytes wasted by tables and code. Load Immediate instructions should be used.
```

```
PSTB    .WORD    DVSTAT    ;status buffer address
PPRB     .WORD    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
;      LDY      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
;      RTS                    ;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 to caller
```

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 to caller
```

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 to caller

;      Space fill buffer.

PPB1     LDA      #' '          ;indicate space fill
;          JSR      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                  ;return to caller
```

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        ;return to caller
```

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 to caller
```

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      #SIDEWAY ;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 to caller

;      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
;      JMP      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
;
VCS1     SEC          ;indicate not verified
;      RTS          ;return to caller
```

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      ;initial sum is zero
;      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 to caller
```

** TARV - Table of Address Ranges to Verify

```
TARV     .WORD   $C002,$D000      ;first 8K ROM, $C002 - $CFFF
        .WORD   $5000,$5800      ;first 8K ROM, $D000 - $D7FF
        .WORD   $D800,$E000      ;first 8K ROM, $D800 - $DFFF

        .WORD   $E000,$FFF8      ;second 8K ROM, $E000 - $FFF7
        .WORD   $FFFA,$0000      ;second 8K ROM, $FFFA - $FFFF
```

Second 8K ROM Identification and Checksum

FIX \$FFEE

** Second 8K ROM Identification and Checksum
--

.BYTE	IDDAY, IDMON, IDYEAR	;date (day, month, year)
.BYTE	IDCPU	;CPU series
.BYTE	IDPN1, IDPN2, IDPN3, IDPN4, IDPN5	;part number
.BYTE	IDREV	;revision number
.WORD	\$0000	;reserved for checksum

6502 Machine Vectors

FIX \$FFFA

** 6502 Machine Vectors

.WORD NMI ;vector to process NMI
.WORD RES ;vector to process RESET
.WORD IRQ ;vector to process IRQ

END

Table of Contents

OS, Operating System, Revision 3, Version 4 for 600XL / 800XL / 1450XLD	3
System Symbol Equates	6
System Address Equates	10
Miscellaneous Address Equates	20
Macro Definitions.....	22
FIX - Fix Address.....	22
First 8K ROM Identification and Checksum	23
Interrupt Handler	24
IIH - Initialize Interrupt Handler	24
NMI - Process NMI	24
IRQ - Process IRQ	25
IIR - Process Immediate IRQ	26
BIR - Process BREAK Key IRQ	27
CIR - Continue IRQ Processing	28
XIR - Exit IRQ Processing	28
RIR - Return from Interrupt	29
AIR - Process ACMI IRQ.....	29
WFR - Wait for RESET	30
IVNM - Process Immediate VBLANK NMI	31
PTO - Process Countdown Timer One Expiration	35
PTT - Process Countdown Timer Two Expiration.....	35
DCT - Decrement Countdown Timer	35
SVP - Set Vertical Blank Parameters.....	36
DVNM - Process Deferred VBLANK NMI	36
Initialization	37
PWS - Perform Warmstart	37
RES - Process RESET.....	38
PCS - Perform Coldstart	38
PRS - Preset Memory	39
ICS - Initialize Cartridge Software	44
PAI - Process ACMI Interrupt.....	44
PMI - Perform Miscellaneous Initialization	46
CCE - Check Cartridge Equivalence.....	47
IHW - Initialize Hardware	48
ISW - Initialize Software	49
ADB - Attempt Disk Boot.....	50
ABI - Attempt Boot and Initialize	50
BAI - Boot and Initialize	51
CBI - Complete Boot and Initialize	52

EBL - Execute Boot Loader.....	53
IBS - Initialize Booted Software.....	53
DBE - Display Boot Error Message.....	53
GNS - Get Next Sector.....	54
ACB - Attempt Cassette Boot.....	55
Disk Input/Output	56
IDIO - Initialize DIO.....	56
DIO - Disk I/O	57
SBA - Set Buffer Address.....	58
Relocating Loader	59
RLR - Relocate Routine	59
END - Handles record type of End Record	60
GBY - Jump to the address pointed to by value in GBYTEA.....	61
CAL - Jump to the address pointed to by value in RUNADR.....	61
TEX - Text Record Subroutine handling several types of record	62
FTX - Relocate text into memory	63
WOR - Word Subroutine handling several types of record.....	64
LOO - Low byte and one byte subroutine handling several types of record.....	65
HIG - High byte subroutine handling several types of record	66
Self-test, Part 1	68
SES - Select and Execute Self-test.....	68
Parallel Input/Output	69
GIN - Initialize Generic Parallel Device.....	69
PIO - Parallel Input/Output	70
PIR - Handle Parallel Device IRQ	71
GOP - Perform Generic Parallel Device OPEN	71
GCL - Perform Generic Parallel Device CLOSE.....	71
GGB - Perform Generic Parallel Device GET-BYTE	72
GPB - Perform Generic Parallel Device PUT-BYTE.....	72
GST - Perform Generic Parallel Device STATUS.....	72
GSP - Perform Generic Parallel Device SPECIAL.....	72
SNP - Select Next Parallel Device	73
IPH - Invoke Parallel Device Handler.....	73
EPC - Execute Parallel Device Handler Command	74
Peripheral Handler Loading Facility, Part 1	76
PHL - Load and Initialize Peripheral Handler.....	76
Self-test, Part 2	77
Peripheral Handler Loading Facility, Part 2	82
CLT - Checksum Linkage Table.....	82
International Character Set	83
Self-test, Part 3	85

STH - Self-test Hardware	85
EMS - Execute Memory Self-test	85
EST - Execute Self-test	85
SEL - Self-test	86
IST - Initialize Self-test	87
SDL - Set Up Display List	87
PMD - Process Main Screen DLI	88
STM - Self-test Memory	92
DFS - Display First ROM Status	94
DSS - Display Second ROM Status	95
SLD - Set LED's	95
DMW - Delay a Middling While	95
DLW - Delay a Long While	95
DAW - Delay a While	96
DRS - Display RAM Block Status	96
POD - Process Other DLI's	97
STK - Self-test Keyboard	99
