SOFTWARE STATUS

	KBD	
	ORIG 1A	NEW
PDS V3 →	V3A ↓ V3D ↓	V3N
SYS 2	SYS 2A (HAS NO ASSEM BUT HAS LOAD & DUMP COMMANDS IN EXEC)	S2N BUG S2NF DEBUGGED

V3D?

PDS-V3A

ORIGINAL FIRMWARE
FOR KBD 1

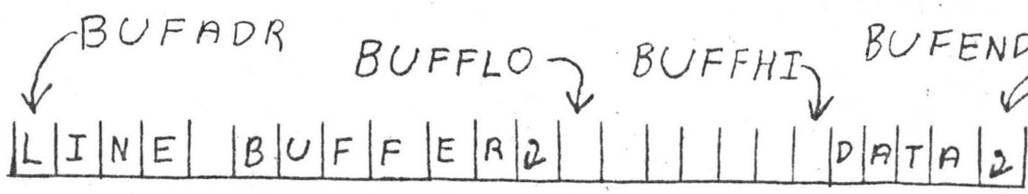
NO CALLS TO CASSETTE
SUBROUTINES

1A


```
                        NAM     PDS-V3A
                        OPT     O,NOG
                *
                *       PROGRAMMED BY ERIC JAMESON
                *
                *       COPYRIGHT 1976 SPHERE CORPORATION
                *       791 SO.  500 W.;   BOUNTIFUL, UTAH   84010
                *
                *       SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
                *       DISTRIBUTION AND USE OF THE PDS SOFTWARE.
                *       NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
                *       WRITTEN PERMISSION OF SPHERE CORP.
                *
                *
                * THE PROGRAM DEVELOPMENT SYSTEM (PDS V3A) IS A SET OF
                * PROGRAMS RESIDING ON ERASABLE PROGRAMMABLE READ ONLY
                * MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS
                * SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE
                * DEVELOPMENT OF COMPUTER PROGRAMS.
                *      TOWARD THIS END, THE 4 PDS EPROMS CONTAIN A CURSOR
                * BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING
                * AID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16
                * BIT MULTIPLY AND DIVIDE, ASCII-TO-BINARY, AND
                * BINARY-TO-ASCII ROUTINES.
                *
                *
                *
                *               MEMORY MAP
                *
        0000    TMP     EQU     $00
        0002    TMP1    EQU     $02
        0004    ARB     EQU     $04     16 BIT ACC.  PSEUDO REG B.
        0004    AR3     EQU     $04     _HI BYTE OF ARB.
        0003    AR2     EQU     $03     _LO BYTE OF ARB.
        0006    ARA     EQU     $06     16 BIY ARITH PSEUDO REG A.
        0006    AR1     EQU     $06     _HI BYTE OF ARA.
        0007    AR0     EQU     $07     _LO BYTE OF ARA.
        0008    DIGIT   EQU     $08     BYTE USED BY ASCBIN FOR TMP.
        000A    OUTEND  EQU     $0A     END OF OUTPUT BUFFER TEXT.
        000C    BUFADR  EQU     $0C     START OF I/O BUFFER (PTR)
        000E    BUFEND  EQU     $0E     PTR.  TO END OF I/O BUFFR.
        0011    OUTBUF  EQU     $11     START OF OUTPUT BUFFER.
        0014    SRCADR  EQU     $14     SOURCE FOR TEXT MOVES.
        0016    DSTADR  EQU     $16     DEST.  ADDR.  FOR TEXT MOVE.
        001A    ENDMEM  EQU     $1A     LAST ADDRES OF REAL MEMORY.
        001C    CSRPTR  EQU     $1C     PTR TO CURSER ON SCREEN.
        001E    BUFPTR  EQU     $1E     TEMP PTR USED BY OUTSTR.
        0020    BUFFLO  EQU     $20     PTR TO END OF LOW EDIT TXT.
        0022    BUFFHI  EQU     $22     PTR TO START OF HI TEXT.
        0024    SCNPTR  EQU     $24     PTR.  TO BUFFRD TXT START.
        0026    SRCASM  EQU     $26     PTR TO ASSMBLR SOURCE CODE.
        002A    ONDVAL  EQU     $2A     HAS ASSMBLR OPERND VALUE.
        002C    SYMVAL  EQU     $2C     VALUE PUT IN ASSM.  SYMTBL.
        002E    BRKSAV  EQU     $2E     TEMP SAVE FOR BRKPT DATA.
        0030    BRKADR  EQU     $30     ADDDRESS OF BREAKPOINT.
        0032    EDIT    EQU     $32     0 IF EDITOR IS NOT RUNNING.
        0035    IOBUFF  EQU     $35     I-O BUFFER FOR DEBUGGER.
        0040    PCVAL   EQU     $40     PROGRAM COUNTER FOR ASSM.
```

```
                    *                     INITIALIZATION
                    *
                    *
                    *
                    *    THE INITIALIZATION ROUTINES SET UP THE INITIAL VALUES
                    * UPON SYSTEM RESET.
                    *
                    *
FC00                       ORG     $FC00
FC00 8E 01FF RESTRT LDS     #$1FF     SETS STACK POINTER
FC03 30                    TSX                 MOVES STK PTR TO INDEX REG
FC04 DF 26                 STX     SRCASM    SETS ASSEMBLR OUTPUT PTR
FC06 DF 0C                 STX     BUFADR    INIT INPUT BUFFR ADDR.
FC08 86 07                 LDA A   #07       INHTLZ. KEYBOARD PIA.
FC0A B7 F001               STA A   $F001
FC0D CE 0FFF               LDX     #$FFF     LAST LOCATN OF MEMORY.
FC10 DF 0E                 STX     BUFEND    INIT END-OF-EDIT BUFFR.
FC12 DF 1A                 STX     ENDMEM    INIT END-OF-MEM ADDR.   4095
                    *
                    *
                    *
                    *
                    *
                    *                     COMMAND LANGUAGE
                    *
                    *     THIS EXECUTIVE ACCEPTS COMMANDS FROM THE KEYBOARD TO
                    * DETERMINE WHAT UTILITY IS TO BE RUN.   INVALID COMMANDS
                    * WILL SPACE THE CURSOR DOWN ONE LINE.  DO NOT SPACE OFF THE
                    * BOTTOM OF THE SCREEN.
                    *
                    *
FC14 8D 1C   EXEC   BSR     HOME      CSRPTR IS HOMED.
FC16 8D 20          BSR     CLEAR     CLEARS SCREEN
FC18 BD FD16 EXEC1  JSR     CRLF      NEW LINE.
FC1B BD FE5C        JSR     INPCHR    GETS & DISPLAYS CHR.
FC1E 81 01          CMP A   #$01      TESTS IF ASSEMBLY COMMD.
FC20 26 03          BNE     EXEC2     SKIPS IF OTHER COMMAND.
FC22 BD FDA1        JSR     ASMBLR    JUMPS TO ASSEMBL PRROGRM.
FC25 81 05   EXEC2  CMP A   #$05      TESTS IF CONTRL 'E'.
FC27 26 02          BNE     EXEC3     SKIPS IF NOT EDIT CMD.
FC29 8D 35          BSR     EDITOR    JUMPS TO EDIT TEXT.
FC2B 81 04   EXEC3  CMP A   #$04      TESTS FFOR CONTRL 'D'.
FC2D 26 E9          BNE     EXEC1     LOOPS BACK IF NOT DEBUG.
FC2F 7E FE4F        JMP     DEBUG     JUMPS TO DEBUGGER.
```

```
*                    THE EDITOR
*
*
*      THE EDITOR ALLOWS INPUT FROM THE KEYBOARD INTO A
* BUFFER MEMORY.  INPUT IS DISPLAYED ON THE SCREEN.  WHEN
* IT IS TYPED IN, THE SCREEN TEXT CAN THEN BE EDITED BY USE
* OF THE CURSOR.  WHEN THE SCREEN IS FULL OR EDITING IS
* FINISHED, THE DATA IS SCROLLED OFF THE SCREEN INTO THE
* EDIT BUFFER.  WHEN TEXT IS SCROLLED OFF THE TOP OF THE
* SCREEN, IT IS STORED FROM THE BUFFER ADDRESS POINTER
* (BUFADR) TO THE LOW BUFFER POINTER (BUFFLO).  BUFFLO
* POINTS TO THE END OF TEXT + ONE (I. E.  IT POINTS TO THE
* FIRST UNUSED BYTE).  WHEN IT IS SCROLLED OFF THE BOTTOM
* OF THE SCREEN, IT IS STORED IN THE TOP OF THE EDIT BUFFER.
* THE TEXT GOES FRROM THE HIGH BUFFER POINTER (BUFFHI) TO
* THE END OF BUFFER POINTER (BUFEND).  BUFFHI POINTS TO
* THE END OF TEXT - ONE (I. E.  IT POINTS TO THE LAST UNUSED
* BYTE IN THE BUFFER).  WHEN THE TEXT IS SCROLLED UP
* OFF THE TOP OF THE SCREEN, TEXT IS TAKEN FROM THE HIGH
* AREA OF THE EDIT BUFFER AND DISPLAYED ON THE LAST LINE OF
* THE SCREEN.  WHEN TEXT IS SCROLLED DOWN OFF THE BOTTOM,
* TEXT, IF ANY EXISTS, IS MOVED FROM THE LOW EDIT BUFFER
* AREA TO THE TOP LINE OF THE SCREEN.
*
*
*
```




```
*
*
*
*                    POINTERS USED
*
* CSRPTR    POSITION OF CHARACTERS INSERTED ON THE SCREEN.
* SCNPTR    POSITION OF START OF EDITED TEXT ON SCREEN.
* BUFADR    START OF TEXT BUFFER IN MAIN MEMORY.
* BUFEND    END OF TEXT BUFFER IN MAIN MEMORY.
* BUFFLO    END OF TEXT SCROLLED OFF TOP OF SCREEN.
* BUFFHI    START OF TEXT SCROLED OFF BOTTOM OF SCREEN.
* BUFLEN    NOT CURRENTLY USED.
*
*
*
*
*
*                  EDITOR COMMANDS
*
* "UP ARROW"  MOVES CURSOR UP ONE LINE; CSRPTR GETS
*  CSRPTR-32; CALL NDRFLO.
*
* "DOWN ARROW"  MOVES CURSOR DOWN ONE LINE; CSRPTR GETS
*  CSRPTR+32; CALL OVRFLO.
*
* "RIGHT ARROW"  MOVES CURSOR ONE POSITION RIGHT; CSRPTR
*  GETS CSRPTR+1; CALL OVRFLO.
```

```
* "LEFT ARROW"  MOVES CURSOR ONE POSITION LEFT; CSRPTR
*  GETS CSRPTR-1; CALL NDRFLO.
*
* "CONTROL & LEFT ARROW (ON KEYBOARD)"  LEFT JUSTIFY CURSOR;
*  CSRPTR GETS CSRPTR TRUNCATED; CALL NDRFLO FOR SCNLOC CHK.
*
* "PUTCHR"  OUTPUTS CHARACTER; CSRPTR GETS
*  CSRPTR+1; GOES TO OVRFLO.
*
* "ENDCHR"  TERMINATION CHAR; CLEAR EDIT FLAG;
*  EXIT THE EDITOR.
*
* "HOME"  HOMES CURSOR POINTER; CSRPTR GETS E000; NDRFLO.
*
* "CLEAR"  CSRPTR TO END OF THE SCREEN GETS SPACES.
*
* "CTRL I"  INSERT A LINE AT THE LAST LINE ON THE SCREEN;
*  CALL OVR1 (SCROLLS UP ONE LINE); CSRPTR GETS E1E0.
*
* "CTRL D"  DELETE LAST LINE; SCROLL DOWN (UNDR2);
*  CSRPTR GETS E1E0.
*
*
*
*
* OVERFLOW CHECKS IF SCROLL UP IS NEEDED; IF IT IS, IT
*  SCROLLS UP AND MOVES DATA TO & FROM THE BUFFERS.
*
* OVRFLO:  IF CSRPTR < E200 THEN RETURN; IF EDIT IS ON THEN
* OVR1:    BUFFLO+ GETS SCNPTR TO 'C.R.';
* DSTADR GETS CSRPTR GETS E1E0 (LAST LINE ON SCREEN);
* IF EDIT IS ON AND BUFFHI < BUFEND THEN MOVE THE TEXT
* FROM BUFFHI TO 'C.R.' TO THE LAST LINE.
*
*
*
*
* UNDERFLOW CHECKS IF SCROLL DOWN IS NEEDED AND MOVES
*  DATA TO AND FROM THE BUFFERS.  CURSER HAD BEEN
*  MOVED OFF THE TOP OF THE SCREEN AND IS NOW PUT AT THE
*  HOME POSITION ON THE SCREEN.
*
* NDRFLO:  IF CSRPTR > DFFF THEN RETURN (GO TO OVRFLO);
*  IF EDIT FLAG IS ON THEN MOVE LAST LINE TO BUFFHI
*   ON DOWN; SCRLDN; CSRPTR GETS E000; MOVE LINE FROM
*   BUFFLO TO FIRST LINE ON THE CRT.
*
*
*
*
*  NOTE:  DON'T SCROLL OFF SCREEN IN EXEC UNTIL AFTER
*    THE EDITOR HAS BEEN RUN.
*
*  NOTE:  EVERY LINE MUST HAVE A C.R. ON IT.
```

X

```
FC32 CE E000 HOME    LDX     #$E000    LOADS HOME POSITION.
FC35 DF 1C           STX     CSRPTR    STORES HOME IN CURSOR PTR.
FC37 39              RTS               RETURNS TO CALLER.
                     *
                     *
FC38 C6 60   CLEAR   LDA B   #$60      LOADS BLANK (C. R. ).
FC3A CE E200         LDX     #$E200    LOADS END-OF-SCREEN PTR.
FC3D 09      CLEAR1  DEX               DECREMENTS BLANKING PTR.
FC3E E7 00           STA B   0,X       BLANKS LOCATION.
FC40 9C 1C           CPX     CSRPTR    TESTS IF DONE.                    LOC 43 = F8
FC42 26 F9 8         BNE     CLEAR1    BRANCHES BACK IF NOT DONE.
FC44 39              RTS               RETURNS.
                     *
                     *
                     *
                     *       GETCHR INPUTS A CHARACTER INTO ACC A WITHOUT
                     *       MOVING THE CURSOR, AND BLINKS THE CURSOR.
                     *
FC45 DE 1C   GETCHR  LDX     CSRPTR    LOADS CRT CURSOR POSITION.
FC47 63 00           COM     0,X       COMPLIMENT (FLASH POSITION).
FC49 CE 26F0         LDX     #9968     LOADS BLINK COUNT VALUE.
FC4C 09      GET1    DEX  DECREMENT X REGISTER  COUNT GETS COUNT-1.
FC4D 27 F6   TEST FOR BEQ  BRANCH IF COND CODE BIT Z=1  GETCHR  RESETS CTR WHEN TIMED OUT.
FC4F 7D F001         TST     $F001  KBD PIA CONTROL REG  TESTS IF A CHAR TYPED IN.
FC52 2A F8           BPL  BRANCH IF PLUSS  GET1  BRANCH IF CHAR NOT ENTERED.
FC54 DE 1C           LDX     CSRPTR    LOADS CURSOR POSITION.
FC56 A6 00           LDA A   0,X       TESTS IF BLINKED (SOLID).
FC58 2A 02           BPL     GET2      SKIPS IF NOT BLINKED.
FC5A 63 00           COM     0,X       CLEARS THE CHARACTER.
FC5C B6 F000 GET2    LDA A   $F000     LOADS A WITH CHAR.
FC5F 39              RTS               RETURNS TO CALLER.
                             KBD PIA DATA REG
                     *
                     *
                     *
                     *
                     *        EDITOR IS THE MAIN ENTRY POINT FOR EDITING.
                     *
FC67
FC60 DE 0C   EDITOR  LDX     BUFADR    BUFFLO GETS THE
FC62 DF 20           STX     BUFFLO    VALUE OF BUFADR.
FC64 DE 0E           LDX     BUFEND    BUFFHI GETS THE
FC66 DF 22           STX     BUFFHI    VALUE OF BUFEND
FC68 8D C8   REEDIT  BSR     HOME      ENTRY POINT FOR
FC6A 8D CC           BSR     CLEAR     RE-EDITING TEXT.
FC6C 97 32   EDITRD  STA A   EDIT      TURNS ON EDIT MODE . 1 <> 0
FC6E DE 1C   EDITIN  LDX     CSRPTR    SETS SCNPTR TO CSRPTR.
FC70 DF 24           STX     SCNPTR
                     *
                     *                 INDEX REGISTER        ACCUMULATOR A
FC72 8D D1   EDREAD  BSR     GETCHR    X GETS CSRPTR & A GETS CHR.
FC74 81 1B   ENDCHR  CMP A   #$1B      TESTS FOR AN "ESC" CHAR.
FC76 26 04           BNE     ED1       SKIPS IF NOT EDIT END.
FC78 7F 0032         CLR     EDIT      TURNS OFF EDIT FLAG.
FC7B 39              RTS               EXITS THE EDITOR.
FC7C 8D 02   ED1     BSR     INSERT    EDITS CHARACTER.
FC7E 20 F2           BRA     EDREAD    GOES FOR NEXT CHARACTER.
```

```
                  *       FOLLOWING IS THE MAIN EDITOR EXECUTION LOOP.
                  *
                  *
                  *
                  *
                  *
FC80 81 09        INSERT CMP A  #$09      TESTS FOR A CONTROL 'I'.
FC82 2D 16               BLT    DELETE    SKIPS TO DELETE COMMD.
FC84 2E 02               BGT    CR        SKIPS FOR NEXT TEST.
FC86 20 70               BRA    OVR1      SCROLLS UP ONE LINE.
                  *
                  *
FC88 81 0D        CR     CMP A  #$0D      TESTS FOR CARRIAGE RETURN.
FC8A 2D A6               BLT    HOME      SKIPS IF HOME CSR COMMD.
FC8C 2E 02               BGT    RTCSR     GOES TO  NEXT COMND TEST.
FC8E 86 60               LDA A  #$60      LOADS INTERNAL C.R.  VALUE.
                  *
                  *
FC90 81 12        RTCSR  CMP A  #$12      TESTS FOR RIGHT ARROW.
FC92 2D 2A               BLT    SUB32     SKIPS IF AN "UP ARROW".
FC94 2E 08               BGT    LFTCSR    SKIPS IF A "LEFT ARROW".
FC96 DE 1C        RTARRO LDX    CSRPTR    LOADS CURSOR POINTER.
FC98 20 1F               BRA    PUTCH1    STORES & INCREMENTS CSR.
                  *
                  *
FC9A 8D 3F        DELETE BSR    UNDR2     SCROLLS DOWN 1 LINE.
FC9C 20 6A               BRA    OVR2      CSRPTR GETS E1E0 (LAST LIN).
                  *
                  *
FC9E 81 14        LFTCSR CMP A  #$14      TESTS IF "LEFT ARROW".
FCA0 2D 26               BLT    ADD32     SKIPS IF "DOWN ARROW".
FCA2 2E 03               BGT    CLER      SKIPS FOR NEXT TEST.
FCA4 09                  DEX              SUB. 1 FROM CSRPTR.
FCA5 20 13               BRA    PUTCH2    STORES CURSER POINTER.
                  *
FCA7 81 1F        CLER   CMP A  #$1F      TESTS FOR CTRL BACK ARROW.
FCA9 2D 8D               BLT    CLEAR     GOES TO CLEAR SCREEN.
FCAB 27 62               BEQ    LFTJST    MOVES CSR TO LEFT OF SCREEN.
                  *
                  * ALL OTHER CHARACTERS FALL THRU TO PUTCHR.
                  *
```

```
                     *      PUTCHR DISPLAYS A CHARACTER ON THE CRT DISPLAY AND
                     *        INCREMENTS THE CURSER POINTER AS WELL AS CHECKING
                     *        AND HANDLING CARRIAGE RETURNS.
                     *
                     *
      FCBC           *
FCAD DE 1C      PUTCHR LDX     CSRPTR     LOADS OLD CSRPTR.
FCAF 81 0D             CMP A   #$0D       TESTS FOR EXTERNAL C. R.
FCB1 27 04             BEQ     CRLF1      SKIPS TO DO A C. R.  L. F.
FCB3 A7 00             STA A   0,X        DISPLAYS CHAR ON SCREEN.
FCB5 81 60             CMP A   #$60       TESTS FOR INTERNAL C. R.
FCB7 27 5D      CRLF1  BEQ     CRLF       SKIPS FOR CR.  LF.
FCB9 08         PUTCH1 INX                INCREMENTS CSRPTR.
FCBA DF 1C      PUTCH2 STX     CSRPTR     SAVES NEW CSRPTR.
FCBC 20 12             BRA     NDRFLO     TESTS FOR OVRFLO & UNDRFLO.


FCBE DE 1C      SUB32  LDX     CSRPTR     LOADS CURRENT CRSR POSITION.
FCC0 C6 20             LDA B   #32        LOADS LOOP COUNT.
FCC2 09         SUB32A DEX                DECRIMENTS CSRPTR.
FCC3 5A                DEC B   -          DECRIMENTS LOOP COUNTR.
FCC4 26 FC             BNE     SUB32A     SKIPS BACK IF NOT DONE.