SBT - Sound Beep Tone	101
SAS - Silence All Sounds	101
KSB - Process Keyboard Space Bar Display	101
KBK - Process Keyboard Backspace Key Display	101
KTK - Process Keyboard Tab Key Display	101
KRK - Process Keyboard Return Key Display	102
STV - Self-test Audio-visual	103
SVN - Sound Note	105
DVN - Display ???	106
AST - Add Sixteen	106
SVR - Set Value in Range	108
SSM - Set Screen Memory	109
SUC - Set Up Colors	109
Floating Point Package	112
FPP - Floating Point Package	112
AFP - Convert ASCII to Floating Point	113
FASC - Convert Floating Point Number to ASCII	116
IFP - Convert Integer to Floating Point Number	118
FPI - Convert Floating Point Number to Integer	119
ZFR0 - Zero FR0	120
ZF1 - Zero Floating Point Number	120
ZXLY - Zero Page Zero Location X for Length Y	121
ILP - Initialize Line Buffer Pointer	121
SIL - Shift Integer Left	121
FSUB - Perform Floating Point Subtract	122
FADD - Perform Floating Point Add	123
FMUL - Perform Floating Point Multiply	125

FDIV - Perform Floating Point Divide	127
GNC - Get Next Character	128
ICX - Increment Character Offset.....	128
SLB - Skip Leading Blanks.....	129
TNC - Test for Numeric Character	129
TVN - Test for Valid Number Character.....	130
S2L - Shift FR2 Left One Digit.....	131
S0L - Shift FR0 Left One Digit.....	131
SML - Shift Mantissa Left One Digit	131
NORM - Normalize FR0	131
N0E - Normalize FR0/FRE	132
S0R - Shift FR0 Right.....	133
S1R - Shift FR1 Right.....	133
SRR - Shift Register Right.....	133
S0ER - Shift FR0/FRE Right	134
C0A - Convert FR0 to ASCII	134
TDP - Test for Decimal Point.....	135
SNL - Store Number in Line Buffer	135
SAL - Store ASCII Character in Line Buffer	135
FNZ - Find Last Non-zero Character in Line Buffer	136
GND - Get Next Digit.....	136
DLP - Decrement Line Buffer Pointer.....	137
SUE - Set Up Exponent for Multiply or Divide.....	137
SUP - Set Up for Multiply or Divide	137
FRA10 - Add FR1 to FR0.....	138
FRA20 - Add FR2 to FR0	138
FRA1E - Add FR1 to FRE	138
F1R - Add FR1 to Register.....	138
FRA2E - Add FR2 to FRE	139
F2R - Add FR2 to Register.....	139
FARR - Add Register to Register	139
M12 - Move FR1 to FR2.....	140
M0E - Move FR0 to FRE.....	140
PLYEVL - Evaluate Polynomial.....	141
FLD0R - ???	142
FLD0P - ???	142
FLD1R - ???	142
FLD1P - ???	143
FST0R - ???	143
FST0P - ???	143
FMOVE - Move FR0 to FR1	144
EXP - ???	144
EXP10 - ???	145
XFORM - $Z = (X-C)/(X+C)$	147

LOG - Indicate Base e Logarithm	147
LOG10 - Indicate Base 10 Logarithm	147
LOGS - Compute Logarithm	148
LOGC - Complete Computation of Logarithm.....	149
LOGQ - ???	151
Domestic Character Set	152
Device Handler Vector Tables	154
Jump Vectors	156
Generic Parallel Device Handler Vector Table	157
\$E4C0 Patch	158
Central Input/Output.....	159
ICIO - Initialize CIO	159
IIN - Indicate IOCB Not Open Error	159
CIO - Central Input/Output	160
IND - Indicate Nonexistent Device Error	160
LHO - Load Peripheral Handler for OPEN	161
PCC - Perform CIO Command.....	161
XOP - Execute OPEN Command	162
IIO - Initialize IOCB for OPEN	162
PPO - Poll Peripheral for OPEN	163
XCL - Execute CLOSE Command	163
XSS - Execute STATUS and SPECIAL Commands	163
XGT - Execute GET Command	164
XPT - Execute PUT Command	166
SSC - Set Status and Complete Operation	167
CCO - Move IOCB in zero page back to user area	167
CEP - Compute Handler Entry Point.....	168
DBL - Decrement Buffer Length.....	168
DBP - Decrement Buffer Pointer	169
IBP - Increment Buffer Pointer	169
SFL - Set Final Buffer Length.....	169
EHC - Execute Handler Command.....	169
IDH - Invoke Device Handler.....	170
SHT - Search Handler Table.....	170
FDH - Find Device Handler	171
Peripheral Handler Loading Facility, Part 3	172
PHR - Peripheral Handler Routine	172
PHP - Peripheral Handler Polling subroutine.....	174
LPH - Load Peripheral Handler.....	175
PHG - Perform Peripheral Handler GET-BYTE	176
GNL - Get Next Load Block.....	177

SHC - Search Handler Chain	178
PHW - Handler Initialization Subroutine	179
PHI - Full Initialization Entry	180
PHC - Peripheral Handler Main Entry	180
PHQ - Handler Initialization Continued	181
PHX - Initialize Handler	182
\$E912 Patch	183
Peripheral Handler Loading Facility, Part 4	184
PHU - Perform Peripheral Handler Unlinking	184
\$E959 Patch	186
Serial Input/Output	187
ISIO - Initialize SIO	187
SIO - Serial Input/Output	188
CSO - Complete SIO Operation	190
WCA - Wait for Completion or ACK	191
SEN - Sends a Buffer over the Serial Bus	192
ORIR - Process Serial Output Ready IRQ	193
OCIR - Process Serial Output Complete IRQ	194
REC - Receive	195
ITO - Indicate Timeout	195
IRIR - Process Serial Input Ready IRQ	196
SBP - Set Buffer Pointers	197
PCI - Process Cassette I/O	198
PTE - Process Timer Expiration	199
ESS - Enable SIO SEND	199
ESR - Enable SIO RECEIVE	199
SSR - Set for SEND or RECEIVE	200
DSR - Disable SEND and RECEIVE	200
GTO - Get Device Timeout	201
SID - Send to Intelligent Device	201
STW - Set Timer and Wait	202
CBR - Compute Baud Rate	203
AVV - Adjust VCOUNT Value	204
SBR - Set Initial Baud Rate	205
PBK - Process BREAK Key	206
SSV - Set SIO VBLANK Parameters	206
Keyboard, Editor and Screen Handler, Part 1	208
Peripheral Handler Loading Facility, Part 5	211
PHE - Perform Peripheral Handler Entry	211
PHO - Perform Peripheral Handler Poll at OPEN???	213
PTL - Put byte entry for provisionally opened IOCB's	214

\$EF6B Patch	216
Keyboard, Editor and Screen Handler, Part 2	217
SIN - Initialize Screen.....	217
SOP - Perform Screen OPEN.....	217
EOP - Perform Editor OPEN.....	217
COC - Complete OPEN Command.....	218
SGB - Perform Screen GET-BYTE.....	222
GDC - Get Data under Cursor.....	222
SPB - Perform Screen PUT-BYTE.....	223
CEL - Check End of Line.....	223
PLO - Plot Point.....	224
SPQ - ???.....	224
SEC - Set Exit Conditions	225
SST - Perform Screen STATUS	225
\$F223 Patch	226
Keyboard, Editor and Screen Handler, Part 3	227
ESP - Perform Editor SPECIAL	227
ECL - Perform Editor CLOSE.....	227
EGB - Perform Editor GET-BYTE.....	228
IRA - Invoke Routine Pointed to by ADDRESS.....	229
EPB - Perform Editor PUT-BYTE.....	229
PCH - Process Character.....	230
IGN - Ignore Character and Perform Keyboard GET-BYTE.....	230
KGB - Perform Keyboard GET-BYTE.....	231
ESC - Escape.....	234
CUP - Move Cursor Up	234
CDN - Move Cursor Down	234
CLF - Move Cursor Left.....	234
CRM - Move Cursor to Right Margin.....	235
SCC - Set Cursor Column.....	235
CRT - Move Cursor Right.....	235
CLM - Move Cursor to Left Margin.....	235
CSC - Clear Screen.....	236
CHM - Move Cursor Home.....	236
BSP - Backspace	237
TAB - Tab.....	237
STB - Set Tab.....	237
CTB - Clear Tab.....	238
ICH - Insert Character.....	238
DCH - Delete Character	239
ILN - Insert Line.....	239
ILN1 - Insert Line Continued	239
DLN - Delete Line.....	240

DLN1 - Delete Line Continued	240
BEL - Sound Bell	240
CBT - Move Cursor to Bottom	241
DDD - Perform Double Byte Double Decrement	241
SDF - Store Data Indirect for Fine Scrolling	241
SDI - Store Data Indirect	241
DSD - Perform Double Byte Single Decrement	242
DBS - Perform Double Byte Subtract	242
SSE - Set Scrolling Display List Entry	243
CCA - Convert Cursor Row/Column to Address	244
SZA - Set Zero Data and Advance Cursor Column	245
SEA - Set EOL Data and Advance Cursor Column	245
SDA - Set Data and Advance Cursor Column	245
ACC - Advance Cursor Column	246
RWS - Return with Scrolling	246
RET - Return	247
SEP - Subtract ENDPT from ROWAC or COLAC	247
CRE - Check Cursor Range for Editor	248
CCR - Check Cursor Range	249
ROD - Restore Old Data under Cursor	250
BMI - Initialize for Bit Map Operation	250
BLR - Rotate Logical Line Bit Map Left	250
BMP - Put Bit in Bit Map	250
BMS - Set Bit in Bit Map	251
BMC - Clear Bit in Bit Map	251
BLG - Get Bit from Logical Line Bit Map	251
BLG1 - Get Bit from Logical Line Bit Map Continued 1	251
BLG2 - Get Bit from Logical Line Bit Map Continued 2	251
BMG - Get Bit from Bit Map	252
CIA - Convert Internal Character to ATASCII	252
MLN - Move Line	253
ELL - Extend Logical Line	254
CLN - Clear Line	254
SCR - ???	255
SSD - Scroll Screen for Delete	256
SLC - Set Logical Column	257
CBC - Compute Buffer Count	258
SBS - Set Buffer Start and Logical Column	259
DQQ - Deletes a Line if it is empty and an Extension	259
DWQ - ???	259
CCC - Check for Control Character	260
SRC - Save Row and Column	260
RRC - Restore Row and Column	260
SWA - If mixed mode, swap ?????? cursors with regular cursors	261

SKC - Sound Key Click	261
SCL - Set Cursor at Left Edge	262
SMS - Set Memory Scan Counter Address	262
SSP - Perform Screen SPECIAL	263
KIR - Process Keyboard IRQ	272
FDL - Process Display List Interrupt for Fine Scrolling.....	274
\$FCD8 Patch.....	275
Cassette Handler	276
CIN - Initialize Cassette	276
CSP - Perform Cassette SPECIAL	276
COP - Perform Cassette OPEN	276
OCI - Open Cassette for Input	277
ICR - Initiate Cassette READ	277
PBC - Process BREAK for Cassette Operation	277
OCO - Open Cassette for Output	278
CGB - Perform Cassette GET-BYTE	279
RCB - Read Cassette Block	279
CPB - Perform Cassette PUT-BYTE	280
CST - Perform Cassette STATUS	280
CCL - Perform Cassette CLOSE	281
AUB - Alert User with Beep	282
WFK - Wait for Key	283
SCB - Perform SIO on Cassette Buffer	283
WCB - Write Cassette Buffer	283
Printer Handler	285
PIN - Initialize Printer	285
PST - Perform Printer STATUS	285
PSP - Perform Printer SPECIAL	285
POP - Perform Printer OPEN	286
PPB - Perform Printer PUT-BYTE	286
FPB - Fill Printer Buffer	287
PPP - Perform Printer PUT	287
PCL - Perform Printer CLOSE	287
SDP - Set Up DCB for Printer	288
STS - Set Printer Timeout from Status	288
PPM - Process Print Mode	289
Self-test, Part 4	290
VFR - Verify First 8K ROM	290
VCS - Verify Checksum	290
VSR - Verify Second 8K ROM	290
CRR - Checksum Region of ROM	291
Second 8K ROM Identification and Checksum	292

6502 Machine Vectors.....	293
---------------------------	-----

Indexes

C

Call to a macro

FIX . 22, 83, 85, 112, 115, 118, 120, 121, 122, 124, 126, 140, 141, 142, 143, 144, 147, 152, 154, 156, 157, 158, 183, 186, 216, 226, 275, 292, 293

E

Equates

Listings

ACMI Address Equates	18
ANTIC Address Equates	18
Assembly Option Equates	6
Bus Command Equates	8
Bus Response Equates.....	8
Cartridge Address Equates.....	16
Cassette Record Type Equates.....	8
Cassette Timing Equates.....	8
Character and Key Code Equates.....	7
Command Auxiliary Byte Equates	8
Configuration Equates	6
CTIA/GTIA Address Equates	16
DCB Device Bus ID Equates	7
Device Code Equates	7
Device Handler Vector Table Address Equates.....	19
Floating Point Package Address Equates [1].....	15
Floating Point Package Address Equates [2].....	18
Floating Point Package Miscellaneous Equates.....	8
Floating Point Package Page Zero Address Equates	11
Generic Parallel Device Handler Vector Table Address Equates	19
ICAX1 Auxiliary Byte 1 Equates	7
Identification Equates	6
IOCB Command Code Equates	6
Jump Vector Address Equates	19
LNBUG Address Equates.....	16
Miscellaneous Equates	9
Page Five Address Equates	15
Page Four Address Equates.....	15
Page One (Stack) Address Equates	12
Page Six Address Equates	15
Page Three Address Equates	14
Page Two Address Equates	12
Page Zero Address Equates.....	10
Parallel Device Address Equates	19
PBI Address Equates	17
PBI RAM Address Equates	18
PIA Address Equates	18
POKEY Address Equates	17

Power-up Validation Byte Value Equates	9
Relocating Loader Miscellaneous Equates	9
Self-test Address Equates	21
Self-test Page Zero Address Equates	20