FCC6 20 06             BRA     ADD2       SKIPS TO CHECK UNDRFLO.

FCC8 C6 20      ADD32  LDA B   #32        LOADS LOOP COUNTER.
FCCA 08         ADD32A INX                INCRE.  CSRPTR IN INDEX.
FCCB 5A                DEC B   -          DECRIMENTS LOOP COUNTER.
FCCC 26 FC             BNE     ADD32A     SKIPS BACK IF NOT DONE.
FCCE DF 1C      ADD2   STX     CSRPTR     SAVES CSRPTR.


                     * NDRFLO (UNDERFLOW) CHECKS FOR THE CURSOR GOING OFF THE
                     *   TOP OF THE SCREEN.  THE INDEX REG.  CONTAINS THE CURSOR
                     *   POINTER WHEN THE ROUTINE IS ENTERED.
                     *
      FCDD           *
FCD0 8C E000    NDRFLO CPX     #$E000     TESTS IF CSRPTR >= DFFF.
FCD3 2C 0B             BGE     OVRFLO     SKIPS IF CSRPTR GREATER.
FCD5 D6 32             LDA B   EDIT       TESTS IF EDITOR IS ON.
FCD7 27 02             BEQ     UNDR2      SKIP TO EXIT IF EDITOR OFF.
FCD9 8D 6D             BSR     MOVE2      MOVES LAST LINE TO BUFFHI.
FCDB BD FD73    UNDR2  JSR     SCRLDN     MOVES ALL LINES DOWN ONE.
FCDE 8D 3A             BSR     MOVE3      MOVES BUFFLO TO TOP OF CRT.
```

```
                       *      OVERFLOW CHECKS FOR SCROLLING UP (CURSOR IS OFF
                       * THE BOTTON OF THE SCREEN); INDEX CONTAINS THE CURSOR
                       * POINTER UPON ENTRY.
                       *
  FCE6                 *
FCE0 8C E200 OVRFLO CPX    #$E200   TESTS AND EXITS IF
FCE3 2B 12          BMI    OVREXT   CURSOR ON SCREEN.
FCE5 8D 11          BSR    OVR1     DOES OVR1 CHECKING.
FCE7 D6 32          LDA B  EDIT     TESTS IF EDIT IS ON.
FCE9 27 0C          BEQ    OVREXT   EXITS IF IT IS OFF.
FCEB DE 22          LDX    BUFFHI   LOADS HI TEXT POINTR.
FCED 9C 0E          CPX    BUFEND   TESTS IF PTRS NOT EQU.
FCEF 27 06          BEQ    OVREXT   EXITS IF NO TEXT.
FCF1 8D 44          BSR    MOVE1A   MOVES CHRS TO LAST LINE.
FCF3 DE 14          LDX    SRCADR   RESETS NEW BUFFHI
FCF5 DF 22          STX    BUFFHI   LOCATION.
FCF7 39      OVREXT RTS             EXITS BACK TO EDIT.
  FCFB                 *
                       *
                       *
                       *      OVR1 DOES ACTUAL SCROLLING UP.
                       *
FCF8 D6 32   OVR1   LDA B  EDIT     TESTS IF EDIT IS ON.
FCFA 27 0A          BEQ    OVR1A    SKIPS IF EDIT OFF.
FCFC DE 20          LDX    BUFFLO   LOADS TEXT PTR LOW.
FCFE DF 16          STX    DSTADR   DESTINATION OF TEXT MOVE.
FD00 DE 24          LDX    SCNPTR   SOURCE FOR MOVE.
FD02 8D 34          BSR    MOVE1    MOVES LIN1 TO BUFFFLO.      2nd CHIP E12
FD04 DF 20          STX    BUFFLO   SAVES NEW BUFFLO PTR.
FD06 8D 58   OVR1A  BSR    SCRLUP   SCROLLS SCREEN UP 1.
FD08 CE E1E0 OVR2   LDX    #$E1E0   CURSOR GETS PUT ON
FD0B DF 1C          STX    CSRPTR   LAST LINE.
FD0D DF 16          STX    DSTADR   INIT DEST FOR NEXT MOVE.


                       *      FOLLOWING MOVE THE CURSOR.
                       *
FD0F D6 1D   LFTJST LDA B  CSRPTR+1 LOADS LO BYTE OF PTR.
FD11 C4 E0          AND B  #$E0     TRUNCATES TO LEFT OF LINE.
FD13 D7 1D          STA B  CSRPTR+1 SAVES L. J. ED PTR.
FD15 39             RTS             RETURNS TO EDITOR.
                       *
                       *
FD16 8D B0   CRLF   BSR    ADD32    LINE FEED.
FD18 20 F5          BRA    LFTJST   CARRIAGE RETURN.
```

```
                *     MOVE INSTRUCTIONS MOVE FROM ONE BUFFER AREA TO
                * ANOTHER BUFFER AREA.
                *
                *
                *
                *     MOVE3 CALCULATES THE SOURCE ADDRESS OF THE DATA IN
                * BUFFLO (IF IT EXISTS) FOR MOVING TO THE FIRST LINE ON
                * THE CRT.  MOVE 1 IS THEN ENTERED TO DO THE MOVING.
                *
                *
FD1A DE 24      MOVE3  LDX     SCNPTR    CSRPTR GETS E000 (HOME).
FD1C DF 16             STX     DSTADR    SETS MOVE ADDRESS.
FD1E DE 20             LDX     BUFFLO    LOADS LO BUFFR ADDR.
FD20 9C 0C             CPX     BUFADR    TESTS IF DATA STRING.
FD22 27 23             BEQ     MOVEXT    EXITS IF EMPTY.
FD24 09                DEX               MOVES BACK FROM BLANK.
FD25 9C 0C      MV31   CPX     BUFADR    TESTS IF SRCADR = BUFFADR.
FD27 27 08             BEQ     MV32      MOVES IF START OF LINE.
FD29 09                DEX               NEXT LOWER CHAR.
FD2A E6 00             LDA B   0,X       GETS SOURCE CHAR FOR TEST.
FD2C C1 60             CMP B   #$60      TESTS FOR "C.R.".
FD2E 26 F5             BNE     MV31      SKIPS BACK UNTIL "C.R.".
FD30 08                INX               POINTS BACK TO FIRST CHAR.
FD31 DF 20      MV32   STX     BUFFLO    SAVES LO ADDRESS.
FD33 20 03             BRA     MOVE1     MOVES DATA.


                *          MOVE1
                *
                *     MOVE1 MOVES A SET OF CHARACTERS FROM EITHER THE TOP
                * LINE OF THE SCREEN TO BUFFLO OR FROM BUFFHI TO THE
                * BOTTOM LINE OF THE SCREEN.  THE SOURCE ADDRESS IS PASSED
                * IN THE INDEX REG., THE DESTINATION ADDRESS IN DSTADR,
                * AND THE MOVE IS TERMINATED BY A "C.R." IN THE LINE OF
                * TEXT BEING MOVED.
                *
                *
                *
 FD33           *
FD35 DE 14      MOVE   LDX     SRCADR    LOADS SOURCE ADDRESS INTO X.E
FD37 08         MOVE1A INX               POINTS TO NEXT SOURCE CHAR.
FD38 E6 00      MOVE1  LDA B   0,X       LOADS SOURCE CHARACTER.
FD3A DF 14             STX     SRCADR    SAVES THE SOURCE POINTER.
FD3C DE 16             LDX     DSTADR    LOADS DESTINATION ADDRESS.
FD3E E7 00             STA B   0,X       STORES CHAR. IN DESTINATION.
FD40 08                INX               NEXT DESTINATION ADDRESS.
FD41 DF 16             STX     DSTADR    SAVES DESTINATION PTR.
FD43 C1 60             CMP B   #$60      TESTS IF MOVE FINSHED (CR).
FD45 26 EE             BNE     MOVE      SKIPS BACK IF NOT DONE.
FD47 39         MOVEXT RTS               RETURNS TO CALLER.
 FD45
```

```
                    *         MOVE2 SUBROUTINE
                    *
                    *      THE MOVE2 SUBROUTINE MOVES THE LAST LINE ON THE
                    * SCREEN TO THE HIGH AREA OF THE BUFFER (BUFFHI) DURING
                    * SCROLLING.  THE TEXT IS TEMPORARILY STORED ON THE STACK
                    * DURING THE MOVE.  THE MOVE IS TERMIINATED BY A "C.R.".
                    * THE TEXT IS STORED AT BUFFHI ON DOWN.
                    *
                    *
                    *
                    *
FD48 8D BE          MOVE2  BSR    OVR2     X GETS ADDR OF LAST LINE.
(FD49 FD4A 5F FD49         CLR B  -        SETS TERMINATION FOR
FD4B 37             MV21   PSH B  STACK    POPPING.
FD4C E6 00                 LDA B  0,X      LOADS SOURCE CHAR.
FD4E 08                    INX             POINTS TO NEXT CHAR.
FD4F C1 60                 CMP B  #$60     TESTS IF LINE TO "C.R."
FD51 26 F8                 BNE    MV21     MOVED TO STACK.
FD53 DE 22          MV22   LDX    BUFFHI   INIT. DESTINATION.
FD55 E7 00          MV23   STA B  0,X      STORES CHAR.
FD57 09                    DEX             POINTS TO NEXT LOCATION.
FD58 DF 22                 STX    BUFFHI   UPDATES BUFFER PTR.
FD5A 33                    PUL B  -        GETS NEXT CHAR.
FD5B C1 00                 CMP B  #00      TESTS IF ALL CHRS STORED.
FD5D 26 F6                 BNE    MV23     SKIPS BACK IF NOT STORED.
FD5F 39             MOVEX  RTS             RETURNS TO CALLER.
(FD5E

                    *      SCROLLUP MOVES ALL LINES UP 1, & CLEARS LAST LINE.
(FD5F               *
FD60 CE E000        SCRLUP LDX    #$E000   SETS CRT HOME POSITION.
FD63 E6 20          SCRP1  LDA B  $20,X    GETS CHAR FROM NEXT LINE.
FD65 E7 00                 STA B  00,X     STORES CHAR ON PREV. LINE.
FD67 08                    INX             POINTS TO NEXT LINE.
FD68 8C E1E0               CPX    #$E1E0   TESTS IF MOVE DONE.
FD6B 26 F6                 BNE    SCRP1    GOES BACK IF NOT DONE.
FD6D DF 1C                 STX    CSRPTR   SETS CSRPTR TO LAST LINE.
FD6F BD FC38               JSR    CLEAR    CLEARS LAST LINE.
FD72 39                    RTS             EXITS.
(FD73
```

```
                      *       SCRLDOWN MOVES ALL LINES DOWN ONE AND
                      * CLEARS THE TOP LINE ON THE SCREEN.
         FD74         *
FD73 CE E1DF  SCRLDN LDX     #$E1DF     INITIALIZES THE POINTER.
FD76 E6 00    SCRD1  LDA B   0,X        LOADS DATA TO BE MOVED.
FD78 E7 20           STA B   $20,X      MOVES DATA DOWN ONE LINE.
FD7A DF 1C           STX     CSRPTR     SAVES CURSOR.
FD7C 09              DEX                POINTS TO NEXT BYTE.
FD7D 8C DFFF         CPX     #$DFFF     TESTS IF MOVE FINISHED.
FD80 26 F4           BNE     SCRD1      SKIPS BACK IF NOT DONE.
FD82 C6 60           LDA B   #$60       LOADS BLANK TO CLEAR LINE.
FD84 08       SCRD2  INX                POINTS TO NEXT CHARACTER.
FD85 E7 00           STA B   0,X        CLEARS BYTE ON LINE 1.
FD87 8C E01F         CPX     #$E01F     TESTS IF LINE 1 CLEARED.
FD8A 26 F8           BNE     SCRD2      SKIPS BACK IF NOT CLEARED.
FD8C 39              RTS                RETURNS.
FD8D



                      *       OUTSTRING PRINTS OUT THE STRING BETWEEN THE
                      *       OUTBUF POINTER AND THE BUFEND POINTER.
                      *
FD8D DE 11    OUTSTR LDX     OUTBUF     BUFPTR GETS START OF TEXT.
FD8F A6 00    OUT1   LDA A   0,X        LOADS CHAR TO BE PUT OUT.
FD91 DF 1E           STX     BUFPTR     SAVES SOURCE POINTER.
FD93 BD FCAD         JSR     PUTCHR     PRINTS CHARACTER.
FD96 DE 1E           LDX     BUFPTR     RESTORES POINTER.
FD98 9C 0A           CPX     OUTEND     TESTS FOR END-OF-TEXT.
FD9A 27 03           BEQ     OUT2       EXITS IF END OF TEXT.
FD9C 08              INX                INCRE.  PTR TO NEXT CHAR.
FD9D 20 F0           BRA     OUT1       GOES BACK FOR NEXT CHAR.
FD9F 39 FDA0  OUT2   RTS                EXITS ROUTINE.
FDA0 01 Ø            NOP                UNUSED LOCATION.          AØ = ØØ
                      *
                      *       END OF EDITOR PROGRAM.
                      *
```

```
*             THE MINI-ASSEMBLER
*
*
*      THE MINI-ASSEMBLER IS A FIXED-FIELD ONE INSTRUCTION
* PER LINE 2 PASS ASSEMBLER.  THE MINI-ASSSEMBLER FORMAT
* IS DESCRIBED ON PAGES 9-2 AND 9-3 OF THE SPHERE
* OPERATORS REFERENCE MANUAL.
*      THE TWO PASSES ARE REQUIRED TO FORM THE LABEL
* ADDRESSES.  THE SECOND PASS EQUATES THE ADDRESS FOR
* LABLES REFRENCED BEFORE THEY ARE DEFINED IN THE PROGRAM.
*
*
* ON ENTRY:
* SRCASM = ADDRESS OF SOURCE TEXT TO BE ASSEMBLED.
* BUFFLO = ADDRESSED OF OBJECT CODE PRODUCED.
*
* ON EXIT:
* PCVAL (PROGRAM COUNTER VALUE) = LAST LOCATION OF
* THE ASSEMBLED OBJECT PROGRAM.
*
*
*
*
* ALGORITHM:
*
*ASMBLR:  SET PASS COUNT TO ZERO; SET PCVAL TO DSTASM;
*ASM1A:  OPERAND VALUE FORMED IN "ONDVAL":
* A GETS CHAR IN X6 (OPERAND TYPE); X GETS X+7;
* IF CHAR X6 IS A "@" THEN ONDVAL GETS VALUE FROM SYMBOL
*   TABLE ELSE ONDVAL GETS VALUE FROM ASCBIN CONVERSION;
*SYMBL:  EQUATES SYMBOL (PC VALUE IS THE " " SYMBOL
*[LABEL]) TO A LABEL VALUE:
* SYMVAL GETS PCVAL;
* IF X(1) IS AN "=" THEN SYMVAL GETS ONDVAL;
* IF X(1) IS NOT A "=" OR A SPACE THEN IF SECOND PASS THEN
*   EXIT ELSE START SECOND PASS;
* LABEL ENTRY IN SYMBOL TABLE GETS SYMVAL;
*LDOP:  PUT OPERATION CODE INTO THE OBJECT CODE:
* CONVERT X(2)-X(3) INTO BINARY;
* SAVE PCVAL;
* P.C.  GETS P.C.+1;
*OPRND:  FORM OPERAND IN OBJECT CODE:
* FORM ONDVAL INTO PROPER SIZE BASED ON CODE IN X(6);
* STORE NEW OPERAND VALUE IN MEMORY;
* P.C.  GETS P.C.+1 OR 2;
* GET NEXT LINE OF SOURCE;
* GO TO ASM1A;
*
*
```

```
FDA1 7F 0004 ASMBLR CLR      AR3       INIT. PASS CTR TO FRST PASS.)
FDA4 DE 20   ASM1   LDX      BUFFLO    SETS PC CNTR TO START OF
FDA6 DF 40          STX      PCVAL     OBJECT CODE.
FDA8 DE 26          LDX      SRCASM    LOADS ADDR FOR FIRST LINE.
FDAA DF 02   ASM1A  STX      TMP1      SAVES ADDR OF CURRENT LINE.
FDAC A6 08          LDA A    8,X       LOADS SYMBOL (LABEL).
FDAE E6 07          LDA B    7,X       LOADS OPERAND TYPE CODE.
FDB0 C1 40          CMP B    #$'@      IF @, LOADS DATA IN SYMBOL
FDB2 27 6B          BEQ      INDADR    ADDRESS, GOES TO SYMBL.
FDB4 08             INX                SETS INDEX TO START OF
FDB5 08             INX                OPERAND NUMBER.
FDB6 08             INX
FDB7 08             INX
FDB8 08             INX
FDB9 08             INX
FDBA 08             INX
FDBB BD FF22        JSR      ASCBIN    CONVRTS # TO BINARY IN B-A.
FDBE D7 2A   ASM1B  STA B    ONDVAL    STORES OPERAND VALUE IN
FDC0 97 2B          STA A    ONDVAL+1  ONDVAL.
             *
             *
             *
             * FOLLOWING FORMS THE VALUE FOR THE LABEL.
             *
FDC2 DE 02   SYMBL  LDX      TMP1      LOADS ORIG LINE PTR INTO X.
FDC4 A6 00          LDA A    0,X       LOADS SYMBOL (LABEL).
FDC6 E6 01          LDA B    1,X       LOADS LABEL CONTROL CHAR.
FDC8 DE 40          LDX      PCVAL     LABEL VALUE GETS PCVAL.
FDCA DF 2C          STX      SYMVAL
FDCC C1 3D          CMP B    #$'=      TESTS IF LABL IS EQUATED.
FDCE 26 06          BNE      ASM2      SKIPS IF NOT EQUATED.
FDD0 DE 2A          LDX      ONDVAL    LABEL VALUE (SYMVAL) GETS
FDD2 DF 2C          STX      SYMVAL    THE OPERAND VALUE.
FDD4 20 0E          BRA      ASM3      CONTINUES EVALUATION.
FDD6 C1 20   ASM2   CMP B    #$'       TESTS FOR END-OF-PROGRAM.
FDD8 27 0A          BEQ      ASM3      SKIPS IF SPACE (NOT END).
FDDA 7D 0004        TST      AR3       TESTS IF SECOND PASS.
FDDD 27 01          BEQ      ASM2A     EXITS IF SECOND PASS.
FDDF 39             RTS
FDE0 D7 04   ASM2A  STA B    AR3       SETS CTR TO SECOND PASS.
FDE2 20 C0          BRA      ASM1      GOES BACK FOR SECOND PASS.
             *
             *
             *
             * FOLLOWING PUTS THE LABLE VALUE IN THE SYMBOL TABLE.
             *
FDE4 8D 41   ASM3   BSR      SYMPTR    X GETS SYMBL TABL ENTRY ADR. E
FDE6 96 2C          LDA A    SYMVAL    STORES THE LABEL
FDE8 A7 00          STA A    0,X       ADDRESS (SYMVAL) INTO THE
FDEA 96 2D          LDA A    SYMVAL+1  SYMBOL TABEL.
FDEC A7 01          STA A    1,X
```

```
                     * FOLLOWING FORMS THE OPERATION CODE.
                     *
FDEE DE 02   LDOP    LDX     TMP1      LOADS ORIG LINE POINTER.
FDF0 08              INX               SETS X TO POINT TO
FDF1 08              INX               THE OP CODE CHARS.
FDF2 08              INX
FDF3 A6 00           LDA A   0,X       GETS OP CODE CHAR INTO A.
FDF5 81 20           CMP A   #$'       TESTS IF OP CODE EXISTS.
FDF7 27 0A           BEQ     OPRND     SKIPS IF NONEXISTANT.
FDF9 BD FF22         JSR     ASCBIN    CONVRTS OP CODE TO BINARY.
FDFC DE 40           LDX     PCVAL     LOADS POINTR TO OBJECT CODE.
FDFE A7 00           STA A   0,X       STORS OP INTO OBJECT CODE.
FE00 08              INX               SETS TO NEXT OBJ CODE LOCTN.   3rd ROM
FE01 DF 40           STX     PCVAL     SAVES P. C. POINTER.            E20.
                     *
                     *
                     *
                     * FOLLOWING STORES INTO THE OBJECT CODE THE SIZED OPERAND.
                     *
FE03 DE 02   OPRND   LDX     TMP1      LOADS SOURCE LINE POINTER.
FE05 A6 06           LDA A   6,X       LOADS OPERAND SIZE CHAR.
FE07 DE 40           LDX     PCVAL     LOADS X WITH OBJ CODE PTR.
FE09 81 45           CMP A   #$'E      TESTS LENGTH TYPE.
FE0B 2E 31           BGT     RELTIV    SKIPS IF AN "R" OPERAND.
FE0D 27 21           BEQ     EXTEND    SKIPS IF AN "E" SIZE OPRND.
FE0F 81 44           CMP A   #$'D      TESTS IF SIZE CHR EXISTS.
FE11 27 22           BEQ     DIRECT    SKIPS IF "D"COMND EXISTS.
                     *
                     *
                     *
                     * FOLLOWING GETS THE NEXT LINE.
                     *
FE13 DE 02   ASM4    LDX     TMP1      LOADS START OF LINE IN
FE15 08      ASM4A   INX               ORDER TO FIND NEXT LINE.