Special Entry Command Equates	6
Status Code Equates	7
Values	
ABUFPT	10, 26
ACK	8, 9, 188, 189, 191, 202
ACMI	6, 10, 14, 15, 16, 18, 24, 26, 29, 31, 42, 44
ACMISR	14, 29, 42
ACMVAR	4, 15, 40, 48
ADRESS	11, 218, 219, 220, 221, 222, 224, 229, 230, 236, 239, 241, 242, 244, 254, 256, 259, 262
AFP	18, 112, 113
ALLPOT	17
AMFLAG	18, 42
AMINIT	18, 42
AMISRA	18, 42
AMNAME	18, 42
AMVTAD	18, 42
ANTIC	18, 20, 48, 87, 208, 219, 220, 221
APPEND	7
APPMHI	10, 40, 242
ATACHR	14, 223, 224, 227, 228, 229, 231, 233, 237, 252, 260, 265
ATTRACT	10, 27, 31, 88, 97, 273
AUDC1	17, 101, 105, 200
AUDC2	17, 98, 101, 200
AUDC3	17, 48, 101, 200
AUDC4	17, 48, 101
AUDCTL	18, 48, 86, 200
AUDF1	17, 101, 105, 199
AUDF2	17, 98, 199
AUDF3	17, 188, 198, 206, 278
AUDF4	17, 188, 198, 206, 278
B00600	9, 198, 276, 278
B19200	9, 188
BADIOC	7, 160, 214
BADMOD	7, 218
BASICF	15, 41, 46
BEEPFN	8, 284
BEEPPF	8, 284
BEEPNN	8, 284
BEEPNP	8, 284
BFENHI	10, 188, 191, 193, 197
BFENLO	10, 188, 191, 193, 197
BITMSK	11, 250, 251, 252
BLIM	13, 277, 279
BLKBDV	19, 156

BOOT?	10, 50, 53, 55
BOOTAD	12, 52, 53
BOTSCR	13, 220, 234, 239, 248, 253, 255, 261
BPTR	10, 277, 278, 279, 280, 281
BRKABT	7, 206, 231, 249, 277
BRKKEY	10, 27, 49, 192, 195, 205, 206, 231, 232, 249, 277, 278
BRKKY	12, 29, 49
BUFADR	10, 57, 58
BUFCNT	11, 228, 258
BUFRFL	10, 195, 196, 197
BUFRHI	10, 188, 191, 193, 196, 197, 206
BUFRLO	10, 188, 191, 192, 193, 196, 197, 206
BUFSTR	11, 228, 236, 247, 258, 259
CART	16, 43, 46
CARTAD	16, 44, 46
CARTCK	15, 47
CARTCS	16, 39, 44
CARTFG	16, 43, 46
CASBUF	15, 50, 52, 176, 177, 279, 280, 281, 283
CASET	7, 188, 199, 200
CASETV	19, 45, 49, 154, 155
CASFLG	14, 188, 195, 198
CASINI	10, 55
CASSBT	15, 51, 52, 54, 55
CASSET	7, 45
CAUX1	12, 188
CAUX2	12, 188
CBAUDH	14, 203, 206, 276
CBAUDL	14, 203, 206, 276
CCOMND	12, 188
CDEVIC	12, 188
CDTMA1	12, 35, 206
CDTMA2	12, 35
CDTMF3	12, 32, 277, 278
CDTMF4	12
CDTMF5	12
CDTMV1	12, 35, 36, 45
CDTMV2	12, 45
CDTMV3	12, 32, 45
CDTMV4	12, 45
CDTMV5	12, 45
CH	14, 33, 86, 87, 99, 217, 230, 231, 273
CH1	14, 272, 273
CHACT	14, 32, 42, 218
CHACTL	14, 18, 32, 42
CHAR	14, 222, 224, 252
CHBAS	14, 32, 42, 218, 273
CHBASE	14, 18, 32, 42

CHKERR.....	7, 196
CHKSNT.....	10, 192, 193, 194
CHKSUM.....	4, 10, 192, 193, 195, 196, 206
CHLINK.....	15, 172, 178, 184
CHSALT	13, 218, 273
CIOCHR.....	10, 160, 164, 165, 166, 167
CIOINV	19, 49, 156
CIOV	19, 43, 53, 156
CIX.....	11, 113, 114, 115, 116, 117, 128, 129, 130
CKEY.....	15, 49, 55
CLOSE	6, 71, 154, 155, 157, 160, 161, 163, 227, 281, 287
CLS.....	7, 223
CMCMD	10
CNTL1.....	7, 33, 272
CNTLF1.....	7, 33, 272
CNTLF2.....	7, 33, 273
CNTLF3.....	7
CNTLF4.....	7, 33, 273
COLAC.....	11, 247, 264, 265
COLBK	17
COLCRS	11, 209, 228, 234, 235, 236, 237, 239, 240, 244, 246, 247, 249, 254, 257, 258, 262, 263, 265
COLDST	12, 37, 40, 43, 68, 184
COLDSV.....	19, 156
COLINC	14, 263, 264, 265
COLOR0	13, 109, 218
COLOR1	13, 31, 109
COLOR2	13, 109, 274
COLOR3	13
COLOR4	13, 97, 109, 219
COLPF0	17, 88
COLPF1	17, 31, 97, 274
COLPF2	17, 97
COLPF3	17
COLPM0	17, 32
COLPM1	17
COLPM2	17
COLPM3	17
COLRSH	11, 31, 32, 274
COMPLT.....	8, 191
CONSOL.....	16, 32, 46, 49, 86, 88, 89, 97, 98, 261, 282
COUNTR.....	11, 256, 257, 264, 266
CRETRI.....	9, 188, 189
CRETRY.....	13, 188, 189
CRITIC	10, 31, 70, 74, 188, 190
CRSINH	14, 27, 218, 225
CRSROR	7, 249
CSOPIV.....	19, 55, 156
CTIA.....	16, 48

CTIM	9, 201
DATAER.....	9, 59, 60
DAUX1	14, 50, 52, 174, 177, 188, 285, 289
DAUX2	14, 50, 174, 188, 198, 283
DBSECT.....	12, 52
DBUFHI.....	14, 50, 57, 58, 197, 283, 288
DBUFLO.....	14, 50, 57, 58, 197, 283, 288
DBUFSZ.....	9, 289
DBYTHI.....	14, 57, 58, 197, 283, 288
DBYTLO.....	14, 57, 58, 197, 283, 288
DCB	7, 14, 172, 174, 177, 285, 287, 288, 289
DCOMND	14, 50, 54, 57, 58, 188, 283, 285, 288, 289
DCSORG	6, 42, 152, 218
DDEVIC	14, 57, 177, 188, 199, 200, 278, 283, 288
DELTAC.....	11, 263, 264, 265
DELTAR.....	11, 263, 264
DERRF.....	15, 40, 222
DERROR.....	7, 191
DFLAGS.....	12, 52
DIGRT.....	11, 113, 114, 115
DINDEX	11, 218, 219, 220, 221, 225, 244, 246, 247, 248, 249, 252, 262
DINITV.....	19, 49, 156
DIRECT.....	7
DISK	7
DISKID	7, 57
DISKRD	17
DISKWR.....	17
DISPLY	7, 45
DLISTH	12, 18, 32
DLISTL.....	12, 18, 32
DMACTL	12, 18, 32, 87
DMASAV.....	14, 273
DMASK.....	13, 222, 224, 244
DMST	17
DNACK.....	7, 191
DOSINI.....	10, 52, 53, 55
DOSVEC.....	10, 39, 40, 44
DOUBLE.....	8, 289
DRAWLN	6
DRETRI.....	9, 188
DRETRY	13, 188, 189
DRKMSK.....	10, 31, 32, 274
DSCTLN	14, 56, 57
DSCTSZ.....	6, 56
DSKCNT	12
DSKFMS	10
DSKINV	19, 50, 54, 156
DSKTIM	12, 56, 57

DSKUTL.....	10
DSPFLG	14, 230
DSTAT	10, 218, 221, 225, 227, 228, 232, 241, 242, 249
DSTATS	14, 57, 58, 70, 189, 190, 198, 283, 288
DT1	8, 279, 281
DTA	8, 280
DTIMLO.....	14, 57, 201, 283, 288
DUNIT	14, 50, 54, 70, 188, 283, 288
DUNUSE	14
DVSTAT	14, 57, 162, 172, 173, 174, 175, 179, 180, 181, 213, 285, 288
EDITRV	19, 45, 49, 154
EEXP	11, 113, 114, 115, 117, 126, 137
ENDPT	11, 247, 264, 265
ENTVEC	10
EOFERR	7, 232, 279
EOL.....	7, 45, 164, 165, 166, 223, 228, 230, 232, 245, 246, 247, 286, 287
EOT.....	8, 279, 281
ERRFLG.....	12, 188, 189, 191
ERROR.....	8, 191
ESCFLG	13, 230, 234
ESIGN	11, 114, 115
EXP	19, 144, 148
EXP10	19, 144, 145
FADD	18, 122, 123, 141, 147, 149
FALSE	6
FASC.....	18, 115, 116
FCHFLG	11, 113, 114
FDIV.....	18, 126, 127, 146, 147, 149, 150
FEOF	10, 277, 279
FILDAT	14, 265
FILFLG	13, 263, 265
FILLIN	6
FINE.....	13, 218, 220, 227, 241, 255
FKDEF	11, 217, 233
FLD0P	19, 142
FLD0R.....	19, 141, 142, 145, 146, 147
FLD1P	19, 142, 143
FLD1R.....	19, 141, 142, 144, 147, 149, 150
FLPTR	12, 142, 143
FMOVE.....	19, 141, 143, 144, 145, 146, 149
FMPREC.....	8, 113, 123, 124, 127, 132, 133, 134, 137, 145, 149
FMSZPG	10
FMUL	18, 124, 125, 126, 127, 141, 144, 145, 146, 149
FNCNOT	7, 73, 169, 215
FOMAT	8, 57, 58
FPI	18, 118, 119, 145
FPREC	8, 123, 125, 127, 128, 138, 139, 140, 141, 142, 143, 144, 145, 146, 150
FPSCR	15, 145, 147, 149

FPSCR1.....	15
FPTR2.....	12, 141, 147
FR0 . 11, 18, 113, 115, 116, 117, 118, 119, 120, 122, 123, 124, 125, 126, 127, 128, 131, 132, 133, 134, 136, 137, 138, 140, 141, 142, 143, 144, 145, 147, 148, 149, 151	
FR0M.....	11, 113, 123, 124, 131, 132, 134
FR1	11, 122, 123, 125, 127, 128, 133, 137, 138, 140, 141, 142, 143, 144, 145, 146, 151
FR1M.....	11, 123, 124, 145, 146, 149
FR2	11, 125, 127, 131, 137, 138, 139, 140
FRE	11, 125, 127, 128, 131, 132, 134, 137, 138, 139, 140
FREQ.....	10, 282
FRMADR.....	11, 253
FRMERR	7, 196
FRSIGN	11, 137
FRX	11, 114, 115, 131, 136, 137
FSTOP.....	19, 143
FSTOR.....	19, 141, 143, 145, 147, 149
FSUB.....	18, 121, 122, 145, 147
FTYPE.....	10, 55, 276, 283
GBYTEA	14, 59, 61, 175
GETCHR.....	6
GETDAT	9, 57, 283
GETREC	6
GINTLK.....	15, 24, 32, 37
GPDVV	19, 49, 157
GPRIOR.....	13, 32, 219
GRCTL.....	17
GRAFM.....	16
GRAFP0.....	16
GRAFP1.....	16
GRAFP2.....	16
GRAFP3.....	16
HATABS.....	6, 14, 42, 45, 168, 171, 211, 212
HDR	8
HELP	7, 14, 33, 87, 91, 97, 272
HELPG.....	14, 87, 97, 272
HIBYTE	13, 59, 62, 63, 65, 66
HITCLR.....	17
HITONE	9, 199
HIUSED	14, 63
HNDLOD	14, 160, 162, 167, 214
HOLD1	11, 219, 221, 240, 257, 259
HOLD2.....	13
HOLD3.....	13, 247
HOLD4.....	13, 265
HOLDCH	11, 231, 233
HPOSM0	16
HPOSM1	16
HPOSM2	16

HPOSM3	16
HPOSP0	16
HPOSP1	16
HPOSP2	16
HPOSP3	16
HSCROL	18
ICAX1	7, 15, 43
ICAX1Z	10, 164, 166, 217, 220, 222, 231, 276
ICAX2	15
ICAX2Z	10, 217, 276, 289
ICBAH	15, 43, 53, 167
ICBAHZ	10, 167, 213
ICBAL	15, 43, 53, 167
ICBALZ	10, 164, 165, 166, 167, 169, 170, 213
ICBLH	15, 169
ICBLHZ	10, 169
ICBLL	15, 53, 169
ICBLLZ	10, 164, 166, 168, 169
ICCOM	15, 43, 53
ICCOMT	10, 76, 161, 162, 168
ICCOMZ	10, 160, 161, 164, 166, 263
ICDNO	15, 286
ICDNOZ	10, 170, 213, 286, 288
ICHID	15, 76, 159, 163
ICHIDZ	10, 76, 160, 162, 163, 168, 171, 213
ICIDNO	10, 76, 160, 163, 167, 169, 170, 213, 214
ICPTH	15, 159
ICPTHZ	10, 162, 163, 213, 215
ICPTL	15, 159
ICPTLZ	10, 162, 163, 213, 215
ICSORG	6, 83, 218, 273
ICSPR	15, 76, 213
ICSPRZ	10, 160, 162, 167, 168, 170
ICSTA	15
ICSTAZ	10, 163, 165, 167, 169
IDCPU	6, 23, 292
IDDAY	6, 23, 292
IDMON	6, 23, 292
IDPN1	6, 23, 292
IDPN2	6, 23, 292
IDPN3	6, 23, 292
IDPN4	6, 23, 292
IDPN5	6, 23, 292
IDREV	6, 23, 292
IDYEAR	6, 23, 292
IFP	18, 118, 145, 149
IMASK	13
INBUFF	11, 113, 116, 117, 121, 128, 129, 137

INIML	6, 49
INSCLR	7
INSDAT.....	11, 238, 245, 246, 247
INTABS.....	12, 27, 29, 41, 45
INTEMP	12, 36
INTINV.....	19, 49, 156
INTZBS	10, 40
INVFLG.....	13, 232, 233
IOCB	6, 7, 9, 10, 15, 43, 53, 76, 159, 160, 162, 163, 164, 166, 167, 168, 169, 170, 213, 214, 215
IOCBAS	10, 160, 167
IOCBSZ	6, 10, 159
IOCFRE	9, 159, 162, 163
IPDIMK	4, 13, 26
IPDVI.....	4, 17, 26
IRQEN	10, 18, 26, 27, 193, 194, 200, 218
IRQST	17, 26
JMPERS	14
JVECK.....	13, 27, 28
KBCODE.....	17, 33, 272
KBD	7, 45
KEYBDV	19, 45, 49, 154, 283
KEYDEF	11, 217, 231
KEYDEL	14, 33, 272, 273
KEYDIS	13, 27, 33, 272
KEYREP	14, 33, 41
KRPDEL.....	14, 41, 273
LBPR1	15
LBPR2	15, 116
LBUFF	15, 116, 117, 121, 135, 136
LCOUNT.....	12, 59, 62, 63, 66
LEDGE.....	6, 41
LMARGN	11, 41, 222, 234, 235, 237, 239, 254, 258, 259, 262
LNBUG	6, 10, 16, 24, 25, 39
LNFLG	10, 24, 25, 39
LNIRQ	16, 25
LNNMI.....	16, 24
LNORG.....	16, 39
LOADAD.....	14, 60, 62, 64, 65, 66, 175
LOG	19, 147, 149, 150
LOG10	19, 144, 147, 149, 150
LOGCOL.....	11, 228, 236, 237, 238, 239, 246, 257, 258, 259
LOGMAP	13, 236, 240, 246, 247, 250, 251, 254
LOTONE	9, 199
LPENH	12, 32
LPENV	12, 32
LTEMP	10, 63, 64, 65, 66
MOPF.....	16
M0PL.....	16

M1PF	16
M1PL	16
M2PF	16
M2PL	16
M3PF	16
M3PL	16
MAXDEV.....	4, 6, 168, 171, 211
MAXIOC	6, 159, 160, 214
MEMERR.....	9, 63
MEMLO	6, 14, 49, 172, 173, 179, 181, 184
MEMTOP	14, 49, 62, 63, 173, 221
MINTLK	15, 24, 31
MLTTMP	11, 244
MOTRGO.....	9, 198, 277, 278
MOTRST	9, 187, 198, 206, 281
MXDMOD	7, 220
NACK	8
NBUFSZ	9, 289
NCOMHI	9, 187, 195, 206
NCOMLO	9, 188
NEWADR.....	13, 62, 63, 64, 65, 66
NEWCOL	14, 263
NEWROW	14, 263
NGFLAG.....	10, 39, 40, 42
NMIEN	18, 24, 86, 87, 218, 227
NMIREs.....	18, 24
NMIST	18, 24
NOCKSM.....	10, 191, 197
NOCLIK.....	14, 231, 232
NODAT	9
NONDEV	7, 74, 160, 171, 213, 214
NORMAL	8, 239, 289
NOTOPN	7, 159, 168
NRINGS	13
NSIGN.....	11, 114, 115
NVALID	7, 161, 263
OLDADR	11, 13, 244, 250
OLDCHR	11, 13, 225, 250
OLDCOL	11, 13, 263
OLDPAR.....	14
OLDROW	11, 13, 224, 263, 264
OPEN	6, 15, 40, 43, 71, 76, 154, 155, 157, 161, 162, 163, 213, 214, 217, 218, 222, 227, 248, 276, 286
OPNIN	7, 43
OPNOT.....	7, 43
OPNTMP	11, 219, 220
OVRRUN	7, 196
P0PF	16
P0PL	16

P1PF	16
P1PL.....	16
P2PF	16
P2PL.....	16
P3PF	16
P3PL.....	16
PACTL	18, 28, 48, 187, 198, 206, 277, 278, 281
PADDL0.....	13, 34
PADDL1.....	13
PADDL2.....	13
PADDL3.....	13
PADDL4.....	13, 34, 41
PADDL5.....	13
PADDL6.....	13
PADDL7.....	13
PAL	8, 11, 16, 41, 198, 207, 284
PALNTS	11, 41, 198, 203, 204, 277, 278, 282
PARMBL	14, 59
PBCTL	18, 28, 48, 187, 188, 195, 206
PBI	3, 17, 18
PBICTL	13, 17
PBIRAM	18
PBPNT.....	14, 286, 287
PBUFSZ.....	14, 285, 286, 287, 288, 289
PCOLR0	13, 32
PCOLR1	13
PCOLR2	13
PCOLR3	13
PDEVN	7, 288
PDID1	19, 69
PDID2	19, 69
PDIMSK.....	4, 12, 26
PDIOV	19, 70
PDIRQV	19, 71
PDVI	4, 17, 26
PDVMSK.....	12, 70, 73
PDVS.....	17, 37, 48, 69, 70, 71, 73, 74
PDVV.....	19, 69, 73
PENH	18, 32
PENV	18, 32
PHENTV	19, 156
PHINIV	19, 156
PHUNLV	19, 156
PIA	9, 18, 48
PLOT	8
PLYARG.....	15, 141, 147
PLYCNT.....	11, 141
PLYEVL.....	19, 140, 141, 145, 149

PMBASE.....	18
PMCMD	17
PMCTL	17
PMDATA.....	17
PMSTAT.....	17
POKEY	9, 12, 17, 26, 27, 45, 48, 86, 187, 199, 200, 203, 206, 207, 278
POKMSK	10, 26, 27, 193, 194, 199, 200, 218
PORTA	18, 28, 33, 48
PORTB	4, 18, 28, 40, 41, 42, 46, 48, 68, 95, 272, 273