FE16 A6 00           LDA A   0,X       LOADS CHAR FROM SORCE LINE.
FE18 81 60           CMP A   #$60      TESTS FOR A CARRAGE RETURN.
FE1A 26 F9           BNE     ASM4A     SKIPS BAK UNTIL C.R. FOUND.
FE1C 08              INX               POINTS TO FIRST LINE CHAR.
FE1D 20 8B           BRA     ASM1A     GOES BACK TO ASSM. NEXT LINE
                     *
                     *
                     *
                     * THE FOLLOWING ARE SUBROUTINES USED BY THE MAIN CODE.
                     *
                     *
FE1F 8D 06   INDADR  BSR     SYMPTR    GETS CONTENTS OF
FE21 EE 00           LDX     0,X       SYMBOL LOCATION.
FE23 DF 2A           STX     ONDVAL    STORES AS OPERAND.
FE25 20 9B           BRA     SYMBL     RETURNS TO FIX LABEL VALUE.
                     *
FE27 48      SYMPTR  ASL A   -         MULT LABEL BY 2 TO FORM
FE28 5F              CLR B   -         POINTR INTO SYMBOL TABLE.
FE29 97 01   LOADX   STA A   TMP+1     LOADS POINTER INTO THE
FE2B D7 00           STA B   TMP       SYMBOL TABLE INTO X.
FE2D DE 00           LDX     TMP       RETURNS TO CALLER.
FE2F 39              RTS               RETURNS.
0
```

```
FE30 D6 2A    EXTEND LDA B  ONDVAL    STORES HI BYTE OF OPERAND
FE32 E7 00           STA B  0,X       INTO OBJECT CODE.
FE34 08              INX              INC PC TO POINT TO NXT WD.
FE35 96 2B    DIRECT LDA A  ONDVAL+1  STORES LO BYTE OF OPERAND
FE37 A7 00           STA A  0,X       INTO OBJECT CODE.
FE39 08              INX              INC & SAVE P.C.  TO POINT TO
FE3A DF 40           STX    PCVAL     NEXT BYTE.
FE3C 20 D5           BRA    ASM4      GOES TO WORK ON NEXT LINE.
              *
FE3E 08       RELTIV INX              INCRIMENT P.C.  PTR TO POINT
FE3F DF 40           STX    PCVAL     TO NXT BYTE & SAVE P.C..
FE41 96 2B           LDA A  ONDVAL+1  LOADS LO BYTE OF OPERAND.
FE43 90 41           SUB A  PCVAL+1   FORMS RELATIVE OFFSET.
FE45 09              DEX              INSERTS RELATIVE BYTE INTO
FE46 A7 00           STA A  0,X       OBJECT CODE.
FE48 20 C9           BRA    ASM4      GOES TO ASSMBL NEXT LINE.
              *
              *      END OF THE ASSEMBLER PROGRAM.
              *
```

```
*          DEBUGGER
*
*      THE DEBUGGER FOR THE PDS SYSTEM WAS DESIGNED TO
* PROVIDE A VERSATILE TOOL FOR USE IN PROGRAM TESTING AND
* DEBUGGING.  IT ALLOWS FOR BREAKPOINTS, MINI-ASSEMBLER
* SYMBOL TABLE REFERENCING, STACK MANIPULATION AND
* INPUT IN EITHER HEXADECIMAL, OCTAL OR DECIMAL.
* THE DEBUGGER PRINTS A PROMT CHARACTER ">" ON EVERY NEW
* LINE.  AN INSTRUCTION CAN BE TYPED IN WHENEVER THE CURRSOR
* IS BLINKING, EXCEPT WHEN A NUMBER IS BEING TYPED IN.
* THE DEBUGGER CALLS THE EDITOR WHENEVER A NUMBER IS TO BE
* INPUT, SO CORRECTIONS CAN BE MADE IF THE WRONG DIGIT IS
* TYPED IN.  THE POINTER "PCVAL" POINTS TO THE CURRENTLY
* OPENED BYTE LOCATION.  THE DEBUGGER OPERATES ON WHATEVER
* BYTE IS POINTED TO BY PCVAL.  FOR FURTHER DETAILS SEE THE
* SECTION ON THE DEBUGGER IN THE OPERATORS REFERENCE MANUAL.
*
*      THE DEBUGGER IS IMPLEMENTED BY A SMALL ROUTINE TO SET
* UP ENTRY (DEBUG) AND A LARGE ROUTINE WHICH DOES A RANGE
* COMPARE TO FIND THE PROPER COMMAND AND THEN EXECUTES THE
* COMMAND (RUNBUG).  NOTE THAT SINCE COMMANDS ARE
* INTERPRETED BY RANGE, ANY KEY STRUCK WILL PRODUCE
* A COMMAND EXECUTION, SUCH AS A "," BEING INTERPRETED
* AS A "+" COMMAND.
*
*
*
* COMMANDS:
*
* "C.R." LINE - PRINTS ">" OUT ON A NEW LINE.
* " " CHANGE - THE SPACE COMND CHANGES CONTENTS FROM Y TO Z.
* "+" OPNNXT - OPENS NEXT LOCATION.
* "-" OPNPRE - OPENS PREVIOUS LOCATION.
* "B" BRKSET - SETS A BREAKPOINT AT THE OPENED LOCATION.
* "C" CLRBRK - CLEARS BRKPOINT. MUST BE DONE BEFORE EXIT.
* "E" EXIT - PERFORM RTI - EXECUTE AT BRKPOINT LOCATION.
* "G" GOLOCN - STARTS EXECUTION AT OPENED LOCATION.
* "O" OPNLOC - OPENS LOCATION THAT IS TYPED IN AFTER "O".
* "R" OPNREG - OPENS THE TOP-OF-STACK LOCATION.
* "S" SETSTK - SETS THE STACK TO THE OPENED LOCATION.
* "T" OPNTBL - OPENS LOCATION IN SYMBOL TABLE OF NEXT CHR.
LATER VERSION * "J"     JUMP TO SUBROUTINE
*
*
*
*
* SUBROUTINES:
*
* INPCHR - INPUTS A CHAR. INTO A AND PRINTS IT.
* INPNUM - INPUTS A NUMBER INTO B-A FROM THE KEYBOARD.
* PNTBYT - PRINTS ACC A AS 2 HEX DIGITS ON THE SCREEN.
* PNTDIG - PRINTS B-A AS 4 HEX DIGITS ON THE SCREEN.
* NEWLIN - PRINTS A C.R. AND A ">" ON THE SCREEN.
* DSPADR - PRINTS BYTE ADDRESS (XXXX) AND BYTE CONTENTS (YY)
*   AS >XXXX YY ON THE SCREEN.
*
0
```

```
FE4A                    ORG     $FE4A
                 *
FE4A BD FCBE BGPOP1 JSR     SUB32    MOVES CURSOR UP ONE LINE.
FE4D 31      POPLIN INS              CLEANS UP STACK FOR
FE4E 31             INS              DISPLAY OF C.R. >.
                 *
     FE64        *
FE4F 8D 74   DEBUG  BSR     NEWLIN   PRINTS "C.R.>".
FE51 8D 09   DBUG1  BSR     INPCHR   READS IN COMMAND.
FE53 BD FC96        JSR     RTARRO   INSERT BLANK.
FE56 DE 40          LDX     PCVAL    LOADS CURRENTLY OPENED LOC.
FE58 8D 09          BSR     RUNBUG   EXECUTES DEBUG COMMAND.
FE5A 20 F5          BRA     DBUG1    GOES BACK FOR NEXT COMND.
                 *
FE5C BD FC45 INPCHR JSR     GETCHR   READ IN CHAR INTO A.
FE5F BD FCAD        JSR     PUTCHR   DISPLAYS CHARACTER.
FE62 39             RTS              RETURNS TO CALLER.
                 *
FE77             *
                 *
                 *
FE63 81 20   RUNBUG CMP A   #$'      TESTS FOR SPACE.
FE65 2D E3          BLT     BGPOP1   SKIPS IF A "C.R." (LINE).
FE67 2E 09          BGT     OPNPRE   SKIPS FOR OTHER CMND TESTS.
FE69 BD FEF4 CHANGE JSR     INPNUM   INPUTS NEW BYTE CONTENTS.
FE6C DE 40          LDX     PCVAL    LOADS OPENED BYTE LOCATION.
FE6E A7 00          STA A   0,X      STORES NEW BYTE DATA.
FE70 20 DB          BRA     POPLIN   GOES TO INPUT NEXT COMMAND.
                 *
                 *
FE72 81 2D   OPNPRE CMP A   #$'-     TESTS FOR A "-" COMMAND.
FE74 2D 05          BLT     OPNNXT   SKIPS IF A "+" COMMAND.
FE8A
FE76 2E 06          BGT     CLRBRK   SKIPS TO TEST FOR OTHER CMND
FE78 09             DEX              FORMS PREV. LOCATION ADDR.
FE79 20 2B          BRA     DSPADR   GOES TO OPEN THE LOCATION.
                 *
                 *
FE7B 08      OPNNXT INX              FORMS NEXT LOCATION ADDR.
FE7C 20 28          BRA     DSPADR   GOES TO OPEN LOCATN BYTE.
                 *
                 *
FE7E 81 43   CLRBRK CMP A   #$'C     TESTS FOR A "C" COMMAND.
FE80 2D 0A          BLT     BRKSET   SKIPS IF A "B" COMMAND.
FE82 2E 78          BGT     EXIT     SKIPS FOR NEXT COMND TEST.
FE84 DE 30          LDX     BRKADR   GETS ADDRESS OF BREAKPOINT.
FE86 96 2E          LDA A   BRKSAV   LOADS ORIG BYTE CONTENTS.
FE88 A7 00          STA A   0,X      RESTORES BYTE DATA.
FE8A 20 1A          BRA     DSPADR   GOES TO OPEN THE LOCATION.
                 *
     FEA1        *
FE8C A6 00   BRKSET LDA A   0,X      LOADS DATA OF OPNED LOCATN.
FE8E 97 2E          STA A   BRKSAV   SAVES DATA OF OPNED BYTE.
FE90 DF 30          STX     BRKADR   SAVES ADDR. OF BREAKPOINT.
FE92 86 3F          LDA A   #$3F     LOADS SOFTWARE INTUP COMND.
FE94 A7 00          STA A   0,X      SETS AN SWI AT OPNED BYTE.
FE96 20 B5          BRA     POPLIN   GOES TO NEXT LINE FOR COMND.
FEAA
```

```
FE98 30        OPNREG TSX                OPENS TOP-OF-STACK.
FE99 08               INX                PCVAL GETS STACK POINTER.
FE9A 08               INX                (CLEANS UP THE STACK).
FE9B 20 09            BRA      DSPADR    GOES TO DISPLAY THE T-O-S.
                      *
                      *
FE9D 81 53     SETSTK CMP A    #$'S      TESTS FOR AN "S" COMMAND.
FE9F 2D F7            BLT      OPNREG    SKIPS IF AN "R" COMMAND.
FEA1 2E 13            BGT      OPNTBL    SKIPS FOR THE "T" COMMAND.
FEA3 35               TXS                STACK POINTER GETS PCVAL.
FEA4 20 A9            BRA      DEBUG     RETURNS TO INPUT COMMAND.
                      *
                      *
                      *
                      *
                      * FOLLOWING ARE SUBROUTINES USED BY DEBUGGER.
                      *
                      *
                      * FOLLOWING DISPLAYS THE LOCATION ADDR. & CONTENTS.
                      *
FEA6 DF 40     DSPADR STX      PCVAL     SAVES OPENED LOCATION ADDR.
FEA8 8D 1B     DSPAD1 BSR      NEWLIN    PRINTS A "C.R." AND ">".
FEAA 8D 10            BSR      PNTDIG    PRINTS OUT "PCVAL" IN HEX.
FEAC BD FC96          JSR      RTARRO    PRINTS A SPACE.
FEAF DE 40            LDX      PCVAL     LOADS PTR. TO OPEND LOC.
FEB1 A6 00            LDA A    0,X       LOADS DATA FROM LOCATN.
FEB3 8D 17            BSR      PNTBYT    PRINTS DATA IN HEX FORMAT.
FEB5 39               RTS                RETURNS TO INPUT COMMAND.
                      *
                      *
FEB6 8D A4     OPNTBL BSR      INPCHR    LOADS A WITH SYMBOL (LABL).
FEB8 48               ASL A    -         ALIGNS ADDRESS FOR
FEB9 5F               CLR B    -         SYMBOL TABLE ENTRY.
FEBA 20 5C            BRA      OPNLC1    SAVES AND DISPLAYS ADDRESS.
                      *
                      *
FEBC 96 40     PNTDIG LDA A    PCVAL     PRINTS THE TWO HI HEX DIGITS
FEBE 8D 0C            BSR      PNTBYT    OF OPENED ADDRESS.
FEC0 96 41            LDA A    PCVAL+1   PRINTS OUT 2 LOW HEX DIGITS
FEC2 8D 08            BSR      PNTBYT    OF THE OPENED ADDRESS.
FEC4 39               RTS                RETURNS TO CALLING CODE.
                      *
                      *
FEC5 CE 0D3E   NEWLIN LDX      #$0D3E    LOADS "C.R." & ">".
FEC8 DF 35            STX      IOBUFF    STORES FOR PRINTING.
FECA 20 1A            BRA      PNTBUF    SKIPS TO PRINT CHAR.
                      *
                      *
                      * FOLLOWING PRINTS OUT 2 HEX DIGITS.
 FF02                 *
FECC CE 0010   PNTBYT LDX      #16       LOADS 16 FOR BASE.
FECF DF 04            STX      ARB       STORES FOR CONVERSION.
FED1 5F               CLR B    -         CLEARS HI 2 DIGITS.
FED2 CE 0035          LDX      #IOBUFF   LOADS OUTPUT BUFF ADDRESS.
```

```
                          * FOLLOWING CONVERTS BYTE TO HEX WITH LEADING ZEROS.
                          *                                                     FF0B
FED5 D7 36       CONVRT   STA B   IOBUFF+1 SETS BYTE FOR SECOND DIGIT.
FED7 BD FF64              JSR     BINASC   CONVERTS TO ASCII DIGITS.
FEDA 96 36                LDA A   IOBUFF+1 TESTS BOTH DIGITS CONVTD.
FEDC 26 08                BNE     PNTBUF   SKIPS IF BOTH DIGITS NOT 0.
FEDE 96 35                LDA A   IOBUFF   MOVES LOW DIGIT.
FEE0 97 36                STA A   IOBUFF+1
FEE2 86 30                LDA A   #$'0     HIGH DIGIT GETS A "0".
FEE4 97 35                STA A   IOBUFFF
                          *
FEE6 CE 0035     PNTBUF   LDX     #IOBUFF  LOADS ADDR OF HEX DIGITS.
FEE9 DF 11                STX     OUTBUF   STORES AS OUTPUT BUFFER.
FEEB CE 0036              LDX     #IOBUFF+1  STORES ADDR. OF
FEEE DF 0A                STX     OUTEND   END OF DIGITS PRINTED.
FEF0 BD FD8D              JSR     OUTSTR   PRINTS OUT BYTES.
FEF3 39                   RTS              RETURNS TO CALLING PROGRAM.
                          *
                          *
                          *
                          * FOLLOWING INPUTS A 16 BIT NUMBER INTO THE BA REG.
                          *
FEF4 BD FC6E     INPNUM   JSR     EDITIN   INPUTS A STRING OF DIGITS.
FEF7 DE 24                LDX     SCNPTR   LOADS ADDR OF FIRST DIGIT.
FEF9 8D 27                BSR     ASCBIN   CONVERTS TO BINARY IN BA.
FEFB 39                   RTS              RETURNS TO CALLER.
                          *
                          *                                                     FEC3
FEFC 81 47       EXIT     CMP A   #$'G     TESTS IF AN EXIT (E) COMMD.
FEFE 27 1E                BEQ     GOLOCN   SKIPS TO THE "GO" COMMAND.
FF00 2C 10                BGE     OPNLOC   SKIPS FOR NEXT COMMD TEST.   4TH ROM
FF02 31                   INS              CLEARS UP THE STACK.
FF03 31                   INS
FF04 30                   TSX              INDEX GETS STACK POINTER.
FF05 E6 05                LDA B   5,X      LOADS RETURN ADDRESS.
FF07 A6 06                LDA A   6,X
FF09 80 01                SUB A   #1       SUB 1 FROM RETURN ADDR.
FF0B C2 00                SBC B   #0
FF0D E7 05                STA B   5,X      RESTORES ADDR. TO BRKPT
FF0F A7 06                STA A   6,X      LOCATION.
FF11 3B                   RTI              RETURNS FROM BREAKPOINT.
                          *
                          *
FF12 81 4F       OPNLOC   CMP A   #'O      TESTS FOR AN "O" COMMAND.
FF14 2E 87                BGT     SETSTK   SKIPS IF AN "R", "S" ETC.
FF16 8D DC                BSR     INPNUM   LOADS A 16 BIT NUMBER.
FF18 D7 40       OPNLC1   STA B   PCVAL    STORES NEW OPENED
FF1A 97 41                STA A   PCVAL+1  LOCATION ADDRESS.
FF1C 20 8A                BRA     DSPAD1   DISPLAYS CONTENTS OF LOC.
                          *
                          *
FF1E 31          GOLOCN   INS              CLEANS UP THE STACK.
FF1F 31                   INS
FF20 6E 00                JMP     0,X      JUMPS TO USERS PROGRAM.
                          *                                                     FFDA
                          *       END OF DEBUGGER
                          *
0
```

```
*        UTILITY PROGRAMS
*
*
*
*
*              ASCII TO BINARY CONVERSION.
*
*      THE ASCII TO BINARY ROUTINE CONVERTS FROM AN ASCII
* NUMBER STRING POINTED TO BY X TO AN UNSIGNED 16 BIT
* BINARY NUMBER IN BA (ACC B HAS THE HI BYTE, ACC A HAS
* THE LO BYTE).  THE ASCII STRING IS TERMINATED BY A NON
* HEXADECIMAL CHARACTER.  UPON EXITING, THE INDEX REGISTER
* WILL POINT TO THE NEXT CHARACTER AFTER THE NUMBER
* STRING.  THE BASE OF THE NUMBER STRING IS PASSED TO
* THE ROUTINE IN ARA (ARA IS THE ARITHMETIC REGESTER A
* LOCATED IN BYTES 06 AND 07 OF LOW MEMORY).  IF THE
* ROUTINE IS ENTERED WITH A KNOWN BASE, PUT THE BASE
* (BETWEEN 2 AND 16) IN ARA AND ENTER THE ROUTINE AT
* THE ENTRY POINT ENTR2.
*
*
*
*      CONVERSION FORMULA:
* ASCII NUMBBER STRING X[4],X[3],X[2],X[1] IN
*BASE Y;
* BINARY NUMBER =
*      X[4]*Y↑3+X[3]*Y↑2+X[2]*Y↑1+X[1]*Y↑0         OR
* BINARY NUMBER =
*      (((0*Y+X[4])*Y+X[3])*Y+X[2])*Y+X[1]
*WHERE ↑ IS THE EXPONENT OPERATOR,
* X IS A CHARACTER & Y IS THE BASE.
*
*
*
*      ALGORITHM:
*ASCBIN:  FORM THE BASE IN ARA BASED ON THE FIRST CHAR.
*  OF THE NUMBER STRING; INCREMENT CHAR. PTR. IN X;
*ENTR2:  NUMBER (IN BA) GETS 0;
*NXTCHR:  IF THE CURRENT CHAR. POINTED TO BY X IS NOT A
*  DIGIT THEN EXIT ELSE INCREMT CHARACTER PTR IN INDEX;
*  CONVERT DIGIT TO BINARY;
*  NUMBER GETS NUMBER * BASE;
*  NUMBER GETS NUMBER + DIGIT;
*  GO TO OPERATE ON THE NEXT DIGIT (NXTCHR);
*
```

```
      FF22
FF22 A6 00    ASCBIN LDA A  0,X       GETS CHR TO FORM BASE.
FF24 81 2E           CMP A  #$'.      TESTS FOR DECML STRNG.
FF26 2D 06           BLT    OCT       SKIPS IF BASE 8 (*).
FF28 2E 09           BGT    HEX       SKIPS IF BASE 16.
FF2A 86 0A           LDA A  #10       LOADS BASE 10 FOR CONVERSN.
FF2C 20 02           BRA    ASC1      SKIPS TO INC. TEXT POINTR.
FF2E 86 08    OCT    LDA A  #8        LOADS BASE 8 FOR CONVERSION.
FF30 08       ASC1   INX              INCREMENT PTR TO NEXT CHAR.
FF31 20 02           BRA    ASC2      SKIPS TO SAVE BASE.
FF33 86 10    HEX    LDA A  #16       LOADS BASE 16 FOR CONVERN.
FF35 97 07    ASC2   STA A  AR0       SAVES BASE IN BASE#.
              *
   FF37       *
FF37 5F       ENTR2  CLR B  NUMBER    GETS 0.
FF38 37              PSH B  -         (LOW NUMBER ON STACK).
FF39 D7 06           STA B  AR1       CLEARS HI OF BASE.
FF3B A6 00    NXTCHR LDA A  0,X       GETS CHAR TO CONVERT.
FF3D 08              INX              INC TO NEXT CHARACTER.
FF3E 81 30           CMP A  #$'0      TESTS FOR END-OF-STRING.