POT0.....	17, 34
POT1.....	17
POT2.....	17
POT3.....	17
POT4.....	17
POT5.....	17
POT6.....	17
POT7.....	17
POTGO.....	18, 34
PPTMPA	12, 73, 74
PPTMPX	12, 73, 74
PRINTR	7, 45
PRINTV.....	19, 45, 49, 154
PRIOR	13, 17, 32
PRNBUF.....	15, 285, 286, 287
PRVOPN.....	7, 162
PTIMOT	14, 285, 288
PTRIG0.....	13, 34
PTRIG1.....	13, 34
PTRIG2.....	13
PTRIG3.....	13
PTRIG4.....	13, 34, 41
PTRIG5.....	13, 34
PTRIG6.....	13
PTRIG7.....	13
PUPBT1.....	15, 38, 41
PUPBT2.....	15, 38, 41
PUPBT3.....	15, 38, 41
PUPDIV	19, 156
PUPVL1.....	9, 38, 41
PUPVL2.....	9, 38, 41
PUPVL3.....	9, 38, 41
PUTCHR.....	6, 162
PUTDAT.....	9, 57, 283
PUTREC	6, 53
PUTSEC	8, 57
QTEMP	11, 127, 128
RAMLO.....	10, 39, 40, 52, 53
RAMSIZ	14, 43, 49, 94, 217

RAMSYS	6, 47
RAMTOP	11, 217, 219, 220, 221, 236, 253, 256, 257
RANDOM.....	17
RBLOKV	19, 54, 156
RDONLY	7, 166
READ.....	8, 54, 198, 216, 277, 279, 283, 284, 288
RECLN	12, 59, 60, 62, 63, 64, 65, 66
RECVDN.....	10, 195, 196
REDGE	6, 41
RELADR.....	12, 62
RIRGHI	8, 198
RIRGLN	8, 207
RIRGLP.....	8, 207
RLEADN	8, 284
RLEADP	8, 284
RMARGN	11, 41, 235, 237, 246, 249, 254, 258, 259
ROWAC.....	11, 247, 264
ROWCRS11, 209, 224, 228, 234, 236, 237, 238, 239, 240, 244, 246, 247, 249, 251, 253, 254, 257, 258, 259, 260, 261, 263, 264, 265	
ROWINC	14, 263, 265
RSIRGN	8, 207
RSIRGP	8, 207
RTCLOK.....	10, 31, 205, 282
RUNADR	14, 59, 60, 61
SAVADR	11, 220, 221, 239
SAVIO.....	14, 205
SAVMSC.....	11, 219, 221, 244, 262
SBUFSZ	9, 289
SCREDT.....	7, 45
SCRENV	19, 45, 49, 154
SCRFLG	13, 238, 247
SCRMEM	7, 242
SDLSTH.....	12, 32, 87
SDLSTL.....	12, 32, 87, 221
SDMCTL	12, 14, 32, 87, 218, 222, 273
SEIOCB	6, 43, 53
SENDEV	19, 156, 278
SERIN	17, 196
SEROUT	18, 48, 192, 193
SETVBV	19, 156, 206, 277, 278
SGNFLG	11, 145, 146, 148, 149, 150
SHFAMT	11, 222, 224, 244
SHFLOK	13, 217, 232, 233
SHPDVS	12, 69, 70, 71, 73, 74
SIDWAY.....	8, 289
SIOINV	19, 49, 156
SIOV.....	19, 57, 156, 174, 177, 283, 285, 287
SIZEM.....	16

SIZEP0	16
SIZEP1	16
SIZEP2	16
SIZEP3	16
SKCTL	12, 18, 48, 86, 187, 199
SKRES	18, 196, 199
SKSTAT	17, 18, 33, 100, 196, 205, 206
SLFTSV	19, 68, 85, 156
SOUNDR	10, 187, 200
SPBCTL	13
SPECIL	6, 161
SRTIMR	12, 33, 273
SSFLAG	14, 27, 224, 272
SSKCTL	12, 187, 199, 206
ST1K	20, 92, 94
ST3000	21, 89, 90, 91, 98, 109, 111
ST3002	21, 107
ST3004	21, 111
ST300B	21, 103
ST301C	21, 97
ST301E	21, 97
ST3020	21, 92, 94, 98, 111
ST3021	21, 100
ST3022	21, 111
ST3024	21, 92, 95, 111
ST3028	21, 111
ST3038	21, 93, 96
ST303C	21, 104
ST304C	21, 111
ST3052	21, 101
ST3062	21, 107
ST306D	21, 102
ST3072	21, 111
ST3092	21, 111
ST309E	21, 105
ST30AB	21, 111
ST30B7	21, 111
ST30C1	21, 107
ST30C2	21, 107
ST30C7	21, 104
ST30CA	21, 104
ST30F8	21, 104
ST3100	21, 91
ST3121	21, 107
ST3122	21, 107
ST313C	21, 104
ST3150	21, 103
ST3154	21, 104

ST3181.....	21, 107
ST3182.....	21, 107
ST3186.....	21, 104
ST318C	21, 104
ST31B0.....	21, 103
ST31C2	21, 107
ST31CA	21, 104
ST31EE.....	21, 105
ST31F1.....	21, 107
ST3210.....	21, 103
ST321A.....	21, 104
ST3248.....	21, 104
ST3270.....	21, 103
ST32D0	21, 103
STACKP	14, 188, 206
STADR1.....	20, 108, 109, 291
STADR2.....	20, 108, 291
STATC	8, 50, 57, 285, 288
STATIS	6
STATUS	10, 57, 72, 154, 155, 157, 161, 163, 189, 190, 191, 192, 195, 196, 206, 222, 225, 227, 261, 266, 280, 285, 286, 288
STAUT	20, 86, 94, 99, 100, 105
STBL.....	20, 88
STCDA.....	20, 103
STCDI.....	20, 103
STCHK.....	20, 290, 291
STICK0	13, 33, 34
STICK1	13, 33
STICK2	13, 33, 41
STICK3	13, 33, 41
STIMER.....	18
STJMP.....	20, 87, 90
STKST.....	20, 87, 97
STLM	20, 92, 93
STMVAL.....	20, 93
STNOT.....	20, 103, 105
STPAG	20, 92, 93, 94
STPASS	20, 86, 88, 89
STPC	20, 93
STRIG0	13, 34
STRIG1	13, 34
STRIG2	13, 34, 41
STRIG3	13, 34, 41
STSEL.....	20, 86, 87, 88, 89
STSKP.....	20, 99
STSM.....	20, 92, 93, 94, 96
STSMP	20, 92, 93, 94, 96
STSPP	20, 88, 89

STTIME	20, 86, 89
STTMP1	20, 100
STTMP2	20, 106
STTMP3	20, 106
STTMP4	20, 106
STTMP5	20, 95
STVOC	20, 103, 105
SUBTMP	13, 242
SUCCESS	7, 60, 163, 189, 191, 192, 195, 213, 218, 227, 249, 278, 279, 280, 281, 286, 287
SUPERF	15, 229, 230, 231, 233
SWPFLG	11, 218, 247, 249, 258, 261, 262
SYSVBV	19, 156
TABMAP	13, 218, 240, 246, 251, 252, 254
TEMP	12, 191
TEMP1	14, 172, 173, 175, 176, 177, 178, 179, 182, 184, 203
TEMP2	14, 172, 173, 175, 176, 177, 178, 179, 184
TEMP3	14, 172, 175, 176, 205
TIMER1	14, 203, 205
TIMER2	14, 203
TIMFLG	14, 195, 198, 199, 205, 206
TIMOUT	7, 191, 195
TINDEX	13, 218
TMPCHR	11, 224
TMPCOL	13
TMPLBT	13
TMPROW	13, 260
TOADR	11, 253
TONE1	8, 278
TONE2	8, 277
TOPSLT	12
TRAMSZ	10, 40, 43, 47, 49
TRIG0	16, 34
TRIG1	16, 34
TRIG2	16, 24, 31, 42
TRIG3	16, 24, 32, 37, 46
TRNRCD	7, 165
TRUE	6
TSTAT	14, 189, 191
TXTCOL	13, 222
TXTMSC	13, 219
TXTOLD	13
TXTROW	13, 222, 261
USAREA	15
VBREAK	12, 28, 45
VCOUNT	18, 86, 203, 204, 205, 207, 219, 255, 261
VDELAY	17
VDSLST	12, 24, 45, 86, 87, 218, 227
VGC	6, 33, 34, 41

VIMIRQ.....	12, 25, 45
VINTER.....	12, 28, 45
VKEYBD.....	12, 29, 45
VMCTR.....	13
VOICE1.....	17
VPCTR.....	13
VPIRQ.....	12, 26, 49
VPRCED.....	12, 28, 45
VSCROL.....	18, 32
VSERIN.....	12, 26, 29, 45
VSEROC.....	12, 29, 45
VSEROR.....	12, 29, 45
VSFLAG.....	13, 32, 255
VSTB.....	17
VTIMR1.....	12, 29, 45
VTIMR2.....	12, 29, 45
VTIMR4.....	12, 29, 45
VVBLKD.....	12, 34, 45
VVBLKI.....	12, 24, 45
WARMST.....	10, 39, 46, 48, 50, 55, 172
WARMSV.....	19, 156
WIRGHI.....	8, 198
WIRGLN.....	8, 207
WIRGLP.....	8, 207
WLEADN.....	8, 284
WLEADP.....	8, 284
WMODE.....	13, 277, 278, 281
WRITE.....	8, 13, 57, 198, 277, 278, 281, 284, 288, 289
WRONLY.....	7, 164
WSIRGN.....	8, 207
WSIRGP.....	8, 207
WSYNC.....	18, 36, 86, 88, 96, 274
XFMFLG.....	11, 145, 149
XITVBV.....	19, 156
XMTDON.....	10, 192, 194
ZCHAIN.....	10, 82, 172, 178, 179, 180, 181, 182, 184, 185
ZF1.....	18, 120
ZFR0.....	18, 113, 118, 120, 126, 132, 137