FF40 2D 20           BLT    AEXIT     EXITS IF END.
FF42 80 30           SUB A  #$'0      FORMS B. C. D. NUMBER.
FF44 81 0A           CMP A  #10       TESTS IF DECIMAL DIGIT.
FF46 2D 0A           BLT    ASC3      SKIPS IF DECIMAL.
FF48 81 10           CMP A  #16       TESTS FOR END OF STRING.
FF4A 2F 16           BLE    AEXIT     EXITS IF NOT A HEX DIGIT.
FF4C 80 07           SUB A  #7        FORMS A HEX B. C. D. DIGIT.
FF4E 81 10           CMP A  #16       TESTS FOR END-OF-STRING.
FF50 2C 10           BGE    AEXIT     EXITS IF CHAR > "F".
FF52 97 08    ASC3   STA A  DIGIT     SAVES DIGIT FOR ADD.
              *
FF54 DF 00    CNVASC STX    TMP       SAVES INDEX REG FOR MULT.
FF56 32              PUL A  -         RESTORES LO OF "NUMBER".
FF57 8D 3A           BSR    MULT      NUMBER GETS NUMBER * BASE.
FF59 9B 08           ADD A  DIGIT     NUMBER GETS NUMBER + DIGIT.
FF5B C9 00           ADC B  #0
FF5D 36              PSH A  -         SAVES LO OF NUMBER.
FF5E DE 00           LDX    TMP       RESTORES STRING POINTER.
FF60 20 D9           BRA    NXTCHR    GOES TO CONVRT NEXT CHAR.
FF62 32       AEXIT  PUL A  -         RESTORES "NUMBER" IN BA.
FF63 39              RTS              RETURNS TO CALLING PROGRAM.
 FF63
```

```
                 *           BINARY TO ASCII
                 *
                 *
                 *      THE BINARY TO ASCII CONVERSION ROUTINE CONVERTS
                 * A 16 BIT BINARY NUMBER IN THE BA REGISTR (REG B & REG A)
                 * TO A STRING OF ASCII DIGITS.  THE ASCII STRING CAN BE IN
                 * ANY BBASE FROM BASE 2 THROUGH BASE 41.  THE VALUE OF THE
                 * BASE IS LOCATED IN THE ARITHMETIC PSEUDO-REGISTER ARB
                 * (ARB IS LOOCATED IN BYTE AR3 [LOC 4] AND AR2 [LOC 5]).
                 * WHEN THE ROUTINE IS ENTERED, THE POINTER TO THE OUTPUT
                 * LOCATION IS PASSED IN THE INDEX REG.  WHEN THE ROUTINE
                 * EXITS, THE INDEX POINTS TO THE LAST DIGIT IN THE STRING
                 * PLUS ONE.
                 *      CONVERSION IS DONE BY THE METHOD OF REPEATED
                 * DIVISION.  THE LOW ORDER DIGIT IS FORMED FIRST.  THE
                 * DIGITS ARE THEN PLACED ON THE STACK UNTIL CONVERSION IS
                 * COMPLETED.  THE DIGITS ARE THEN POPPED OFF THE STACK
                 * AND PLACED IN THE OUTPUT STRING.  THE TOP-OF-STACK IS
                 * INITIALIZED TO ALL ONES TO TELL WHEN ALL THE DIGITS
                 * HAVE BEEN POPPED OFF THE STACK.  AFTER THE DIVISION, THE
                 * DIGIT (THE REMAINDER OF THE DIVISION OPERATION)
                 * IS LOCATED IN THE AR 0 PART OF ARA (BYTE 7).  WHEN
                 * THE QUOTIENT OF THE DIVISION IS 0, THEN THE CONVERSION
                 * IS COMPLEATED.
                 *
                 *
     FF64        *
FF64 DF 00       BINASC STX    TMP      SAVES OUTPUT POINTER.
FF66 34                 DES             /SETS THE TOP-OF-STACK TO
FF67 30                 TSX             /ALL ONES TO TELL END OF
FF68 6F 00              CLR    0,X      /CHAR STRING (LAST CHAR IS
FF6A 63 00              COM    0,X      /PUT ON STACK FIRST).
FF6C DE 04       BIN1   LDX    ARB      RESTORES DEVISOR (BASE).
FF6E DF 06              STX    ARA
FF70 8D 3D              BSR    DIVIDE   * QUOTIENT IN BA GETS THE
                 * REMAINDER OF # TO BE CONVERTED; REMAINDER IN ARA GETS
                 * THE LOW ORDER DIGIT.
FF72 97 02              STA A  TMP1     SAVES A OF BA.
FF74 96 07              LDA A  AR0      LOAD DIGIT (REMAINDER).
FF76 36                 PSH A  -        STACK DIGIT (REVERSE ORDER). G
FF77 96 02              LDA A  TMP1     RESTORES A OF BA.
FF79 4D                 TST A  -        /TESTS IF QUOTIENT IS = 0
FF7A 26 F0              BNE    BIN1     /(SIGNIFYING THAT
FF7C 5D                 TST B  -        /THE CONVERRSION
FF7D 26 ED              BNE    BIN1     /IS DONE).
                 *
FF7F DE 00       BINSTR LDX    TMP      RESTORES OUTPUT POINTER.
FF81 32          BIN3   PUL A  -        UNSTACK A DIGIT.
FF82 4D                 TST A  -        TESTS IF NEG (END?).
FF83 2A 01              BPL    BIN4     SKIPS IF A DIGIT.
FF85 39                 RTS             EXITS FROM SUBROUTINE.
FF86 81 09       BIN4   CMP A  #9       TESTS IF RESULT IS HEX.
FF88 2F 02              BLE    BIN5     SKIPS IF DIGIT NOT HEX.
FF8A 8B 07              ADD A  #7       FORMS HEX VALUE OF DIGIT.
FF8C 8B 30       BIN5   ADD A  #$'0     FORMS DECIMAL CHARACTER.
FF8E A7 00              STA A  0,X      OUTPUTS CHARACTER.
FF90 08                 INX             POINTS TO NEXT CHARACTER.
FF91 20 EE              BRA    BIN3     GOES BACK FOR NEXT DIGIT.
FF91
```

```
                   *         MULTIPLY ROUTINE
                   *
                   *     THE MULTIPLY ROUTINE MULTIPLIES TWO 16 BIT BINARY
                   * NUMBERS TOGETHER TO PRODUCE A 16 BIT RESULT.  THE BA
                   * REGISTERS AND ARA (BYTES 6 & 7) REGISTER ARE USED.
                   * THE CONTENTS OF ARA ARE UNCHANGED UPON PROGRAM EXIT.
                   *
                   *
                   *  BA GETS BA * ARA
                   *
                   *
                   *     MULTIPLYING IS ACCOMPLISHED BY REPEATED ADDITIONS
                   * OF ONE OF THE OPERATORS (OPERATOR ARA) INTO THE RESULT.
                   * THE RESULT STARTS OUT WITH A ZERO VALUE AND IS SHIFTED
                   * OVER ONE AFTER EACH ADDITION.  THE HIGHEST ORDER VALUE
                   * IS ADDED IN FIRST AND TTHEN, GOING TO THE RIGHT,
                   * (THUS SHIFTING THE ANSWER LEFT ONE TO BRING IN THE NEXT
                   * RIGHTMOST DIGIT) GETTING THE NEXT LOWERMOST SIGNIFICANT
                   * DIGIT.  THE NEXT RIGHTMOST BIT OF THE OTHER OPERAND
                   * (THE ONE ORIGINALLY IN BA) IS TESTED, AND IF ONE,
                   * ANOTHER ADDITION TAKES PLACE.   THIS IS REPEATED UNTIL
                   * THE FINAL SUM IS FORMED.
                   *
                   *
                   *
                   *  MULTIPLY ALGORITHM:
                   *
                   *MULT:   STACK BA; BA GETS 0; SET COUNT VALUE TO 16;
                   *MUL1:   SHIFT BA LEFT 1;
                   *  SHIFT LEFT ORIG BA VALUE ON STACK INTO CARRY;
                   *  IF CARRRY = 0 THEN GO TO MUL2
                   *  BA GETS BA + ARA;
                   *MUL2:   DECREMENT COUNT;
                   *  IF COUNT # 0 THEN GO TO MUL1 ELSE EXIT.
                   *
                   *
                   *
 FF93              *
 FF93 36           MULT    PSH A    -         PUTS THE ORIGINAL CONTENTS
 FF94 37                   PSH B    -         OF BA ONTO THE STACK.
 FF95 86 10                LDA A    #16       LOADS COUNT VALUE
 FF97 36                   PSH A    -         ONTO THE STACK.
 FF98 4F                   CLR A    -         BA GETS ZEROED.
 FF99 5F                   CLR B
 FF9A 30                   TSX                SET INDEX TO STACK.
 FF9B 48           MUL1    ASL A    -         SHIFT LEFT BA.
 FF9C 59                   ROL B
 FF9D 68 02                ASL      2,X       SHIFTS ORIG. BA OPERAND
 FF9F 69 01                ROL      1,X       ONE LEFT INTO CARRY.
 FFA1 24 04                BCC      MUL2      SKIPS ADDING IF CARRY = 0.
 FFA3 9B 07                ADD A    AR0       BA GETS BA + ARA.
 FFA5 D9 06                ADC B    AR1
 FFA7 6A 00        MUL2    DEC      0,X       TESTS IF DONE.
 FFA9 26 F0                BNE      MUL1      GOES BACK IF NOT DONE.
 FFAB 31                   INS                CLEANS UP THE STACK.
 FFAC 31                   INS
 FFAD 31                   INS
 FFAE 39                   RTS                EXITS ROUTINE.
 FEAE
```

```
                *          DIVIDE ROUTINE
                *
                *     THE DIVIDE SUBROUTINE DIVIDES THE 16 BIT NUMBER
                * IN THE BA REGESTERS BY THE 16 BIT NUMBER IN THE PSEUDO
                * REGESTER ARA (LOCATED IN BYTES 6 & 7).  UPON EXITING,
                * BA WILL CONTAIN THE QUOTIENT OF THE DIVISION AND ARA
                * WILL CONTAIN THE REMAINDER.  THE DIVIDEND BA IS DIVIDED
                * BY THE DIVISOR ARA (I.E.  BA/ARA ).
                *
                *
                *
                *
                *  DIVIDE ALGORITHM:
                *
                *DIVIDE:  X3,4 GETS BA (BA IS PUT ON THE STACK);
                *  X1,2 GETS THE 16 BIT ARA VALUE (ARA PUT ON THE STACK);
                *  COUNT GETS 1 + THE NUMBER OF NONSIGNIFICANT BITS IN
                * THE DIVISOR [LEFT JUSTIFY X1,2 TO FIRST 1 BIT, COUNT
                * GETS 1 + THE # OF LEADING ZEROS IN X1,2] [COUNT WILL BE
                * FROM 1 TO 17];
                *  BA GETS X3,4 [RESTORES BA];
                *  X3,4 GETS 0 [INITIALIZES THE QUOTIENT];
                *DIV3:  BA GETS BA - X1,2;
                *  IF THERE WAS A BORROW [I.E. DIVIDEND IN BA < DIVISOR
                * IN X1,2 (ARA)] THEN BA GETS BA + X1,2 [ORIGINAL BA
                * VALUE RESTORED] & CARRY CLEARED ELSE CARRY IS SET;
                *DIV5:  X3,4 [QUOTIENT] GETS CARRY LEFT SHIFTED IN;
                *  X1,2 [DIVISOR] GETS SHIFTED RIGHT ONE PLACE WITH ZERO
                * FILLED IN FROM THE LEFT SIDE;
                *  DECREMENT COUNT;
                *  EXIT IF DONE ELSE GO TO DIV3;
                *
                *
                *
                *
FFAF            *
FFAF 36         DIVIDE PSH A  -      LOADS DIVIDEND INTO X3,4.
FFB0 37                PSH B
FFB1 96 06             LDA A  AR1    LOADS DIVISOR FROM ARA.
FFB3 D6 07             LDA B  AR0
FFB5 37                PSH B  -      PUTS DIVISOR INTO X1,2.
FFB6 36                PSH A
FFB7 34                DES           SET UP SPACE FOR COUNT.
FFB8 30                TSX           INDEX GETS STACK POINTER.
FFB9 86 01             LDA A  #1     INITIALIZE COUNT.
FFBB 6D 01             TST    1,X    TESTS FOR HI DIVISR BIT ON.
FFBD 2B 0B             BMI    DIV2   SKIPS IF ON.
FFBF 4C         DIV1   INC A  -      COUNTS LEADING ZEROS.
FFC0 68 02             ASL    2,X    LEFT JUSTIFIES X1,2.
FFC2 69 01             ROL    1,X
FFC4 2B 04             BMI    DIV2   SKIPS IF NO LEADING ZERO.
FFC6 81 11             CMP A  #17    TESTS FOR ALL ZERO DIVISOR.
FFC8 26 F5             BNE    DIV1   GOES BACK IF BITS LEFT.
FFCA A7 00      DIV2   STA A  0,X    SETS COUNTER.
FFCC E6 03             LDA B  3,X    BA GETS ORIGINAL
FFCE A6 04             LDA A  4,X    DIVIDEND VALUE.
FFD0 6F 03             CLR    3,X    CLEARS X3,4 FOR FORMATION
FFD2 6F 04             CLR    4,X    OF THE QUOTIENT.
```

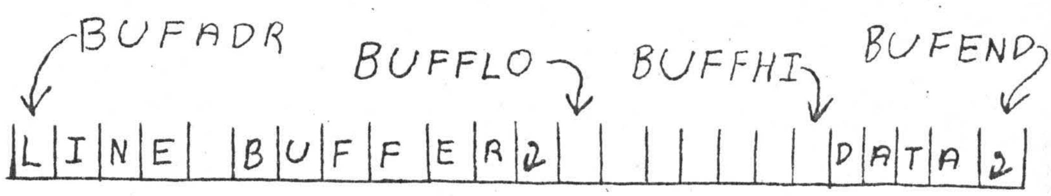
```
FFD4 A0 02   DIV3   SUB A  2,X      START OF DIVIDE LOOP.
FFD6 E2 01          SBC B  1,X
FFD8 24 07          BCC    DIV4     SKIP IF DIVIDEND < DIVISOR.
FFDA AB 02          ADD A  2,X      RESTORES DIVIDEND IN BA.
FFDC E9 01          ADC B  1,X
FFDE 0C             CLC             CLEARS THE CARRY.
FFDF 20 01          BRA    DIV5     SKIPS WITH CARRY CLEAR.
FFE1 0D      DIV4   SEC             SETS CARRY TO 1.
FFE2 69 04   DIV5   ROL    4,X      SHIFT CARRY INTO
FFE4 69 03          ROL    3,X      QUOTIENT X3,4
FFE6 64 01          LSR    1,X      SHIFTS DEVISOR X1,2
FFE8 66 02          ROR    2,X      RIGHT ONE.
FFEA 6A 00          DEC    0,X      DECREMENTS COUNTER.
FFEC 26 E6          BNE    DIV3     GOES BACK IF NOT DONE.
FFEE D7 06          STA B  AR1      STORES REMAINDER IN ARA.
FFF0 97 07          STA A  AR0
FFF2 31             INS             CLEANS UP THE STACK.
FFF3 31             INS
FFF4 31             INS
FFF5 33             PUL B  STORES   QUOTIENT IN BA.
FFF6 32             PUL A
FFF7 39             RTS             EXITS ROUTINE.
   FFF7      *
             *
             *  LOCATION OF DEBUG
FFF8 0104    IRQ    FDB    $0104    INTERRUPT REQUEST VECTOR.
FFFA FE4F    SWI    FDB    $FE4F    SOFTWARE INT. VECTOR ADDR.
FFFC 0108    NMI    FDB    $0108    NIN-MASKABLE-INT. VECT.
FFFE FC00    RST    FDB    $FC00    RESTART VECTOR ADDRESS.
             *
             *
             *
             *  END OF PDS SOURCE LISTING.
             *
             *
                    END
TOTAL ERRORS 00000
```

2A


```
                         NAM    PDS-V3A
                         OPT    O,NOG
                *
                *        PROGRAMMED BY ERIC JAMESON
                *
                *        COPYRIGHT 1976 SPHERE CORPORATION
                *        791 SO. 500 W.;  BOUNTIFUL, UTAH   84010
                *
                *        SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
                *        DISTRIBUTION AND USE OF THE PDS SOFTWARE.
                *        NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
                *        WRITTEN PERMISSION OF SPHERE CORP.
                *
                *
                * THE PROGRAM DEVELOPMENT SYSTEM (PDS V3A) IS A SET OF
                * PROGRAMS RESIDING ON ERASABLE PROGRAMMABLE READ ONLY
                * MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS
                * SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE
                * DEVELOPMENT OF COMPUTER PROGRAMS.
                *     TOWARD THIS END, THE 4 PDS EPROMS CONTAIN A CURSOR
                * BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING
                * AID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16
                * BIT MULTIPLY AND DIVIDE, ASCII-TO-BINARY, AND
                * BINARY-TO-ASCII ROUTINES.
                *
                *
                *
                *                  MEMORY MAP
                *
        0000    TMP     EQU     $00
        0002    TMP1    EQU     $02
        0004    ARB     EQU     $04       16 BIT ACC.  PSEUDO REG B.
        0004    AR3     EQU     $04       _HI BYTE OF ARB.
        0003    AR2     EQU     $03       _LO BYTE OF ARB.
        0006    ARA     EQU     $06       16 BIY ARITH PSEUDO REG A.
        0006    AR1     EQU     $06       _HI BYTE OF ARA.
        0007    AR0     EQU     $07       _LO BYTE OF ARA.
        0008    DIGIT   EQU     $08       BYTE USED BY ASCBIN FOR TMP.
        000A    OUTEND  EQU     $0A       END OF OUTPUT BUFFER TEXT.
        000C    BUFADR  EQU     $0C       START OF I/O BUFFER (PTR)
        000E    BUFEND  EQU     $0E       PTR.  TO END OF I/O BUFFR.
        0011    OUTBUF  EQU     $11       START OF OUTPUT BUFFER.
        0014    SRCADR  EQU     $14       SOURCE FOR TEXT MOVES.
        0016    DSTADR  EQU     $16       DEST.  ADDR.  FOR TEXT MOVE.
        001A    ENDMEM  EQU     $1A       LAST ADDRES OF REAL MEMORY.
        001C    CSRPTR  EQU     $1C       PTR TO CURSER ON SCREEN.
        001E    BUFPTR  EQU     $1E       TEMP PTR USED BY OUTSTR.
        0020    BUFFLO  EQU     $20       PTR TO END OF LOW EDIT TXT.
        0022    BUFFHI  EQU     $22       PTR TO START OF HI TEXT.
        0024    SCNPTR  EQU     $24       PTR.  TO BUFFRD TXT START.
        0026    SRCASM  EQU     $26       PTR TO ASSMBLR SOURCE CODE.
        002A    ONDVAL  EQU     $2A       HAS ASSMBLR OPERND VALUE.
        002C    SYMVAL  EQU     $2C       VALUE PUT IN ASSM.  SYMTBL.
        002E    BRKSAV  EQU     $2E       TEMP SAVE FOR BRKPT DATA.
        0030    BRKADR  EQU     $30       ADDDRESS OF BREAKPOINT.
        0032    EDIT    EQU     $32       0 IF EDITOR IS NOT RUNNING.
        0035    IOBUFF  EQU     $35       I-O BUFFER FOR DEBUGGER.
        0040    PCVAL   EQU     $40       PROGRAM COUNTER FOR ASSM.
```

```
                    *                   INITIALIZATION
                    *
                    *
                    *
                    *   THE INITIALIZATION ROUTINES SET UP THE INITIAL VALUES
                    * UPON SYSTEM RESET.
                    *
                    *
FC00                       ORG    $FC00
FC00 8E 01FF RESTRT LDS    #$1FF      SETS STACK POINTER
FC03 30                    TSX               MOVES STK PTR TO INDEX REG
FC04 DF 26                 STX    SRCASM     SETS ASSEMBLR OUTPUT PTR
FC06 DF 0C                 STX    BUFADR     INIT INPUT BUFFR ADDR.
FC08 86 07                 LDA A  #07        INHTLZ. KEYBOARD PIA.
FC0A B7 F001               STA A  $F001
FC0D CE 0FFF               LDX    #$FFF      LAST LOCATN OF MEMORY.
FC10 DF 0E                 STX    BUFEND     INIT END-OF-EDIT BUFFR.
FC12 DF 1A                 STX    ENDMEM     INIT END-OF-MEM ADDR.
                    *
                    *
                    *
                    *
                    *
                    *                   COMMAND LANGUAGE
                    *
                    *    THIS EXECUTIVE ACCEPTS COMMANDS FROM THE KEYBOARD TO
                    * DETERMINE WHAT UTILITY IS TO BE RUN.  INVALID COMMANDS
                    * WILL SPACE THE CURSOR DOWN ONE LINE.  DO NOT SPACE OFF THE
                    * BOTTOM OF THE SCREEN.
                    *
                    *
FC14 8D 1C   EXEC          BSR    HOME       CSRPTR IS HOMED.
FC16 8D 20                 BSR    CLEAR      CLEARS SCREEN
FC18 BD FD16 EXEC1         JSR    CRLF       NEW LINE.
FC1B BD FE5C               JSR    INPCHR     GETS & DISPLAYS CHR.
FC1E 81 01                 CMP A  #$01       TESTS IF ASSEMBLY COMMD.
FC20 26 03                 BNE    EXEC2      SKIPS IF OTHER COMMAND.
FC22 BD FDA1               JSR    ASMBLR     JUMPS TO ASSEMBL PRROGRM.
FC25 81 05   EXEC2         CMP A  #$05       TESTS IF CONTRL 'E'.
FC27 26 02                 BNE    EXEC3      SKIPS IF NOT EDIT CMD.
FC29 8D 35                 BSR    EDITOR     JUMPS TO EDIT TEXT.
FC2B 81 04   EXEC3         CMP A  #$04       TESTS FFOR CONTRL 'D'.
FC2D 26 E9                 BNE    EXEC1      LOOPS BACK IF NOT DEBUG.
FC2F 7E FE4F               JMP    DEBUG      JUMPS TO DEBUGGER.
```

```
*                    THE EDITOR
*
*
*        THE EDITOR ALLOWS INPUT FROM THE KEYBOARD INTO A
* BUFFER MEMORY.  INPUT IS DISPLAYED ON THE SCREEN.  WHEN
* IT IS TYPED IN, THE SCREEN TEXT CAN THEN BE EDITED BY USE
* OF THE CURSOR.  WHEN THE SCREEN IS FULL OR EDITING IS
* FINISHED, THE DATA IS SCROLLED OFF THE SCREEN INTO THE
* EDIT BUFFER.  WHEN TEXT IS SCROLLED OFF THE TOP OF THE
* SCREEN, IT IS STORED FROM THE BUFFER ADDRESS POINTER
* (BUFADR) TO THE LOW BUFFER POINTER (BUFFLO).  BUFFLO
* POINTS TO THE END OF TEXT + ONE (I.E. IT POINTS TO THE
* FIRST UNUSED BYTE).  WHEN IT IS SCROLLED OFF THE BOTTOM
* OF THE SCREEN, IT IS STORED IN THE TOP OF THE EDIT BUFFER.
* THE TEXT GOES FRROM THE HIGH BUFFER POINTER (BUFFHI) TO
* THE END OF BUFFER POINTER (BUFEND).  BUFFHI POINTS TO
* THE END OF TEXT - ONE (I.E. IT POINTS TO THE LAST UNUSED
* BYTE IN THE BUFFER).  WHEN THE TEXT IS SCROLLED UP
* OFF THE TOP OF THE SCREEN, TEXT IS TAKEN FROM THE HIGH
* AREA OF THE EDIT BUFFER AND DISPLAYED ON THE LAST LINE OF
* THE SCREEN.  WHEN TEXT IS SCROLLED DOWN OFF THE BOTTEM,
* TEXT, IF ANY EXISTS, IS MOVED FROM THE LOW EDIT BUFFER
* AREA TO THE TOP LINE OF THE SCREEN.
*
*
*
*
*
*
*
*
*
*
```




```
*
*                    POINTERS USED
*
* CSRPTR   POSITION OF CHARACTERS INSERTED ON THE SCREEN.
* SCNPTR   POSITION OF START OF EDITED TEXT ON SCREEN.
* BUFADR   START OF TEXT BUFFER IN MAIN MEMORY.
* BUFEND   END OF TEXT BUFFER IN MAIN MEMORY.
* BUFFLO   END OF TEXT SCROLLED OFF TOP OF SCREEN.
* BUFFHI   START OF TEXT SCROLED OFF BOTTOM OF SCREEN.
* BUFLEN   NOT CURRENTLY USED.
*
*
*
*
*
*                 EDITOR COMMANDS
*
* "UP ARROW"  MOVES CURSOR UP ONE LINE; CSRPTR GETS
*  CSRPTR-32; CALL NDRFLO.
*
* "DOWN ARROW"  MOVES CURSOR DOWN ONE LINE; CSRPTR GETS
*  CSRPTR+32; CALL OVRFLO.
*
* "RIGHT ARROW"  MOVES CURSOR ONE POSITION RIGHT; CSRPTR
*  GETS CSRPTR+1; CALL OVRFLO.
```

```
* "LEFT ARROW"  MOVES CURSOR ONE POSITION LEFT; CSRPTR
*  GETS CSRPTR-1; CALL NDRFLO.
*
* "CONTROL & LEFT ARROW (ON KEYBOARD)"  LEFT JUSTIFY CURSOR;
*  CSRPTR GETS CSRPTR TRUNCATED; CALL NDRFLO FOR SCNLOC CHK.
*
* "PUTCHR"  OUTPUTS CHARACTER; CSRPTR GETS
*  CSRPTR+1; GOES TO OVRFLO.
*
* "ENDCHR"  TERMINATION CHAR; CLEAR EDIT FLAG;
*  EXIT THE EDITOR.
*
* "HOME"  HOMES CURSOR POINTER; CSRPTR GETS E000; NDRFLO.
*
* "CLEAR"  CSRPTR TO END OF THE SCREEN GETS SPACES.
*
* "CTRL I"  INSERT A LINE AT THE LAST LINE ON THE SCREEN;
*  CALL OVR1 (SCROLLS UP ONE LINE); CSRPTR GETS E1E0.
*
* "CTRL D"  DELETE LAST LINE; SCROLL DOWN (UNDR2);
*  CSRPTR GETS E1E0.
*
*
*
*
* OVERFLOW CHECKS IF SCROLL UP IS NEEDED; IF IT IS, IT
*  SCROLLS UP AND MOVES DATA TO & FROM THE BUFFERS.
*
* OVRFLO: IF CSRPTR < E200 THEN RETURN; IF EDIT IS ON THEN
* OVR1:    BUFFLO+ GETS SCNPTR TO 'C.R.';
* DSTADR GETS CSRPTR GETS E1E0 (LAST LINE ON SCREEN);
* IF EDIT IS ON AND BUFFHI < BUFEND THEN MOVE THE TEXT
* FROM BUFFHI TO 'C.R.' TO THE LAST LINE.
*
*
*
*
* UNDERFLOW CHECKS IF SCROLL DOWN IS NEEDED AND MOVES
*  DATA TO AND FROM THE BUFFERS.  CURSER HAD BEEN
*  MOVED OFF THE TOP OF THE SCREEN AND IS NOW PUT AT THE
*  HOME POSITION ON THE SCREEN.
*
* NDRFLO:   IF CSRPTR > DFFF THEN RETURN (GO TO OVRFLO);
*  IF EDIT FLAG IS ON THEN MOVE LAST LINE TO BUFFHI
*   ON DOWN; SCRLDN; CSRPTR GETS E000; MOVE LINE FROM
*   BUFFLO TO FIRST LINE ON THE CRT.
*
*
*
*
*  NOTE:  DON'T SCROLL OFF SCREEN IN EXEC UNTIL AFTER
*    THE EDITOR HAS BEEN RUN.
*
*  NOTE:  EVERY LINE MUST HAVE A C.R. ON IT.
```

```
FC32 CE E000 HOME   LDX    #$E000   LOADS HOME POSITION.
FC35 DF 1C          STX    CSRPTR   STORES HOME IN CURSOR PTR.
FC37 39             RTS             RETURNS TO CALLER.
                    *
                    *
FC38 C6 60   CLEAR  LDA B  #$60     LOADS BLANK (C. R. ).
FC3A CE E200        LDX    #$E200   LOADS END-OF-SCREEN PTR.
FC3D 09      CLEAR1 DEX             DECREMENTS BLANKING PTR.
FC3E E7 00          STA B  0,X      BLANKS LOCATION.
FC40 9C 1C          CPX    CSRPTR   TESTS IF DONE.
FC42 26 F9          BNE.   CLEAR1   BRANCHES BACK IF NOT DONE.
FC44 39             RTS             RETURNS.
                    *
                    *
                    *
                    *       GETCHR INPUTS A CHARACTER INTO ACC A WITHOUT
                    *       MOVING THE CURSOR, AND BLINKS THE CURSOR.
                    *
FC45 DE 1C   GETCHR LDX    CSRPTR   LOADS CRT CURSOR POSITION.
FC47 63 00          COM    0,X      COMPLIMENT (FLASH POSITION).
FC49 CE 26F0        LDX    #9968    LOADS BLINK COUNT VALUE.
FC4C 09      GET1   DEX             COUNT GETS COUNT-1.
FC4D 27 F6          BEQ    GETCHR   RESETS CTR WHEN TIMED OUT.
FC4F 7D F001        TST    $F001    TESTS IF A CHAR TYPED IN.
FC52 2A F8          BPL    GET1     BRANCH IF CHAR NOT ENTERED.
FC54 DE 1C          LDX    CSRPTR   LOADS CURSOR POSITION.
FC56 A6 00          LDA A  0,X      TESTS IF BLINKED (SOLID).
FC58 2A 02          BPL    GET2     SKIPS IF NOT BLINKED.
FC5A 63 00          COM    0,X      CLEARS THE CHARACTER.
FC5C B6 F000 GET2   LDA A  $F000    LOADS A WITH CHAR.
FC5F 39             RTS             RETURNS TO CALLER.
                    *
                    *
                    *
                    *
                    *        EDITOR IS THE MAIN ENTRY POINT FOR EDITING.
                    *
FC60 DE 0C   EDITOR LDX    BUFADR   BUFFLO GETS THE
FC62 DF 20          STX    BUFFLO   VALUE OF BUFADR.
FC64 DE 0E          LDX    BUFEND   BUFFHI GETS THE
FC66 DF 22          STX    BUFFHI   VALUE OF BUFEND
FC68 8D C8   REEDIT BSR    HOME     ENTRY POINT FOR
FC6A 8D CC          BSR    CLEAR    RE-EDITING TEXT.
FC6C 97 32   EDITRD STA A  EDIT     TURNS ON EDIT MODE . A <> ∅
FC6E DE 1C   EDITIN LDX    CSRPTR   SETS SCNPTR TO CSRPTR.
FC70 DF 24          STX    SCNPTR
                    *
                    *
FC72 8D D1   EDREAD BSR    GETCHR   X GETS CSRPTR & A GETS CHR.
FC74 81 1B   ENDCHR CMP A  #$1B     TESTS FOR AN "ESC" CHAR.
FC76 26 04          BNE    ED1      SKIPS IF NOT EDIT END.
FC78 7F 0032        CLR    EDIT     TURNS OFFF EDIT FLAG.
FC7B 39             RTS             EXITS THE EDITOR.
FC7C 8D 02   ED1    BSR    INSERT   EDITS CHARACTER.
FC7E 20 F2          BRA    EDREAD   GOES FOR NEXT CHARACTER.
```

```
                *       FOLLOWING IS THE MAIN EDITOR EXECUTION LOOP.
                *
                *
                *
                *
                *
FC80 81 09      INSERT CMP A  #$09      TESTS FOR A CONTROL 'I'.
FC82 2D 16             BLT    DELETE    SKIPS TO DELETE COMMD.
FC84 2E 02             BGT    CR        SKIPS FOR NEXT TEST.
FC86 20 70             BRA    OVR1      SCROLLS UP ONE LINE.
                *
                *
FC88 81 0D      CR     CMP A  #$0D      TESTS FOR CARRIAGE RETURN.
FC8A 2D A6             BLT    HOME      SKIPS IF HOME CSR COMMD.
FC8C 2E 02             BGT    RTCSR     GOES TO  NEXT COMND TEST.
FC8E 86 60             LDA A  #$60      LOADS INTERNAL C.R.  VALUE.
                *
                *
FC90 81 12      RTCSR  CMP A  #$12      TESTS FOR RIGHT ARROW.
FC92 2D 2A             BLT    SUB32     SKIPS IF AN "UP ARROW".
FC94 2E 08             BGT    LFTCSR    SKIPS IF A "LEFT ARROW".
FC96 DE 1C      RTARRO LDX    CSRPTR    LOADS CURSOR POINTER.
FC98 20 1F             BRA    PUTCH1    STORES & INCREMENTS CSR.
                *
                *
FC9A 8D 3F      DELETE BSR    UNDR2     SCROLLS DOWN 1 LINE.
FC9C 20 6A             BRA    OVR2      CSRPTR GETS E1E0 (LAST LIN).
                *
                *
FC9E 81 14      LFTCSR CMP A  #$14      TESTS IF "LEFT ARROW".
FCA0 2D 26             BLT    ADD32     SKIPS IF "DOWN ARROW".
FCA2 2E 03             BGT    CLER      SKIPS FOR NEXT TEST.
FCA4 09                DEX              SUB.  1 FROM CSRPTR.
FCA5 20 13             BRA    PUTCH2    STORES CURSER POINTER.
                *
FCA7 81 1F      CLER   CMP A  #$1F      TESTS FOR CTRL BACK ARROW.
FCA9 2D 8D             BLT    CLEAR     GOES TO CLEAR SCREEN.
FCAB 27 62             BEQ    LFTJST    MOVES CSR TO LEFT OF SCREEN.
                *
                * ALL OTHER CHARACTERS FALL THRU TO PUTCHR.
                *
```

```
                    *       PUTCHR DISPLAYS A CHARACTER ON THE CRT DISPLAY AND
                    *         INCREMENTS THE CURSER POINTER AS WELL AS CHECKING
                    *         AND HANDLING CARRIAGE RETURNS.
                    *
                    *
                    *
FCAD DE 1C          PUTCHR LDX      CSRPTR     LOADS OLD CSRPTR.
FCAF 81 0D                 CMP A    #$0D       TESTS FOR EXTERNAL C.R.
FCB1 27 04                 BEQ      CRLF1      SKIPS TO DO A C.R. L.F.
FCB3 A7 00                 STA A    0,X        DISPLAYS CHAR ON SCREEN.
FCB5 81 60                 CMP A    #$60       TESTS FOR INTERNAL C.R.
FCB7 27 5D          CRLF1  BEQ      CRLF       SKIPS FOR CR. LF.
FCB9 08             PUTCH1 INX                 INCREMENTS CSRPTR.
FCBA DF 1C          PUTCH2 STX      CSRPTR     SAVES NEW CSRPTR.
FCBC 20 12                 BRA      NDRFLO     TESTS FOR OVRFLO & UNDRFLO.


FCBE DE 1C          SUB32  LDX      CSRPTR     LOADS CURRENT CRSR POSITION.
FCC0 C6 20                 LDA B    #32        LOADS LOOP COUNT.
FCC2 09             SUB32A DEX                 DECRIMENTS CSRPTR.
FCC3 5A                    DEC B    -          DECRIMENTS LOOP COUNTR.
FCC4 26 FC                 BNE      SUB32A     SKIPS BACK IF NOT DONE.
FCC6 20 06                 BRA      ADD2       SKIPS TO CHECK UNDRFLO.

FCC8 C6 20          ADD32  LDA B    #32        LOADS LOOP COUNTER.
FCCA 08             ADD32A INX                 INCRE. CSRPTR IN INDEX.
FCCB 5A                    DEC B    -          DECRIMENTS LOOP COUNTER.
FCCC 26 FC                 BNE      ADD32A     SKIPS BACK IF NOT DONE.
FCCE DF 1C          ADD2   STX      CSRPTR     SAVES CSRPTR.


                    * NDRFLO (UNDERFLOW) CHECKS FOR THE CURSOR GOING OFF THE
                    *   TOP OF THE SCREEN. THE INDEX REG. CONTAINS THE CURSOR
                    *   POINTER WHEN THE ROUTINE IS ENTERED.
                    *
                    *
FCD0 8C E000        NDRFLO CPX      #$E000     TESTS IF CSRPTR >= DFFF.
FCD3 2C 0B                 BGE      OVRFLO     SKIPS IF CSRPTR GREATER.
FCD5 D6 32                 LDA B    EDIT       TESTS IF EDITOR IS ON.
FCD7 27 02                 BEQ      UNDR2      SKIP TO EXIT IF EDITOR OFF.
FCD9 8D 6D                 BSR      MOVE2      MOVES LAST LINE TO BUFFHI.
FCDB BD FD73        UNDR2  JSR      SCRLDN     MOVES ALL LINES DOWN ONE.
FCDE 8D 3A                 BSR      MOVE3      MOVES BUFFLO TO TOP OF CRT.
```

```
                 *      OVERFLOW CHECKS FOR SCROLLING UP (CURSOR IS OFF
                 * THE BOTTON OF THE SCREEN); INDEX CONTAINS THE CURSOR
                 * POINTER UPON ENTRY.
                 *
                 *
FCE0 8C E200 OVRFLO CPX     #$E200    TESTS AND EXITS IF
FCE3 2B 12          BMI     OVREXT    CURSOR ON SCREEN.
FCE5 8D 11          BSR     OVR1      DOES OVR1 CHECKING.
FCE7 D6 32          LDA B   EDIT      TESTS IF EDIT IS ON.
FCE9 27 0C          BEQ     OVREXT    EXITS IF IT IS OFF.
FCEB DE 22          LDX     BUFFHI    LOADS HI TEXT POINTR.
FCED 9C 0E          CPX     BUFEND    TESTS IF PTRS NOT EQU.
FCEF 27 06          BEQ     OVREXT    EXITS IF NO TEXT.
FCF1 8D 44          BSR     MOVE1A    MOVES CHRS TO LAST LINE.
FCF3 DE 14          LDX     SRCADR    RESETS NEW BUFFHI
FCF5 DF 22          STX     BUFFHI    LOCATION.
FCF7 39      OVREXT RTS               EXITS BACK TO EDIT.
                 *
                 *
                 *
                 *      OVR1 DOES ACTUAL SCROLLING UP.
                 *
FCF8 D6 32   OVR1   LDA B   EDIT      TESTS IF EDIT IS ON.
FCFA 27 0A          BEQ     OVR1A     SKIPS IF EDIT OFF.
FCFC DE 20          LDX     BUFFLO    LOADS TEXT PTR LOW.
FCFE DF 16          STX     DSTADR    DESTINATION OF TEXT MOVE.
FD00 DE 24          LDX     SCNPTR    SOURCE FOR MOVE.
FD02 8D 34          BSR     MOVE1     MOVES LIN1 TO BUFFFLO.
FD04 DF 20          STX     BUFFLO    SAVES NEW BUFFLO PTR.
FD06 8D 58   OVR1A  BSR     SCRLUP    SCROLLS SCREEN UP 1.
FD08 CE E1E0 OVR2   LDX     #$E1E0    CURSOR GETS PUT ON
FD0B DF 1C          STX     CSRPTR    LAST LINE.
FD0D DF 16          STX     DSTADR    INIT DEST FOR NEXT MOVE.


                 *      FOLLOWING MOVE THE CURSOR.
                 *
FD0F D6 1D   LFTJST LDA B   CSRPTR+1  LOADS LO BYTE OF PTR.
FD11 C4 E0          AND B   #$E0      TRUNCATES TO LEFT OF LINE.
FD13 D7 1D          STA B   CSRPTR+1  SAVES L. J. ED PTR.
FD15 39             RTS               RETURNS TO EDITOR.
                 *
                 *
FD16 8D B0   CRLF   BSR     ADD32     LINE FEED.
FD18 20 F5          BRA     LFTJST    CARRIAGE RETURN.
```

```
                    *      MOVE INSTRUCTIONS MOVE FROM ONE BUFFER AREA TO
                    * ANOTHER BUFFER AREA.
                    *
                    *
                    *
                    *      MOVE3 CALCULATES THE SOURCE ADDRESS OF THE DATA IN
                    * BUFFLO (IF IT EXISTS) FOR MOVING TO THE FIRST LINE ON
                    * THE CRT.  MOVE 1 IS THEN ENTERED TO DO THE MOVING.
                    *
                    *
      FD1A DE 24    MOVE3   LDX     SCNPTR    CSRPTR GETS E000 (HOME).
      FD1C DF 16            STX     DSTADR    SETS MOVE ADDRESS.
      FD1E DE 20            LDX     BUFFLO    LOADS LO BUFFR ADDR.
      FD20 9C 0C            CPX     BUFADR    TESTS IF DATA STRING.
      FD22 27 23            BEQ     MOVEXT    EXITS IF EMPTY.
      FD24 09               DEX               MOVES BACK FROM BLANK.
      FD25 9C 0C    MV31    CPX     BUFADR    TESTS IF SRCADR = BUFFADR.
      FD27 27 08            BEQ     MV32      MOVES IF START OF LINE.
      FD29 09               DEX               NEXT LOWER CHAR.
      FD2A E6 00            LDA B   0,X       GETS SOURCE CHAR FOR TEST.
      FD2C C1 60            CMP B   #$60      TESTS FOR "C.R.".
      FD2E 26 F5            BNE     MV31      SKIPS BACK UNTIL "C.R.".
      FD30 08               INX               POINTS BACK TO FIRST CHAR.
      FD31 DF 20    MV32    STX     BUFFLO    SAVES LO ADDRESS.
      FD33 20 03            BRA     MOVE1     MOVES DATA.


                    *          MOVE1
                    *
                    *      MOVE1 MOVES A SET OF CHARACTERS FROM EITHER THE TOP
                    * LINE OF THE SCREEN TO BUFFLO OR FROM BUFFHI TO THE
                    * BOTTOM LINE OF THE SCREEN.  THE SOURCE ADDRESS IS PASSED
                    * IN THE INDEX REG., THE DESTINATION ADDRESS IN DSTADR,
                    * AND THE MOVE IS TERMINATED BY A "C.R." IN THE LINE OF
                    * TEXT BEING MOVED.