ZHIUSE.....	14
ZIOCB.....	10, 76, 215
ZLOADA.....	14, 175
ZTEMP1.....	12, 119, 125, 127, 128, 137
ZTEMP3.....	12, 119, 133
ZTEMP4.....	12, 118, 119, 120, 121, 123, 133, 134, 135, 139

/

IF...ENDIF sections of source code

ACMI.....	6, 10, 14, 15, 16, 18, 24, 26, 29, 31, 42, 44
-----------	---

LNBUG	6, 10, 16, 24, 25, 39
RAMSYS.....	6, 47
VGC.....	6, 33, 34, 41

P

Portion of source code that exits with

Return to caller (RTS)

ACB	43, 55
AFP	18, 112, 113
AST	106
AUB	277, 278, 282
AVV	203, 204
BAI	50, 51, 55
BEL	228, 240, 267
BLR	250, 255
BMC.....	238, 250, 251
BMG.....	237, 246, 251, 252
BMI	25, 42, 50, 94, 113, 115, 119, 123, 125, 127, 141, 145, 146, 148, 161, 164, 166, 173, 189, 196, 198, 204, 205, 211, 214, 215, 228, 231, 234, 238, 241, 242, 250, 251, 252, 254, 266, 277, 278, 279, 281, 285
BMS	237, 250, 251
CBI.....	51, 52
CBR	203, 205
CCA	222, 224, 239, 240, 244, 254, 259
CCC.....	230, 233, 260
CCE	37, 43, 47
CCL	155, 281
CCO.....	162, 163, 164, 165, 167
CCR	222, 223, 248, 249, 265
CEP.....	162, 163, 164, 166, 168
CGB	155, 279
CHM	236, 241, 249, 267
CIA.....	222, 252
CLN	239, 254
CLT	82, 172, 178, 179, 181, 184
COC.....	217, 218
COP	155, 276
CPB	155, 280
CRR	290, 291
CSO	189, 190, 198, 206
CSP.....	155, 276
CST	155, 280
DAW	95, 96, 100, 105
DBL.....	164, 166, 168
DBP	165, 169
DBS	219, 241, 242
DCT.....	31, 32, 35
DFS.....	92, 94
DIO	19, 49, 54, 56, 57, 156

DLP	116, 117, 137
DRS.....	93, 94, 96
DSR.....	190, 192, 200, 206
DSS.....	92, 95
DVN	103, 104, 105, 106
EHC	162, 163, 164, 166, 169
END	9, 59, 60, 67, 293
EPC.....	71, 72, 74
ESC.....	13, 110, 234, 266
ESP	154, 227
EXP10	19, 144, 145
FARR.....	138, 139
FASC.....	18, 115, 116
FDH.....	76, 170, 171
FLD0P	19, 142
FLD1P	19, 142, 143
FMOVE.....	19, 141, 143, 144, 145, 146, 149
FMUL	18, 124, 125, 126, 127, 141, 144, 145, 146, 149
FNZ	116, 136
FPI.....	18, 118, 119, 145
FSTOP.....	19, 143
GDC.....	222, 225, 238, 239, 258, 265
GIN	69, 157
GND.....	119, 136
GTO.....	189, 198, 201
HIG	66, 67
IBP	164, 165, 166, 169
ICIO	156, 159
ICX.....	114, 128
IDH.....	169, 170
IDIO	56, 156
IHW.....	37, 46, 48
IIH24, 156	
IIN159, 163	
ILP.....	116, 121
IPH.....	73, 74
ISIO	156, 187
IST85, 87	
ISW	42, 49
ITO	195, 205
LOGC.....	149, 151
LOGS	147, 148
M0E.....	137, 140
M12.....	137, 140
N0E.....	126, 131, 132
OCO.....	276, 278
PAI	42, 44
PBC.....	277, 278

PCL.....	154, 287
PHE.....	156, 211
PHG.....	175, 176
PHO.....	163, 213
PHQ.....	179, 180, 181
PHR.....	43, 172, 179
PHU.....	156, 179, 181, 184
PIN.....	154, 285
PIO.....	70, 156, 186
PLYEVL.....	19, 140, 141, 145, 149
PMI.....	39, 46
POP.....	154, 286
PPM.....	286, 287, 289
PSP.....	154, 285
PTE.....	199, 206
PTL.....	213, 214
RCB.....	156, 279
REC.....	189, 191, 195, 198
RLR.....	59, 175
ROD.....	223, 228, 230, 250
RRC.....	238, 239, 258, 260, 265
S0ER.....	125, 128, 134
SAL.....	117, 135
SAS.....	86, 99, 100, 101, 105
SBA.....	57, 58
SBP.....	189, 197, 198
SBR.....	198, 205
SBS.....	234, 259
SBT.....	99, 100, 101
SCB.....	279, 283
SCL.....	236, 247, 262
SDL.....	86, 87, 92, 99, 103
SDP.....	285, 287, 288
SEN.....	192, 198, 201
SEP.....	247, 265
SFL.....	165, 167, 169
SHC.....	178, 179, 180, 184
SIL.....	119, 121
SIN.....	154, 217
SKC.....	231, 232, 240, 261, 275
SLB.....	113, 129
SLC.....	235, 237, 238, 239, 240, 246, 247, 257, 259
SLD.....	93, 94, 95, 97
SML.....	131
SMS.....	236, 255, 262
SNP.....	70, 73, 74
SPQ.....	224, 225, 238
SRC.....	238, 239, 258, 260, 265

SRR	133
SSE	220, 221, 243
SSM.....	92, 99, 100, 101, 103, 109
SSP	154, 263
SSR	199, 200
SSV	198, 202, 206
STS.....	285, 288
STW.....	189, 201, 202
SUC	86, 92, 99, 103, 109
SUE.....	125, 127, 137
SUP.....	125, 127, 137
SVN.....	104, 105
SVP	36, 156, 183
SVR	86, 87, 94, 95, 108
TDP	134, 135
TEX.....	62, 67
TNC.....	128, 129
TVN	113, 130
VCS.....	290
WCA.....	191, 202
WCB.....	280, 281, 283
WFK	282, 283
XFORM.....	147, 149
ZXLY.....	113, 120, 121

Return to caller, Interrupt (RTI)

BIR	27, 49
DVNM.....	31, 36, 45, 156
FDL	218, 274
IRIR	45, 196, 201
KIR	45, 272
OCIR	45, 194, 201
ORIR.....	45, 193, 201
PIR	49, 71
POD	87, 97
RIR	28, 29, 45, 227

Portion of source code that jumps to another portion (conditional and unconditional branching)

ABI	50, 51
ACB	43, 55
ACC	238, 245, 246, 265
ADB	43, 50
AIR.....	26, 29
BLG.....	237, 240, 251
BLG1.....	251, 254, 257
BLG2.....	240, 251
BMP	240, 250, 254
BSP	237, 266
COA.....	116, 134
CAL.....	59, 61

CBC.....	228, 258
CBT.....	241, 267
CCR.....	222, 223, 248, 249, 265
CDN	234, 266
CEL.....	223, 230
CEP	162, 163, 164, 166, 168
CGB.....	155, 279
CIN	155, 276
CIO.....	4, 10, 19, 43, 49, 53, 76, 156, 159, 160, 161, 162, 163, 164, 165, 167, 179, 214, 249
CIR	26, 27, 28
CLF.....	234, 237, 258, 266
CLM.....	235, 240, 267
COC	217, 218
COP.....	155, 276
CRE	228, 229, 248
CRM	234, 235, 267
CRT.....	235, 237, 266
CSC.....	222, 223, 236, 266
CTB.....	238, 266
CUP.....	234, 237, 241, 266
DBE	51, 52, 53
DCH	239, 267
DDD.....	220, 241
DLN	240, 266
DLN1.....	240, 259
DLW	94, 95, 99, 105
DMW	92, 93, 95, 103
DQQ	239, 259
DSD	220, 241, 242
DWQ.....	237, 259
EBL	52, 53
ECL.....	154, 227
EGB	154, 228
ELL	239, 254
EMS.....	39, 42, 85
EOP	154, 217, 222, 227, 248
EPB.....	154, 229
EPC	71, 72, 74
ESR.....	195, 199
ESS.....	156, 192, 198, 199
EST	85
EXP	19, 144, 148
F1R.....	138
F2R.....	138, 139
FADD	18, 122, 123, 141, 147, 149
FDIV.....	18, 126, 127, 146, 147, 149, 150
FLD0R.....	19, 141, 142, 145, 146, 147
FLD1R.....	19, 141, 142, 144, 147, 149, 150

FMUL	18, 124, 125, 126, 127, 141, 144, 145, 146, 149
FNZ	116, 136
FPB	286, 287
FRA10	125, 138
FRA1E	128, 138
FRA20	125, 138
FRA2E	127, 139
FST0R	19, 141, 143, 145, 147, 149
FSUB	18, 121, 122, 145, 147
GBY	59, 60, 61
GCL	71, 157
GGB	72, 157
GNC	113, 114, 115, 128, 130
GNL	172, 176, 177