                    *
                    *
                    *
                    *
      FD35 DE 14    MOVE    LDX     SRCADR    LOADS SOURCE ADDRESS INTO X.E-
      FD37 08       MOVE1A  INX               POINTS TO NEXT SOURCE CHAR.
      FD38 E6 00    MOVE1   LDA B   0,X       LOADS SOURCE CHARACTER.
      FD3A DF 14            STX     SRCADR    SAVES THE SOURCE POINTER.
      FD3C DE 16            LDX     DSTADR    LOADS DESTINATION ADDRESS.
      FD3E E7 00            STA B   0,X       STORES CHAR. IN DESTINATION.
      FD40 08               INX               NEXT DESTINATION ADDRESS.
      FD41 DF 16            STX     DSTADR    SAVES DESTINATION PTR.
      FD43 C1 60            CMP B   #$60      TESTS IF MOVE FINSHED (CR).
      FD45 26 EE            BNE     MOVE      SKIPS BACK IF NOT DONE.
      FD47 39       MOVEXT  RTS               RETURNS TO CALLER.
```

```
                   *          MOVE2 SUBROUTINE
                   *
                   *     THE MOVE2 SUBROUTINE MOVES THE LAST LINE ON THE
                   * SCREEN TO THE HIGH AREA OF THE BUFFER (BUFFHI) DURING
                   * SCROLLING.  THE TEXT IS TEMPORARILY STORED ON THE STACK
                   * DURING THE MOVE.  THE MOVE IS TERMIINATED BY A "C.R.".
                   * THE TEXT IS STORED AT BUFFHI ON DOWN.
                   *
                   *
                   *
                   *
FD48 8D BE         MOVE2  BSR    OVR2      X GETS ADDR OF LAST LINE.
FD4A 5F                   CLR B  -         SETS TERMINATION FOR
FD4B 37            MV21   PSH B  STACK     POPPING.
FD4C E6 00                LDA B  0,X       LOADS SOURCE CHAR.
FD4E 08                   INX              POINTS TO NEXT CHAR.
FD4F C1 60                CMP B  #$60      TESTS IF LINE TO "C.R."
FD51 26 F8                BNE    MV21      MOVED TO STACK.
FD53 DE 22         MV22   LDX    BUFFHI    INIT. DESTINATION.
FD55 E7 00         MV23   STA B  0,X       STORES CHAR.
FD57 09                   DEX              POINTS TO NEXT LOCATION.
FD58 DF 22                STX    BUFFHI    UPDATES BUFFER PTR.
FD5A 33                   PUL B  -         GETS NEXT CHAR.
FD5B C1 00                CMP B  #00       TESTS IF ALL CHRS STORED.
FD5D 26 F6                BNE    MV23      SKIPS BACK IF NOT STORED.
FD5F 39            MOVEX  RTS              RETURNS TO CALLER.


                   *      SCROLLUP MOVES ALL LINES UP 1, & CLEARS LAST LINE.
                   *
FD60 CE E000       SCRLUP LDX    #$E000    SETS CRT HOME POSITION.
FD63 E6 20         SCRP1  LDA B  $20,X     GETS CHAR FROM NEXT LINE.
FD65 E7 00                STA B  00,X      STORES CHAR ON PREV. LINE.
FD67 08                   INX              POINTS TO NEXT LINE.
FD68 8C E1E0              CPX    #$E1E0    TESTS IF MOVE DONE.
FD6B 26 F6                BNE    SCRP1     GOES BACK IF NOT DONE.
FD6D DF 1C                STX    CSRPTR    SETS CSRPTR TO LAST LINE.
FD6F BD FC38              JSR    CLEAR     CLEARS LAST LINE.
FD72 39                   RTS              EXITS.
```

```
                          *      SCRLDOWN MOVES ALL LINES DOWN ONE AND
                          * CLEARS THE TOP LINE ON THE SCREEN.
                          *
FD73 CE E1DF   SCRLDN LDX       #$E1DF   INITIALIZES THE POINTER.
FD76 E6 00     SCRD1  LDA B     0,X      LOADS DATA TO BE MOVED.
FD78 E7 20            STA B     $20,X    MOVES DATA DOWN ONE LINE.
FD7A DF 1C            STX       CSRPTR   SAVES CURSOR.
FD7C 09               DEX                POINTS TO NEXT BYTE.
FD7D 8C DFFF          CPX       #$DFFF   TESTS IF MOVE FINISHED.
FD80 26 F4            BNE       SCRD1    SKIPS BACK IF NOT DONE.
FD82 C6 60            LDA B     #$60     LOADS BLANK TO CLEAR LINE.
FD84 08        SCRD2  INX                POINTS TO NEXT CHARACTER.
FD85 E7 00            STA B     0,X      CLEARS BYTE ON LINE 1.
FD87 8C E01F          CPX       #$E01F   TESTS IF LINE 1 CLEARED.
FD8A 26 F8            BNE       SCRD2    SKIPS BACK IF NOT CLEARED.
FD8C 39               RTS                RETURNS.



                          *      OUTSTRING PRINTS OUT THE STRING BETWEEN THE
                          *      OUTBUF POINTER AND THE BUFEND POINTER.
                          *
FD8D DE 11     OUTSTR LDX       OUTBUF   BUFPTR GETS START OF TEXT.
FD8F A6 00     OUT1   LDA A     0,X      LOADS CHAR TO BE PUT OUT.
FD91 DF 1E            STX       BUFPTR   SAVES SOURCE POINTER.
FD93 BD FCAD          JSR       PUTCHR   PRINTS CHARACTER.
FD96 DE 1E            LDX       BUFPTR   RESTORES POINTER.
FD98 9C 0A            CPX       OUTEND   TESTS FOR END-OF-TEXT.
FD9A 27 03            BEQ       OUT2     EXITS IF END OF TEXT.
FD9C 08               INX                INCRE. PTR TO NEXT CHAR.
FD9D 20 F0            BRA       OUT1     GOES BACK FOR NEXT CHAR.
FD9F 39        OUT2   RTS                EXITS ROUTINE.
FDA0 01               NOP                UNUSED LOCATION.
                          *
                          *      END OF EDITOR PROGRAM.
                          *
```

```
*               THE MINI-ASSEMBLER
*
*
*      THE MINI-ASSEMBLER IS A FIXED-FIELD ONE INSTRUCTION
* PER LINE 2 PASS ASSEMBLER.  THE MINI-ASSSEMBLER FORMAT
* IS DESCRIBED ON PAGES 9-2 AND 9-3 OF THE SPHERE
* OPERATORS REFERENCE MANUAL.
*      THE TWO PASSES ARE REQUIRED TO FORM THE LABEL
* ADDRESSES.  THE SECOND PASS EQUATES THE ADDRESS FOR
* LABLES REFRENCED BEFORE THEY ARE DEFINED IN THE PROGRAM.
*
*
* ON ENTRY:
* SRCASM = ADDRESS OF SOURCE TEXT TO BE ASSEMBLED.
* BUFFLO = ADDRESSED OF OBJECT CODE PRODUCED.
*
* ON EXIT:
* PCVAL (PROGRAM COUNTER VALUE) = LAST LOCATION OF
* THE ASSEMBLED OBJECT PROGRAM.
*
*
*
*
* ALGORITHM:
*
*ASMBLR:  SET PASS COUNT TO ZERO; SET PCVAL TO DSTASM;
*ASM1A:  OPERAND VALUE FORMED IN "ONDVAL":
* A GETS CHAR IN X6 (OPERAND TYPE); X GETS X+7;
* IF CHAR X6 IS A "@" THEN ONDVAL GETS VALUE FROM SYMBOL
*   TABLE ELSE ONDVAL GETS VALUE FROM ASCBIN CONVERSION;
*SYMBL:  EQUATES SYMBOL (PC VALUE IS THE " " SYMBOL
*[LABEL]) TO A LABEL VALUE:
* SYMVAL GETS PCVAL;
* IF X(1) IS AN "=" THEN SYMVAL GETS ONDVAL;
* IF X(1) IS NOT A "=" OR A SPACE THEN IF SECOND PASS THEN
*   EXIT ELSE START SECOND PASS;
* LABEL ENTRY IN SYMBOL TABLE GETS SYMVAL;
*LDOP:  PUT OPERATION CODE INTO THE OBJECT CODE:
* CONVERT X(2)-X(3) INTO BINARY;
* SAVE PCVAL;
* P.C.  GETS P.C.+1;
*OPRND:  FORM OPERAND IN OBJECT CODE:
* FORM ONDVAL INTO PROPER SIZE BASED ON CODE IN X(6);
* STORE NEW OPERAND VALUE IN MEMORY;
* P.C.  GETS P.C.+1 OR 2;
* GET NEXT LINE OF SOURCE;
* GO TO ASM1A;
*
*
```

```
FDA1 7F 0004 ASMBLR CLR     AR3       INIT. PASS CTR TO FRST PASS. )
FDA4 DE 20   ASM1   LDX     BUFFLO    SETS PC CNTR TO START OF
FDA6 DF 40          STX     PCVAL     OBJECT CODE.
FDA8 DE 26          LDX     SRCASM    LOADS ADDR FOR FIRST LINE.
FDAA DF 02   ASM1A  STX     TMP1      SAVES ADDR OF CURRENT LINE.
FDAC A6 08          LDA A   8,X       LOADS SYMBOL (LABEL).
FDAE E6 07          LDA B   7,X       LOADS OPERAND TYPE CODE.
FDB0 C1 40          CMP B   #$'@      IF @, LOADS DATA IN SYMBOL
FDB2 27 6B          BEQ     INDADR    ADDRESS, GOES TO SYMBL.
FDB4 08             INX               SETS INDEX TO START OF
FDB5 08             INX               OPERAND NUMBER.
FDB6 08             INX
FDB7 08             INX
FDB8 08             INX
FDB9 08             INX
FDBA 08             INX
FDBB BD FF22        JSR     ASCBIN    CONVRTS # TO BINARY IN B-A.
FDBE D7 2A   ASM1B  STA B   ONDVAL    STORES OPERAND VALUE IN
FDC0 97 2B          STA A   ONDVAL+1  ONDVAL.
             *
             *
             *
             * FOLLOWING FORMS THE VALUE FOR THE LABEL.
             *
FDC2 DE 02   SYMBL  LDX     TMP1      LOADS ORIG LINE PTR INTO X.
FDC4 A6 00          LDA A   0,X       LOADS SYMBOL (LABEL).
FDC6 E6 01          LDA B   1,X       LOADS LABEL CONTROL CHAR.
FDC8 DE 40          LDX     PCVAL     LABEL VALUE GETS PCVAL.
FDCA DF 2C          STX     SYMVAL
FDCC C1 3D          CMP B   #$'=      TESTS IF LABL IS EQUATED.
FDCE 26 06          BNE     ASM2      SKIPS IF NOT EQUATED.
FDD0 DE 2A          LDX     ONDVAL    LABEL VALUE (SYMVAL) GETS
FDD2 DF 2C          STX     SYMVAL    THE OPERAND VALUE.
FDD4 20 0E          BRA     ASM3      CONTINUES EVALUATION.
FDD6 C1 20   ASM2   CMP B   #$'       TESTS FOR END-OF-PROGRAM.
FDD8 27 0A          BEQ     ASM3      SKIPS IF SPACE (NOT END).
FDDA 7D 0004        TST     AR3       TESTS IF SECOND PASS.
FDDD 27 01          BEQ     ASM2A     EXITS IF SECOND PASS.
FDDF 39             RTS
FDE0 D7 04   ASM2A  STA B   AR3       SETS CTR TO SECOND PASS.
FDE2 20 C0          BRA     ASM1      GOES BACK FOR SECOND PASS.
             *
             *
             *
             * FOLLOWING PUTS THE LABLE VALUE IN THE SYMBOL TABLE.
             *
FDE4 8D 41   ASM3   BSR     SYMPTR    X GETS SYMBL TABL ENTRY ADR. E
FDE6 96 2C          LDA A   SYMVAL    STORES THE LABEL
FDE8 A7 00          STA A   0,X       ADDRESS (SYMVAL) INTO THE
FDEA 96 2D          LDA A   SYMVAL+1  SYMBOL TABEL.
FDEC A7 01          STA A   1,X
```

```
                    * FOLLOWING FORMS THE OPERATION CODE.
                    *
FDEE DE 02          LDOP   LDX    TMP1     LOADS ORIG LINE POINTER.
FDF0 08                    INX             SETS X TO POINT TO
FDF1 08                    INX             THE OP CODE CHARS.
FDF2 08                    INX
FDF3 A6 00                 LDA A  0,X      GETS OP CODE CHAR INTO A.
FDF5 81 20                 CMP A  #$'      TESTS IF OP CODE EXISTS.
FDF7 27 0A                 BEQ    OPRND    SKIPS IF NONEXISTANT.
FDF9 BD FF22               JSR    ASCBIN   CONVRTS OP CODE TO BINARY.
FDFC DE 40                 LDX    PCVAL    LOADS POINTR TO OBJECT CODE.
FDFE A7 00                 STA A  0,X      STORS OP INTO OBJECT CODE.
FE00 08                    INX             SETS TO NEXT OBJ CODE LOCTN.
FE01 DF 40                 STX    PCVAL    SAVES P.C. POINTER.
                    *
                    *
                    *
                    * FOLLOWING STORES INTO THE OBJECT CODE THE SIZED OPERAND.
                    *
FE03 DE 02          OPRND  LDX    TMP1     LOADS SOURCE LINE POINTER.
FE05 A6 06                 LDA A  6,X      LOADS OPERAND SIZE CHAR.
FE07 DE 40                 LDX    PCVAL    LOADS X WITH OBJ CODE PTR.
FE09 81 45                 CMP A  #$'E     TESTS LENGTH TYPE.
FE0B 2E 31                 BGT    RELTIV   SKIPS IF AN "R" OPERAND.
FE0D 27 21                 BEQ    EXTEND   SKIPS IF AN "E" SIZE OPRND.
FE0F 81 44                 CMP A  #$'D     TESTS IF SIZE CHR EXISTS.
FE11 27 22                 BEQ    DIRECT   SKIPS IF "D"COMND EXISTS.
                    *
                    *
                    *
                    * FOLLOWING GETS THE NEXT LINE.
                    *
FE13 DE 02          ASM4   LDX    TMP1     LOADS START OF LINE IN
FE15 08             ASM4A  INX             ORDER TO FIND NEXT LINE.
FE16 A6 00                 LDA A  0,X      LOADS CHAR FROM SORCE LINE.
FE18 81 60                 CMP A  #$60     TESTS FOR A CARRAGE RETURN.
FE1A 26 F9                 BNE    ASM4A    SKIPS BAK UNTIL C.R. FOUND.
FE1C 08                    INX             POINTS TO FIRST LINE CHAR.
FE1D 20 8B                 BRA    ASM1A    GOES BACK TO ASSM. NEXT LINE
                    *
                    *
                    *
                    * THE FOLLOWING ARE SUBROUTINES USED BY THE MAIN CODE.
                    *
                    *
FE1F 8D 06          INDADR BSR    SYMPTR   GETS CONTENTS OF
FE21 EE 00                 LDX    0,X      SYMBOL LOCATION.
FE23 DF 2A                 STX    ONDVAL   STORES AS OPERAND.
FE25 20 9B                 BRA    SYMBL    RETURNS TO FIX LABEL VALUE.
                    *
FE27 48             SYMPTR ASL A  -        MULT LABEL BY 2 TO FORM
FE28 5F                    CLR B  -        POINTR INTO SYMBOL TABLE.
FE29 97 01          LOADX  STA A  TMP+1    LOADS POINTER INTO THE
FE2B D7 00                 STA B  TMP      SYMBOL TABLE INTO X.
FE2D DE 00                 LDX    TMP      RETURNS TO CALLER.
FE2F 39                    RTS             RETURNS.
0
```

```
00690 FE30 D6 2A    EXTEND LDA B  ONDVAL    STORES HI BYTE OF OPERAND
00691 FE32 E7 00           STA B  0,X       INTO OBJECT CODE.
00692 FE34 08              INX              INC PC TO POINT TO NXT WD.
00693 FE35 96 2B    DIRECT LDA A  ONDVAL+1  STORES LO BYTE OF OPERAND
00694 FE37 A7 00           STA A  0,X       INTO OBJECT CODE.
00695 FE39 08              INX              INC & SAVE P.C. TO POINT TO
00696 FE3A DF 40           STX    PCVAL     NEXT BYTE.
00697 FE3C 20 D5           BRA    ASM4      GOES TO WORK ON NEXT LINE.
00698               *
00699 FE3E 08       RELTIV INX              INCRIMENT P.C. PTR TO POINT
00700 FE3F DF 40           STX    PCVAL     TO NXT BYTE & SAVE P.C..
00701 FE41 96 2B           LDA A  ONDVAL+1  LOADS LO BYTE OF OPERAND.
00702 FE43 90 41           SUB A  PCVAL+1   FORMS RELATIVE OFFSET.
00703 FE45 09              DEX              INSERTS RELATIVE BYTE INTO
00704 FE46 A7 00           STA A  0,X       OBJECT CODE.
00705 FE48 20 C9           BRA    ASM4      GOES TO ASSMBL NEXT LINE.
00706               *
00707               *      END OF THE ASSEMBLER PROGRAM.
00708               *
```

```
*             DEBUGGER
*
*      THE DEBUGGER FOR THE PDS SYSTEM WAS DESIGNED TO
* PROVIDE A VERSATILE TOOL FOR USE IN PROGRAM TESTING AND
* DEBUGGING.  IT ALLOWS FOR BREAKPOINTS, MINI-ASSEMBLER
* SYMBOL TABLE REFERENCING, STACK MANIPULATION AND
* INPUT IN EITHER HEXADECIMAL, OCTAL OR DECIMAL.
* THE DEBUGGER PRINTS A PROMT CHARACTER ">" ON EVERY NEW
* LINE.  AN INSTRUCTION CAN BE TYPED IN WHENEVER THE CURRSOR
* IS BLINKING, EXCEPT WHEN A NUMBER IS BEING TYPED IN.
* THE DEBUGGER CALLS THE EDITOR WHENEVER A NUMBER IS TO BE
* INPUT, SO CORRECTIONS CAN BE MADE IF THE WRONG DIGIT IS
* TYPED IN.  THE POINTER "PCVAL" POINTS TO THE CURRENTLY
* OPENED BYTE LOCATION.  THE DEBUGGER OPERATES ON WHATEVER
* BYTE IS POINTED TO BY PCVAL.  FOR FURTHER DETAILS SEE THE
* SECTION ON THE DEBUGGER IN THE OPERATORS REFERENCE MANUAL.
*
*      THE DEBUGGER IS IMPLEMENTED BY A SMALL ROUTINE TO SET
* UP ENTRY (DEBUG) AND A LARGE ROUTINE WHICH DOES A RANGE
* COMPARE TO FIND THE PROPER COMMAND AND THEN EXECUTES THE
* COMMAND (RUNBUG).  NOTE THAT SINCE COMMANDS ARE
* INTERPRETED BY RANGE, ANY KEY STRUCK WILL PRODUCE
* A COMMAND EXECUTION, SUCH AS A "," BEING INTERPRETED
* AS A "+" COMMAND.
*
*
*
* COMMANDS:
*
* "C.R." LINE - PRINTS ">" OUT ON A NEW LINE.
* " " CHANGE - THE SPACE COMND CHANGES CONTENTS FROM Y TO Z.
* "+" OPNNXT - OPENS NEXT LOCATION.
* "-" OPNPRE - OPENS PREVIOUS LOCATION.
* "B" BRKSET - SETS A BREAKPOINT AT THE OPENED LOCATION.
* "C" CLRBRK - CLEARS BRKPOINT. MUST BE DONE BEFORE EXIT.
* "E" EXIT - PERFORM RTI - EXECUTE AT BRKPOINT LOCATION.
* "G" GOLOCN - STARTS EXECUTION AT OPENED LOCATION.
* "O" OPNLOC - OPENS LOCATION THAT IS TYPED IN AFTER "O".
* "R" OPNREG - OPENS THE TOP-OF-STACK LOCATION.
* "S" SETSTK - SETS THE STACK TO THE OPENED LOCATION.
* "T" OPNTBL - OPENS LOCATION IN SYMBOL TABLE OF NEXT CHR.
*
*
*
*
*
* SUBROUTINES:
*
* INPCHR - INPUTS A CHAR. INTO A AND PRINTS IT.
* INPNUM - INPUTS A NUMBER INTO B-A FROM THE KEYBOARD.
* PNTBYT - PRINTS ACC A AS 2 HEX DIGITS ON THE SCREEN.
* PNTDIG - PRINTS B-A AS 4 HEX DIGITS ON THE SCREEN.
* NEWLIN - PRINTS A C.R. AND A ">" ON THE SCREEN.
* DSPADR - PRINTS BYTE ADDRESS (XXXX) AND BYTE CONTENTS (YY)
*   AS >XXXX YY ON THE SCREEN.
*
```

0

```
FE4A                ORG    $FE4A
                    *
FE4A BD FCBE BGPOP1 JSR    SUB32    MOVES CURSOR UP ONE LINE.
FE4D 31      POPLIN INS             CLEANS UP STACK FOR
FE4E 31             INS             DISPLAY OF C.R. >.
                    *
                    *
FE4F 8D 74   DEBUG  BSR    NEWLIN   PRINTS "C.R.>".
FE51 8D 09   DBUG1  BSR    INPCHR   READS IN COMMAND.
FE53 BD FC96        JSR    RTARRO   INSERT BLANK.
FE56 DE 40          LDX    PCVAL    LOADS CURRENTLY OPENED LOC.
FE58 8D 09          BSR    RUNBUG   EXECUTES DEBUG COMMAND.
FE5A 20 F5          BRA    DBUG1    GOES BACK FOR NEXT COMND.
                    *
FE5C BD FC45 INPCHR JSR    GETCHR   READ IN CHAR INTO A.
FE5F BD FCAD        JSR    PUTCHR   DISPLAYS CHARACTER.
FE62 39             RTS             RETURNS TO CALLER.
                    *
                    *
                    *
                    *
FE63 81 20   RUNBUG CMP A  #$'      TESTS FOR SPACE.
FE65 2D E3          BLT    BGPOP1   SKIPS IF A "C.R." (LINE).
FE67 2E 09          BGT    OPNPRE   SKIPS FOR OTHER CMND TESTS.
FE69 BD FEF4 CHANGE JSR    INPNUM   INPUTS NEW BYTE CONTENTS.
FE6C DE 40          LDX    PCVAL    LOADS OPENED BYTE LOCATION.
FE6E A7 00          STA A  0,X      STORES NEW BYTE DATA.
FE70 20 DB          BRA    POPLIN   GOES TO INPUT NEXT COMMAND.
                    *
                    *
FE72 81 2D   OPNPRE CMP A  #$'-     TESTS FOR A "-" COMMAND.
FE74 2D 05          BLT    OPNNXT   SKIPS IF A "+" COMMAND.
FE76 2E 06          BGT    CLRBRK   SKIPS TO TEST FOR OTHER CMND
FE78 09             DEX             FORMS PREV. LOCATION ADDR.
FE79 20 2B          BRA    DSPADR   GOES TO OPEN THE LOCATION.
                    *
                    *
FE7B 08      OPNNXT INX             FORMS NEXT LOCATION ADDR.
FE7C 20 28          BRA    DSPADR   GOES TO OPEN LOCATN BYTE.
                    *
                    *
FE7E 81 43   CLRBRK CMP A  #$'C     TESTS FOR A "C" COMMAND.
FE80 2D 0A          BLT    BRKSET   SKIPS IF A "B" COMMAND.
FE82 2E 78          BGT    EXIT     SKIPS FOR NEXT COMND TEST.
FE84 DE 30          LDX    BRKADR   GETS ADDRESS OF BREAKPOINT.
FE86 96 2E          LDA A  BRKSAV   LOADS ORIG BYTE CONTENTS.
FE88 A7 00          STA A  0,X      RESTORES BYTE DATA.
FE8A 20 1A          BRA    DSPADR   GOES TO OPEN THE LOCATION.
                    *
                    *
FE8C A6 00   BRKSET LDA A  0,X      LOADS DATA OF OPNED LOCATN.
FE8E 97 2E          STA A  BRKSAV   SAVES DATA OF OPNED BYTE.
FE90 DF 30          STX    BRKADR   SAVES ADDR. OF BREAKPOINT.
FE92 86 3F          LDA A  #$3F     LOADS SOFTWARE INTUP COMND.
FE94 A7 00          STA A  0,X      SETS AN SWI AT OPNED BYTE.
FE96 20 B5          BRA    POPLIN   GOES TO NEXT LINE FOR COMND.
```

```
FE98 30        OPNREG TSX                OPENS TOP-OF-STACK.
FE99 08               INX                PCVAL GETS STACK POINTER.
FE9A 08               INX                (CLEANS UP THE STACK).
FE9B 20 09            BRA    DSPADR      GOES TO DISPLAY THE T-O-S.
               *
               *
FE9D 81 53     SETSTK CMP A  #$'S        TESTS FOR AN "S" COMMAND.
FE9F 2D F7            BLT    OPNREG      SKIPS IF AN "R" COMMAND.
FEA1 2E 13            BGT    OPNTBL      SKIPS FOR THE "T" COMMAND.
FEA3 35               TXS                STACK POINTER GETS PCVAL.
FEA4 20 A9            BRA    DEBUG       RETURNS TO INPUT COMMAND.
               *
               *
               *
               *
               * FOLLOWING ARE SUBROUTINES USED BY DEBUGGER.
               *
               *
               * FOLLOWING DISPLAYS THE LOCATION ADDR. & CONTENTS.
               *
FEA6 DF 40     DSPADR STX    PCVAL       SAVES OPENED LOCATION ADDR.
FEA8 8D 1B     DSPAD1 BSR    NEWLIN      PRINTS A "C.R." AND ">".
FEAA 8D 10            BSR    PNTDIG      PRINTS OUT "PCVAL" IN HEX.
FEAC BD FC96          JSR    RTARRO      PRINTS A SPACE.
FEAF DE 40            LDX    PCVAL       LOADS PTR. TO OPEND LOC.
FEB1 A6 00            LDA A  0,X         LOADS DATA FROM LOCATN.
FEB3 8D 17            BSR    PNTBYT      PRINTS DATA IN HEX FORMAT.
FEB5 39               RTS                RETURNS TO INPUT COMMAND.
               *
               *
FEB6 8D A4     OPNTBL BSR    INPCHR      LOADS A WITH SYMBOL (LABL).
FEB8 48               ASL A  -           ALIGNS ADDRESS FOR
FEB9 5F               CLR B  -           SYMBOL TABLE ENTRY.
FEBA 20 5C            BRA    OPNLC1      SAVES AND DISPLAYS ADDRESS.
               *
               *
FEBC 96 40     PNTDIG LDA A  PCVAL       PRINTS THE TWO HI HEX DIGITS
FEBE 8D 0C            BSR    PNTBYT      OF OPENED ADDRESS.
FEC0 96 41            LDA A  PCVAL+1     PRINTS OUT 2 LOW HEX DIGITS
FEC2 8D 08            BSR    PNTBYT      OF THE OPENED ADDRESS.
FEC4 39               RTS                RETURNS TO CALLING CODE.
               *
               *
FEC5 CE 0D3E   NEWLIN LDX    #$0D3E      LOADS "C.R." & ">".
FEC8 DF 35            STX    IOBUFF      STORES FOR PRINTING.
FECA 20 1A            BRA    PNTBUF      SKIPS TO PRINT CHAR.
               *
               *
               * FOLLOWING PRINTS OUT 2 HEX DIGITS.
               *
FECC CE 0010   PNTBYT LDX    #16         LOADS 16 FOR BASE.
FECF DF 04            STX    ARB         STORES FOR CONVERSION.
FED1 5F               CLR B  -           CLEARS HI 2 DIGITS.
FED2 CE 0035          LDX    #IOBUFF     LOADS OUTPUT BUFF ADDRESS.
```

```
                        * FOLLOWING CONVERTS BYTE TO HEX WITH LEADING ZEROS.
                        *
FED5 D7 36      CONVRT STA B   IOBUFF+1 SETS BYTE FOR SECOND DIGIT.
FED7 BD FF64           JSR     BINASC   CONVERTS TO ASCII DIGITS.
FEDA 96 36             LDA A   IOBUFF+1 TESTS BOTH DIGITS CONVTD.
FEDC 26 08             BNE     PNTBUF   SKIPS IF BOTH DIGITS NOT 0.
FEDE 96 35             LDA A   IOBUFF   MOVES LOW DIGIT.
FEE0 97 36             STA A   IOBUFF+1
FEE2 86 30             LDA A   #$'0     HIGH DIGIT GETS A "0".
FEE4 97 35             STA A   IOBUFFF
                        *
FEE6 CE 0035    PNTBUF LDX     #IOBUFF  LOADS ADDR OF HEX DIGITS.
FEE9 DF 11             STX     OUTBUF   STORES AS OUTPUT BUFFER.
FEEB CE 0036           LDX     #IOBUFF+1  STORES ADDR. OF
FEEE DF 0A             STX     OUTEND   END OF DIGITS PRINTED.
FEF0 BD FD8D           JSR     OUTSTR   PRINTS OUT BYTES.
FEF3 39                RTS              RETURNS TO CALLING PROGRAM.
                        *
                        *
                        *
                        * FOLLOWING INPUTS A 16 BIT NUMBER INTO THE BA REG.
                        *
FEF4 BD FC6E    INPNUM JSR     EDITIN   INPUTS A STRING OF DIGITS.
FEF7 DE 24             LDX     SCNPTR   LOADS ADDR. OF FIRST DIGIT.
FEF9 8D 27             BSR     ASCBIN   CONVERTS TO BINARY IN BA.
FEFB 39                RTS              RETURNS TO CALLER.
                        *
                        *
FEFC 81 47      EXIT   CMP A   #$'G     TESTS IF AN EXIT (E) COMMD.
FEFE 27 1E             BEQ     GOLOCN   SKIPS TO THE "GO" COMMAND.
FF00 2C 10             BGE     OPNLOC   SKIPS FOR NEXT COMMD TEST.
FF02 31                INS              CLEARS UP THE STACK.
FF03 31                INS
FF04 30                TSX              INDEX GETS STACK POINTER.
FF05 E6 05             LDA B   5,X      LOADS RETURN ADDRESS.
FF07 A6 06             LDA A   6,X
FF09 80 01             SUB A   #1       SUB 1 FROM RETURN ADDR.
FF0B C2 00             SBC B   #0
FF0D E7 05             STA B   5,X      RESTORES ADDR. TO BRKPT
FF0F A7 06             STA A   6,X      LOCATION.
FF11 3B                RTI              RETURNS FROM BREAKPOINT.
                        *
                        *
FF12 81 4F      OPNLOC CMP A   #'O      TESTS FOR AN "O" COMMAND.
FF14 2E 87             BGT     SETSTK   SKIPS IF AN "R", "S" ETC.
FF16 8D DC             BSR     INPNUM   LOADS A 16 BIT NUMBER.
FF18 D7 40      OPNLC1 STA B   PCVAL    STORES NEW OPENED
FF1A 97 41             STA A   PCVAL+1  LOCATION ADDRESS.
FF1C 20 8A             BRA     DSPAD1   DISPLAYS CONTENTS OF LOC.
                        *
                        *
FF1E 31         GOLOCN INS              CLEANS UP THE STACK.
FF1F 31                INS
FF20 6E 00             JMP     0,X      JUMPS TO USERS PROGRAM.
                        *
                        *      END OF DEBUGGER
                        *
0
```

```
*       UTILITY PROGRAMS
*
*
*
*
*            ASCII TO BINARY CONVERSION.
*
*      THE ASCII TO BINARY ROUTINE CONVERTS FROM AN ASCII
* NUMBER STRING POINTED TO BY X TO AN UNSIGNED 16 BIT
* BINARY NUMBER IN BA (ACC B HAS THE HI BYTE, ACC A HAS
* THE LO BYTE).  THE ASCII STRING IS TERMINATED BY A NON
* HEXADECIMAL CHARACTER.  UPON EXITING, THE INDEX REGISTER
* WILL POINT TO THE NEXT CHARACTER AFTER THE NUMBER
* STRING.  THE BASE OF THE NUMBER STRING IS PASSED TO
* THE ROUTINE IN ARA (ARA IS THE ARITHMETIC REGESTER A
* LOCATED IN BYTES 06 AND 07 OF LOW MEMORY).  IF THE
* ROUTINE IS ENTERED WITH A KNOWN BASE, PUT THE BASE
* (BETWEEN 2 AND 16) IN ARA AND ENTER THE ROUTINE AT
* THE ENTRY POINT ENTR2.
*
*
*
*      CONVERSION FORMULA:
* ASCII NUMBBER STRING X[4],X[3],X[2],X[1] IN
*BASE Y;
* BINARY NUMBER =
*      X[4]*Y↑3+X[3]*Y↑2+X[2]*Y↑1+X[1]*Y↑0          OR
* BINARY NUMBER =
*      (((0*Y+X[4])*Y+X[3])*Y+X[2])*Y+X[1]
*WHERE ↑ IS THE EXPONENT OPERATOR.
* X IS A CHARACTER & Y IS THE BASE.
*
*
*
*      ALGORITHM:
*ASCBIN:  FORM THE BASE IN ARA BASED ON THE FIRST CHAR.
*  OF THE NUMBER STRING; INCREMENT CHAR. PTR. IN X;
*ENTR2:  NUMBER (IN BA) GETS 0;
*NXTCHR:  IF THE CURRENT CHAR. POINTED TO BY X IS NOT A
*  DIGIT THEN EXIT ELSE INCREMT CHARACTER PTR IN INDEX;
*  CONVERT DIGIT TO BINARY;
*  NUMBER GETS NUMBER * BASE;
*  NUMBER GETS NUMBER + DIGIT;
*  GO TO OPERATE ON THE NEXT DIGIT (NXTCHR);
*
```

```
FF22 A6 00  ASCBIN LDA A  0,X      GETS CHR TO FORM BASE.
FF24 81 2E         CMP A  #$'.     TESTS FOR DECML STRNG.
FF26 2D 06         BLT    OCT      SKIPS IF BASE 8 (*).
FF28 2E 09         BGT    HEX      SKIPS IF BASE 16.
FF2A 86 0A         LDA A  #10      LOADS BASE 10 FOR CONVERSN.
FF2C 20 02         BRA    ASC1     SKIPS TO INC. TEXT POINTR.
FF2E 86 08  OCT    LDA A  #8       LOADS BASE 8 FOR CONVERSION.
FF30 08     ASC1   INX             INCREMENT PTR TO NEXT CHAR.
FF31 20 02         BRA    ASC2     SKIPS TO SAVE BASE.
FF33 86 10  HEX    LDA A  #16      LOADS BASE 16 FOR CONVERN.
FF35 97 07  ASC2   STA A  AR0      SAVES BASE IN BASE#.
            *
            *
FF37 5F     ENTR2  CLR B  NUMBER   GETS 0.
FF38 37            PSH B  -        (LOW NUMBER ON STACK).
FF39 D7 06         STA B  AR1      CLEARS HI OF BASE.
FF3B A6 00  NXTCHR LDA A  0,X      GETS CHAR TO CONVERT.
FF3D 08            INX             INC TO NEXT CHARACTER.
FF3E 81 30         CMP A  #$'0     TESTS FOR END-OF-STRING.
FF40 2D 20         BLT    AEXIT    EXITS IF END.
FF42 80 30         SUB A  #$'0     FORMS B. C. D. NUMBER.
FF44 81 0A         CMP A  #10      TESTS IF DECIMAL DIGIT.
FF46 2D 0A         BLT    ASC3     SKIPS IF DECIMAL.
FF48 81 10         CMP A  #16      TESTS FOR END OF STRING.
FF4A 2F 16         BLE    AEXIT    EXITS IF NOT A HEX DIGIT.
FF4C 80 07         SUB A  #7       FORMS A HEX B. C. D. DIGIT.
FF4E 81 10         CMP A  #16      TESTS FOR END-OF-STRING.
FF50 2C 10         BGE    AEXIT    EXITS IF CHAR > "F".
FF52 97 08  ASC3   STA A  DIGIT    SAVES DIGIT FOR ADD.
            *
FF54 DF 00  CNVASC STX    TMP      SAVES INDEX REG FOR MULT.
FF56 32            PUL A  -        RESTORES LO OF "NUMBER".
FF57 8D 3A         BSR    MULT     NUMBER GETS NUMBER * BASE.
FF59 9B 08         ADD A  DIGIT    NUMBER GETS NUMBER + DIGIT.
FF5B C9 00         ADC B  #0
FF5D 36            PSH A  -        SAVES LO OF NUMBER.
FF5E DE 00         LDX    TMP      RESTORES STRING POINTER.
FF60 20 D9         BRA    NXTCHR   GOES TO CONVRT NEXT CHAR.
FF62 32     AEXIT  PUL A  -        RESTORES "NUMBER" IN BA.
FF63 39            RTS             RETURNS TO CALLING PROGRAM.
```

```
                    *             BINARY TO ASCII
                    *
                    *
                    *        THE BINARY TO ASCII CONVERSION ROUTINE CONVERTS
                    * A 16 BIT BINARY NUMBER IN THE BA REGISTR (REG B & REG A)
                    * TO A STRING OF ASCII DIGITS.  THE ASCII STRING CAN BE IN
                    * ANY BBASE FROM BASE 2 THROUGH BASE 41.  THE VALUE OF THE
                    * BASE IS LOCATED IN THE ARITHMETIC PSEUDO-REGISTER ARB
                    * (ARB IS LOOCATED IN BYTE AR3 [LOC 4] AND AR2 [LOC 5]).
                    * WHEN THE ROUTINE IS ENTERED, THE POINTER TO THE OUTPUT
                    * LOCATION IS PASSED IN THE INDEX REG.  WHEN THE ROUTINE
                    * EXITS, THE INDEX POINTS TO THE LAST DIGIT IN THE STRING
                    * PLUS ONE.
                    *     CONVERSION IS DONE BY THE METHOD OF REPEATED
                    * DIVISION.  THE LOW ORDER DIGIT IS FORMED FIRST.  THE
                    * DIGITS ARE THEN PLACED ON THE STACK UNTIL CONVERSION IS
                    * COMPLETED.  THE DIGITS ARE THEN POPPED OFF THE STACK
                    * AND PLACED IN THE OUTPUT STRING.  THE TOP-OF-STACK IS
                    * INITIALIZED TO ALL ONES TO TELL WHEN ALL THE DIGITS
                    * HAVE BEEN POPPED OFF THE STACK.  AFTER THE DIVISION, THE
                    * DIGIT (THE REMAINDER OF THE DIVISION OPERATION)
                    * IS LOCATED IN THE AR 0 PART OF ARA (BYTE 7).  WHEN
                    * THE QUOTIENT OF THE DIVISION IS 0, THEN THE CONVERSION
                    * IS COMPLEATED.
                    *
                    *
                    *
FF64 DF 00          BINASC STX     TMP        SAVES OUTPUT POINTER.
FF66 34                    DES                /SETS THE TOP-OF-STACK TO
FF67 30                    TSX                /ALL ONES TO TELL END OF
FF68 6F 00                 CLR     0,X        /CHAR STRING (LAST CHAR IS
FF6A 63 00                 COM     0,X        /PUT ON STACK FIRST).
FF6C DE 04          BIN1   LDX     ARB        RESTORES DEVISOR (BASE).
FF6E DF 06                 STX     ARA
FF70 8D 3D                 BSR     DIVIDE     * QUOTIENT IN BA GETS THE
                    * REMAINDER OF # TO BE CONVERTED; REMAINDER IN ARA GETS
                    * THE LOW ORDER DIGIT.
FF72 97 02                 STA A   TMP1       SAVES A OF BA.
FF74 96 07                 LDA A   AR0        LOAD DIGIT (REMAINDER).
FF76 36                    PSH A   -          STACK DIGIT (REVERSE ORDER). G
FF77 96 02                 LDA A   TMP1       RESTORES A OF BA.
FF79 4D                    TST A   -          /TESTS IF QUOTIENT IS = 0
FF7A 26 F0                 BNE     BIN1       /(SIGNIFYING THAT
FF7C 5D                    TST B   -          /THE CONVERRSION
FF7D 26 ED                 BNE     BIN1       /IS DONE).
                    *
FF7F DE 00          BINSTR LDX     TMP        RESTORES OUTPUT POINTER.
FF81 32             BIN3   PUL A   -          UNSTACK A DIGIT.
FF82 4D                    TST A   -          TESTS IF NEG (END?).
FF83 2A 01                 BPL     BIN4       SKIPS IF A DIGIT.
FF85 39                    RTS                EXITS FROM SUBROUTINE.
FF86 81 09          BIN4   CMP A   #9         TESTS IF RESULT IS HEX.
FF88 2F 02                 BLE     BIN5       SKIPS IF DIGIT NOT HEX.
FF8A 8B 07                 ADD A   #7         FORMS HEX VALUE OF DIGIT.
FF8C 8B 30          BIN5   ADD A   #$'0       FORMS DECIMAL CHARACTER.
FF8E A7 00                 STA A   0,X        OUTPUTS CHARACTER.
FF90 08                    INX                POINTS TO NEXT CHARACTER.
FF91 20 EE                 BRA     BIN3       GOES BACK FOR NEXT DIGIT.
```

```
                   *          MULTIPLY ROUTINE
                   *
                   *      THE MULTIPLY ROUTINE MULTIPLIES TWO 16 BIT BINARY
                   * NUMBERS TOGETHER TO PRODUCE A 16 BIT RESULT.  THE BA
                   * REGISTERS AND ARA (BYTES 6 & 7) REGISTER ARE USED.
                   * THE CONTENTS OF ARA ARE UNCHANGED UPON PROGRAM EXIT.
                   *
                   *
                   *  BA GETS BA * ARA
                   *
                   *
                   *      MULTIPLYING IS ACCOMPLISHED BY REPEATED ADDITIONS
                   * OF ONE OF THE OPERATORS (OPERATOR ARA) INTO THE RESULT.