GNS	51, 52, 54
GOP	71, 157
GPB	72, 157
GSP	72, 157
GST	72, 157
HIG	66, 67
IBS	50, 53
ICH	238, 267
ICR	216, 277
ICS	43, 44
IFP	18, 118, 145, 149
IGN	230, 231, 232
IHW	37, 46, 48
IIO76, 162	
IIR 26, 45	
ILN	239, 266
ILN1	239, 246
IND	76, 160
IRA	229, 230
IRQ 4, 12, 13, 16, 17, 18, 19, 25, 26, 27, 28, 29, 31, 42, 45, 49, 71, 193, 194, 196, 199, 200, 201, 218, 272, 293	
IVNM	31, 45, 156
KBK	100, 101
KGB	154, 228, 230, 231, 283
KRK	100, 102
KSB	100, 101
KTK	100, 101
LHO	160, 161
LOG	19, 147, 149, 150
LOG10	19, 144, 147, 149, 150
LOGQ	148, 149, 151
LOGS	147, 148
LPH	76, 172, 173, 175
MLN	239, 253
NMI	12, 16, 18, 19, 24, 31, 34, 36, 45, 86, 156, 293

NORM.....	115, 118, 123, 124, 131
OCI.....	156, 276, 277
PBK.....	192, 195, 205, 206
PCC.....	160, 161
PCH.....	228, 229, 230
PCI.....	188, 198
PCL.....	154, 287
PCS.....	37, 38, 156
PHC.....	76, 172, 173, 179, 180
PHI.....	156, 179, 180
PHL.....	76, 161, 179, 214, 215
PHP.....	166, 172, 173, 174, 179, 180, 213, 240, 254
PHW.....	172, 179
PHX.....	181, 182
PLO.....	223, 224, 237, 265
PMD.....	86, 88
PPB.....	154, 286
PPO.....	162, 163
PPP.....	286, 287
PRS.....	37, 38, 39
PST.....	154, 285, 286
PTO.....	31, 35
PTT.....	32, 35
PWS.....	37, 38, 156
RCB.....	156, 279
REC.....	189, 191, 195, 198
RES.....	38, 43, 293
RET.....	237, 246, 247
RWS.....	223, 228, 246, 266
S0L.....	113, 115, 131, 136
S0R.....	123, 133
S1R.....	123, 133
SBR.....	198, 205
SCC.....	235
SCR.....	247, 255
SDA.....	245
SDF.....	220, 241
SDI.....	220, 221, 241, 243
SEA.....	223, 245, 258
SEC 32, 62, 63, 115, 116, 117, 119, 120, 123, 124, 125, 126, 127, 128, 129, 130, 132, 134, 137, 145, 146, 148, 151, 169, 170, 171, 173, 176, 178, 179, 181, 185, 203, 204, 211, 212, 221, 222, 223, 225, 228, 230, 239, 240, 242, 247, 252, 253, 254, 256, 259, 263, 266, 278, 290	
SEL.....	85, 86, 97
SEN.....	192, 198, 201
SES.....	68, 226
SGB.....	154, 222, 228
SHT.....	162, 163, 170
SID.....	188, 189, 201

SIO4, 9, 10, 14, 19, 49, 57, 70, 156, 174, 176, 177, 187, 188, 189, 190, 192, 195, 198, 199, 202, 206, 279, 283, 285, 287, 288	
SML.....	131
SNL.....	117, 134, 135
SOP	154, 217
SPB.....	154, 223
SSC.....	160, 162, 163, 164, 166, 167
SSD.....	240, 255, 256
SSP	154, 263
SST	154, 222, 225, 227, 233, 249, 261, 266
STB	237, 267
STH.....	85, 156
STK	87, 99, 105
STM.....	85, 87, 92, 99
STV	87, 94, 103
SWA.....	228, 229, 230, 249, 261
SZA	222, 239, 245, 258
TAB	110, 237, 266
TNC	128, 129
TVN.....	113, 130
VFR.....	40, 92, 290
VSR.....	40, 92, 290
WFR.....	30, 31, 32
XCL.....	160, 161, 163
XGT.....	161, 164
XIR	28, 45
XOP.....	161, 162
XPT	161, 166
XSS	161, 163
ZF1.....	18, 120
ZFR0	18, 113, 118, 120, 126, 132, 137

T

Table of Data (.BYTE or .WORD sections)

6502 Machine Vectors	5, 293
ATCOEF - Arctangent Coefficients	150
BMSG - Boot Error Message	45
CASETV - Cassette Handler Vector Table	155
Cassette Handler NTSC/PAL Constant Tables	284
DISL1 - Display List for Main Screen	89
DISL2 - Display List for Memory Test	90
DISL3 - Display List for Exit Text	90
DISL4 - Display List for Keyboard Test	91
DISL5 - Display List for Audio-visual Test	91
Domestic Character Set.....	5, 152
EDITRV - Editor Handler Vector Table	154
FHALF - 0.5	150
First 8K ROM Identification and Checksum	4, 23

FONE - 1.0	146
GPDVV - Generic Parallel Device Handler Vector Table	157
International Character Set	5, 83
Jump Vectors	5, 156
KEYBDV - Keyboard Handler Vector Table	154
LGCOEF - Logarithm Coefficients	150
LOG10E - Base 10 Logarithm of e	146
P10COF - Used in EXP10 calculation	146
Printer Handler Address Data	285
PRINTV - Printer Handler Vector Table	154
Screen Editor Device Specification	45
SCRENV - Screen Handler Vector Table	154
Second 8K ROM Identification and Checksum	5, 292
Serial Input/Output NTSC/PAL Constant Tables	207
SMEM1 -	89
SMEM2 -	90
SMEM3 -	90
SMEM4 -	91
SMEM5 -	91
SQR10 - Square Root of 10	150
TAGM - Table of ANTIC Graphics Modes	208
TAIC - Table of ATASCII to Internal Conversion Constants	267
TARS - Table of Address Ranges to Set	111
TARV - Table of Address Ranges to Verify	291
TAVD - Table of Audio-visual Test Display Data	107
TBTM - Table of Bit Masks	210
TCCR - Table of Control Character Routines	266
TCDA - Table of Cleft Display Addresses	107
TCKD - Table of Character Key Definitions	268
TCVO - Table of Command Vector Offsets	171
TDLE - Table of Display List Entry Counts	208
TDLV - Table of Display List Vulnerability	209
TDSC - Table of Default Screen Colors	266
TDSM - Table of Display Masks	210
TFKD - Table of Function Key Definitions	271
THAV - Table of Handler Vectors	45
TIAC - Table of Internal to ATASCII Conversion Constants	267
TIHV - Table of Interrupt Handler Vectors	45
TIRQ - Table of IRQ Types	29
TLSC - Table of Left Shift Counts	209
TMCC - Table of Mode Column Counts	209
TMNT - Table of Memory Not to Test	98
TMRC - Table of Mode Row Counts	210
TMSK - Mask used in CCA, Convert Cursor Row/Column to Address	266
TNDD - Table of Note Duration Delays	106
TOIH - Table of Offsets to Interrupt Handlers	29
TPDS - Table of Parallel Device Selects	75
TPFV - Table of POKEY Frequency Values	207

TRPR - Table of Record Processing Routines	67
TRSC - Table of Right Shift Counts.....	210
TSFR - Table of Super Function (Shifted Function Key) Routines	267
TSIH - Table of SIO Interrupt Handlers.....	201
TSKP - Table of Simulated Keypresses.....	102
TSMa - Table of Screen Memory Allocation	208
TSMC - Table of Screen Memory Character Codes	110
TSTD - Table of Self-test Text Destination Offsets.....	111
TSTL - Table of Self-test Text Lengths	111
TSTO - Table of Self-test Text Offsets.....	77
TTXT - Table of Text Sequences	77
TXT0 -	77
TXT1 -	77
TXT2 -	77
TXT3 -	81
TXT4 -	81
TXT5 -	81
TXT6 -	81
TXT7 - Keyboard	78
TXT8 - Control Key.....	81
TXT9 -	81