                   * THE RESULT STARTS OUT WITH A ZERO VALUE AND IS SHIFTED
                   * OVER ONE AFTER EACH ADDITION.  THE HIGHEST ORDER VALUE
                   * IS ADDED IN FIRST AND TTHEN, GOING TO THE RIGHT,
                   * (THUS SHIFTING THE ANSWER LEFT ONE TO BRING IN THE NEXT
                   * RIGHTMOST DIGIT) GETTING THE NEXT LOWERMOST SIGNIFICANT
                   * DIGIT.  THE NEXT RIGHTMOST BIT OF THE OTHER OPERAND
                   * (THE ONE ORIGINALLY IN BA) IS TESTED, AND IF ONE,
                   * ANOTHER ADDITION TAKES PLACE.  THIS IS REPEATED UNTIL
                   * THE FINAL SUM IS FORMED.
                   *
                   *
                   *
                   *  MULTIPLY ALGORITHM:
                   *
                   *MULT:  STACK BA; BA GETS 0; SET COUNT VALUE TO 16;
                   *MUL1:  SHIFT BA LEFT 1;
                   *  SHIFT LEFT ORIG BA VALUE ON STACK INTO CARRY;
                   *  IF CARRRY = 0 THEN GO TO MUL2
                   *  BA GETS BA + ARA;
                   *MUL2:  DECREMENT COUNT;
                   *  IF COUNT # 0 THEN GO TO MUL1 ELSE EXIT.
                   *
                   *
                   *
                   *
FF93 36            MULT   PSH A   -      PUTS THE ORIGINAL CONTENTS
FF94 37                   PSH B   -      OF BA ONTO THE STACK.
FF95 86 10                LDA A   #16    LOADS COUNT VALUE
FF97 36                   PSH A   -      ONTO THE STACK.
FF98 4F                   CLR A   -      BA GETS ZEROED.
FF99 5F                   CLR B
FF9A 30                   TSX            SET INDEX TO STACK.
FF9B 48            MUL1   ASL A   -      SHIFT LEFT BA.
FF9C 59                   ROL B
FF9D 68 02                ASL     2,X    SHIFTS ORIG. BA OPERAND
FF9F 69 01                ROL     1,X    ONE LEFT INTO CARRY.
FFA1 24 04                BCC     MUL2   SKIPS ADDING IF CARRY = 0.
FFA3 9B 07                ADD A   AR0    BA GETS BA + ARA.
FFA5 D9 06                ADC B   AR1
FFA7 6A 00         MUL2   DEC     0,X    TESTS IF DONE.
FFA9 26 F0                BNE     MUL1   GOES BACK IF NOT DONE.
FFAB 31                   INS            CLEANS UP THE STACK.
FFAC 31                   INS
FFAD 31                   INS
FFAE 39                   RTS            EXITS ROUTINE.
```

```
                    *          DIVIDE ROUTINE
                    *
                    *      THE DIVIDE SUBROUTINE DIVIDES THE 16 BIT NUMBER
                    * IN THE BA REGESTERS BY THE 16 BIT NUMBER IN THE PSEUDO
                    * REGESTER ARA (LOCATED IN BYTES 6 & 7).  UPON EXITING,
                    * BA WILL CONTAIN THE QUOTIENT OF THE DIVISION AND ARA
                    * WILL CONTAIN THE REMAINDER.  THE DIVIDEND BA IS DIVIDED
                    * BY THE DIVISOR ARA (I.E.  BA/ARA ).
                    *
                    *
                    *
                    *
                    *  DIVIDE ALGORITHM:
                    *
                    *DIVIDE:  X3,4 GETS BA (BA IS PUT ON THE STACK);
                    *  X1,2 GETS THE 16 BIT ARA VALUE (ARA PUT ON THE STACK);
                    *  COUNT GETS 1 + THE NUMBER OF NONSIGNIFICANT BITS IN
                    * THE DIVISOR [LEFT JUSTIFY X1,2 TO FIRST 1 BIT, COUNT
                    * GETS 1 + THE # OF LEADING ZEROS IN X1,2] [COUNT WILL BE
                    * FROM 1 TO 17];
                    *  BA GETS X3,4 [RESTORES BA];
                    *  X3,4 GETS 0 [INITIALIZES THE QUOTIENT];
                    *DIV3:  BA GETS BA - X1,2;
                    *  IF THERE WAS A BORROW [I.E. DIVIDEND IN BA < DIVISOR
                    * IN X1,2 (ARA)] THEN BA GETS BA + X1,2 [ORIGINAL BA
                    * VALUE RESTORED] & CARRY CLEARED ELSE CARRY IS SET;
                    *DIV5:  X3,4 [QUOTIENT] GETS CARRY LEFT SHIFTED IN;
                    *  X1,2 [DIVISOR] GETS SHIFTED RIGHT ONE PLACE WITH ZERO
                    * FILLED IN FROM THE LEFT SIDE;
                    *  DECREMENT COUNT;
                    *  EXIT IF DONE ELSE GO TO DIV3;
                    *
                    *
                    *
                    *
                    *
FFAF 36             DIVIDE PSH A   -       LOADS DIVIDEND INTO X3,4.
FFB0 37                    PSH B
FFB1 96 06                 LDA A   AR1     LOADS DIVISOR FROM ARA.
FFB3 D6 07                 LDA B   AR0
FFB5 37                    PSH B   -       PUTS DIVISOR INTO X1,2.
FFB6 36                    PSH A
FFB7 34                    DES             SET UP SPACE FOR COUNT.
FFB8 30                    TSX             INDEX GETS STACK POINTER.
FFB9 86 01                 LDA A   #1      INITIALIZE COUNT.
FFBB 6D 01                 TST     1,X     TESTS FOR HI DIVISR BIT ON.
FFBD 2B 0B                 BMI     DIV2    SKIPS IF ON.
FFBF 4C             DIV1   INC A   -       COUNTS LEADING ZEROS.
FFC0 68 02                 ASL     2,X     LEFT JUSTIFIES X1,2.
FFC2 69 01                 ROL     1,X
FFC4 2B 04                 BMI     DIV2    SKIPS IF NO LEADING ZERO.
FFC6 81 11                 CMP A   #17     TESTS FOR ALL ZERO DIVISOR.
FFC8 26 F5                 BNE     DIV1    GOES BACK IF BITS LEFT.
FFCA A7 00          DIV2   STA A   0,X     SETS COUNTER.
FFCC E6 03                 LDA B   3,X     BA GETS ORIGINAL
FFCE A6 04                 LDA A   4,X     DIVIDEND VALUE.
FFD0 6F 03                 CLR     3,X     CLEARS X3,4 FOR FORMATION
FFD2 6F 04                 CLR     4,X     OF THE QUOTIENT.
```

```
FFD4 A0 02   DIV3   SUB A   2,X      START OF DIVIDE LOOP.
FFD6 E2 01          SBC B   1,X
FFD8 24 07          BCC     DIV4     SKIP IF DIVIDEND < DIVISOR.
FFDA AB 02          ADD A   2,X      RESTORES DIVIDEND IN BA.
FFDC E9 01          ADC B   1,X
FFDE 0C             CLC              CLEARS THE CARRY.
FFDF 20 01          BRA     DIV5     SKIPS WITH CARRY CLEAR.
FFE1 0D      DIV4   SEC              SETS CARRY TO 1.
FFE2 69 04   DIV5   ROL     4,X      SHIFT CARRY INTO
FFE4 69 03          ROL     3,X      QUOTIENT X3,4
FFE6 64 01          LSR     1,X      SHIFTS DEVISOR X1,2
FFE8 66 02          ROR     2,X      RIGHT ONE.
FFEA 6A 00          DEC     0,X      DECREMENTS COUNTER.
FFEC 26 E6          BNE     DIV3     GOES BACK IF NOT DONE.
FFEE D7 06          STA B   AR1      STORES REMAINDER IN ARA.
FFF0 97 07          STA A   AR0
FFF2 31             INS              CLEANS UP THE STACK.
FFF3 31             INS
FFF4 31             INS
FFF5 33             PUL B   STORES   QUOTIENT IN BA.
FFF6 32             PUL A
FFF7 39             RTS              EXITS ROUTINE.
             *
             *
             *
FFF8 0104    IRQ    FDB     $0104    INTERRUPT REQUEST VECTOR.
FFFA FE4F    SWI    FDB     $FE4F    SOFTWARE INT.  VECTOR ADDR.
FFFC 0108    NMI    FDB     $0108    NIN-MASKABLE-INT.  VECT.
FFFE FC00    RST    FDB     $FC00    RESTART VECTOR ADDRESS.
             *
             *
             *
             *  END OF PDS SOURCE LISTING.
             *
             *
                    END
TOTAL ERRORS 00000
```

```
                    NAM     PDS-V3A
                    OPT     0,NOG
              *
              *     PROGRAMMED BY ERIC JAMESON
              *
              *     COPYRIGHT 1976 SPHERE CORPORATION
              *     791 SO. 500 W.;  BOUNTIFUL, UTAH   84010
              *
              *     SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
              *     DISTRIBUTION AND USE OF THE PDS SOFTWARE.
              *     NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
              *     WRITTEN PERMISSION OF SPHERE CORP.
              *
              *
              * THE PROGRAM DEVELOPMENT SYSTEM (PDS V3A) IS A SET OF
              * PROGRAMS RESIDING ON ERASABLE PROGRAMMABLE READ ONLY
              * MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS
              * SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE
              * DEVELOPMENT OF COMPUTER PROGRAMS.
              *     TOWARD THIS END, THE 4 PDS EPROMS CONTAIN A CURSOR
              * BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING
              * AID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16
              * BIT MULTIPLY AND DIVIDE, ASCII-TO-BINARY, AND
              * BINARY-TO-ASCII ROUTINES.
              *
              *
              *
              *                 MEMORY MAP
              *
       0000   TMP     EQU     $00
       0002   TMP1    EQU     $02
       0004   ARB     EQU     $04        16 BIT ACC. PSEUDO REG B.
       0004   AR3     EQU     $04        _HI BYTE OF ARB.
       0003   AR2     EQU     $03        _LO BYTE OF ARB.
       0006   ARA     EQU     $06        16 BIY ARITH PSEUDO REG A.
       0006   AR1     EQU     $06        _HI BYTE OF ARA.
       0007   AR0     EQU     $07        _LO BYTE OF ARA.
       0008   DIGIT   EQU     $08        BYTE USED BY ASCBIN FOR TMP.
       000A   OUTEND  EQU     $0A        END OF OUTPUT BUFFER TEXT.
       000C   BUFADR  EQU     $0C        START OF I/O BUFFER (PTR)
       000E   BUFEND  EQU     $0E        PTR. TO END OF I/O BUFFR.
       0011   OUTBUF  EQU     $11        START OF OUTPUT BUFFER.
       0014   SRCADR  EQU     $14        SOURCE FOR TEXT MOVES.
       0016   DSTADR  EQU     $16        DEST. ADDR. FOR TEXT MOVE.
       001A   ENDMEM  EQU     $1A        LAST ADDRES OF REAL MEMORY.
       001C   CSRPTR  EQU     $1C        PTR TO CURSER ON SCREEN.
       001E   BUFPTR  EQU     $1E        TEMP PTR USED BY OUTSTR.
       0020   BUFFLO  EQU     $20        PTR TO END OF LOW EDIT TXT.
       0022   BUFFHI  EQU     $22        PTR TO START OF HI TEXT.
       0024   SCNPTR  EQU     $24        PTR. TO BUFFRD TXT START.
       0026   SRCASM  EQU     $26        PTR TO ASSMBLR SOURCE CODE.
       002A   ONDVAL  EQU     $2A        HAS ASSMBLR OPERND VALUE.
       002C   SYMVAL  EQU     $2C        VALUE PUT IN ASSM. SYMTBL.
       002E   BRKSAV  EQU     $2E        TEMP SAVE FOR BRKPT DATA.
       0030   BRKADR  EQU     $30        ADDDRESS OF BREAKPOINT.
       0032   EDIT    EQU     $32        0 IF EDITOR IS NOT RUNNING.
       0035   IOBUFF  EQU     $35        I-O BUFFER FOR DEBUGGER.
       0040   PCVAL   EQU     $40        PROGRAM COUNTER FOR ASSM.
```

```
                    *                  INITIALIZATION
                    *
                    *
                    *
                    *    THE INITIALIZATION ROUTINES SET UP THE INITIAL VALUES
                    * UPON SYSTEM RESET.
                    *
                    *
FC00                       ORG      $FC00
FC00 8E 01FF RESTRT LDS    #$1FF    SETS STACK POINTER
FC03 30                    TSX               MOVES STK PTR TO INDEX REG
FC04 DF 26                 STX      SRCASM   SETS ASSEMBLR OUTPUT PTR
FC06 DF 0C                 STX      BUFADR   INIT INPUT BUFFR ADDR.
FC08 86 07                 LDA A    #07      INHTLZ. KEYBOARD PIA.
FC0A B7 F001               STA A    $F001
FC0D CE 0FFF               LDX      #$FFF    LAST LOCATN OF MEMORY.
FC10 DF 0E                 STX      BUFEND   INIT END-OF-EDIT BUFFR.
FC12 DF 1A                 STX      ENDMEM   INIT END-OF-MEM ADDR.
                    *
                    *
                    *
                    *
                    *
                    *                  COMMAND LANGUAGE
                    *
                    *     THIS EXECUTIVE ACCEPTS COMMANDS FROM THE KEYBOARD TO
                    * DETERMINE WHAT UTILITY IS TO BE RUN.  INVALID COMMANDS
                    * WILL SPACE THE CURSOR DOWN ONE LINE.  DO NOT SPACE OFF THE
                    * BOTTOM OF THE SCREEN.
                    *
                    *
FC14 8D 1C    EXEC         BSR      HOME     CSRPTR IS HOMED.
FC16 8D 20                 BSR      CLEAR    CLEARS SCREEN
FC18 BD FD16  EXEC1        JSR      CRLF     NEW LINE.
FC1B BD FE5C               JSR      INPCHR   GETS & DISPLAYS CHR.
FC1E 81 01                 CMP A    #$01     TESTS IF ASSEMBLY COMMD.
FC20 26 03                 BNE      EXEC2    SKIPS IF OTHER COMMAND.
FC22 BD FDA1               JSR      ASMBLR   JUMPS TO ASSEMBL PRROGRM.
FC25 81 05    EXEC2        CMP A    #$05     TESTS IF CONTRL 'E'.
FC27 26 02                 BNE      EXEC3    SKIPS IF NOT EDIT CMD.
FC29 8D 35                 BSR      EDITOR   JUMPS TO EDIT TEXT.
FC2B 81 04    EXEC3        CMP A    #$04     TESTS FFOR CONTRL 'D'.
FC2D 26 E9                 BNE      EXEC1    LOOPS BACK IF NOT DEBUG.
FC2F 7E FE4F               JMP      DEBUG    JUMPS TO DEBUGGER.
```

```
*                      THE EDITOR
*
*
*      THE EDITOR ALLOWS INPUT FROM THE KEYBOARD INTO A
* BUFFER MEMORY.  INPUT IS DISPLAYED ON THE SCREEN.  WHEN
* IT IS TYPED IN, THE SCREEN TEXT CAN THEN BE EDITED BY USE
* OF THE CURSOR.  WHEN THE SCREEN IS FULL OR EDITING IS
* FINISHED, THE DATA IS SCROLLED OFF THE SCREEN INTO THE
* EDIT BUFFER.  WHEN TEXT IS SCROLLED OFF THE TOP OF THE
* SCREEN, IT IS STORED FROM THE BUFFER ADDRESS POINTER
* (BUFADR) TO THE LOW BUFFER POINTER (BUFFLO).  BUFFLO
* POINTS TO THE END OF TEXT + ONE (I.E. IT POINTS TO THE
* FIRST UNUSED BYTE).  WHEN IT IS SCROLLED OFF THE BOTTOM
* OF THE SCREEN, IT IS STORED IN THE TOP OF THE EDIT BUFFER.
* THE TEXT GOES FRROM THE HIGH BUFFER POINTER (BUFFHI) TO
* THE END OF BUFFER POINTER (BUFEND).  BUFFHI POINTS TO
* THE END OF TEXT - ONE (I.E. IT POINTS TO THE LAST UNUSED
* BYTE IN THE BUFFER).  WHEN THE TEXT IS SCROLLED UP
* OFF THE TOP OF THE SCREEN, TEXT IS TAKEN FROM THE HIGH
* AREA OF THE EDIT BUFFER AND DISPLAYED ON THE LAST LINE OF
* THE SCREEN.  WHEN TEXT IS SCROLLED DOWN OFF THE BOTTEM,
* TEXT, IF ANY EXISTS, IS MOVED FROM THE LOW EDIT BUFFER
* AREA TO THE TOP LINE OF THE SCREEN.
*
*
*
```


```
*
*
*                       POINTERS USED
*
* CSRPTR    POSITION OF CHARACTERS INSERTED ON THE SCREEN.
* SCNPTR    POSITION OF START OF EDITED TEXT ON SCREEN.
* BUFADR    START OF TEXT BUFFER IN MAIN MEMORY.
* BUFEND    END OF TEXT BUFFER IN MAIN MEMORY.
* BUFFLO    END OF TEXT SCROLLED OFF TOP OF SCREEN.
* BUFFHI    START OF TEXT SCROLED OFF BOTTOM OF SCREEN.
* BUFLEN    NOT CURRENTLY USED.
*
*
*
*
*
*                     EDITOR COMMANDS
*
* "UP ARROW"  MOVES CURSOR UP ONE LINE; CSRPTR GETS
*  CSRPTR-32; CALL NDRFLO.
*
* "DOWN ARROW"  MOVES CURSOR DOWN ONE LINE; CSRPTR GETS
*  CSRPTR+32; CALL OVRFLO.
*
* "RIGHT ARROW"  MOVES CURSOR ONE POSITION RIGHT; CSRPTR
*  GETS CSRPTR+1; CALL OVRFLO.
```
0

```
* "LEFT ARROW"  MOVES CURSOR ONE POSITION LEFT; CSRPTR
*  GETS CSRPTR-1; CALL NDRFLO.
*
* "CONTROL & LEFT ARROW (ON KEYBOARD)"  LEFT JUSTIFY CURSOR;
*  CSRPTR GETS CSRPTR TRUNCATED; CALL NDRFLO FOR SCNLOC CHK.
*
* "PUTCHR"  OUTPUTS CHARACTER; CSRPTR GETS
*  CSRPTR+1; GOES TO OVRFLO.
*
* "ENDCHR"  TERMINATION CHAR; CLEAR EDIT FLAG;
*  EXIT THE EDITOR.
*
* "HOME"  HOMES CURSOR POINTER; CSRPTR GETS E000; NDRFLO.
*
* "CLEAR"  CSRPTR TO END OF THE SCREEN GETS SPACES.
*
* "CTRL I"  INSERT A LINE AT THE LAST LINE ON THE SCREEN;
*  CALL OVR1 (SCROLLS UP ONE LINE); CSRPTR GETS E1E0.
*
* "CTRL D"  DELETE LAST LINE; SCROLL DOWN (UNDR2);
*  CSRPTR GETS E1E0.
*
*
*
*
* OVERFLOW CHECKS IF SCROLL UP IS NEEDED; IF IT IS, IT
*  SCROLLS UP AND MOVES DATA TO & FROM THE BUFFERS.
*
* OVRFLO: IF CSRPTR < E200 THEN RETURN; IF EDIT IS ON THEN
* OVR1:   BUFFLO+ GETS SCNPTR TO 'C.R.';
* DSTADR GETS CSRPTR GETS E1E0 (LAST LINE ON SCREEN);
* IF EDIT IS ON AND BUFFHI < BUFEND THEN MOVE THE TEXT
* FROM BUFFHI TO 'C.R.' TO THE LAST LINE.
*
*
*
*
* UNDERFLOW CHECKS IF SCROLL DOWN IS NEEDED AND MOVES
*  DATA TO AND FROM THE BUFFERS.  CURSER HAD BEEN
*  MOVED OFF THE TOP OF THE SCREEN AND IS NOW PUT AT THE
*  HOME POSITION ON THE SCREEN.
*
* NDRFLO:  IF CSRPTR > DFFF THEN RETURN (GO TO OVRFLO);
*  IF EDIT FLAG IS ON THEN MOVE LAST LINE TO BUFFHI
*   ON DOWN; SCRLDN; CSRPTR GETS E000; MOVE LINE FROM
*   BUFFLO TO FIRST LINE ON THE CRT.
*
*
*
*
*  NOTE:  DON'T SCROLL OFF SCREEN IN EXEC UNTIL AFTER
*    THE EDITOR HAS BEEN RUN.
*
*  NOTE:  EVERY LINE MUST HAVE A C.R. ON IT.
```

```
FC32 CE E000 HOME   LDX     #$E000   LOADS HOME POSITION.
FC35 DF 1C          STX     CSRPTR   STORES HOME IN CURSOR PTR.
FC37 39             RTS              RETURNS TO CALLER.
                    *
                    *
FC38 C6 60   CLEAR  LDA B   #$60     LOADS BLANK (C. R. ).
FC3A CE E200        LDX     #$E200   LOADS END-OF-SCREEN PTR.
FC3D 09      CLEAR1 DEX              DECREMENTS BLANKING PTR.
FC3E E7 00          STA B   0,X      BLANKS LOCATION.
FC40 9C 1C          CPX     CSRPTR   TESTS IF DONE.
FC42 26 F9          BNE.    CLEAR1   BRANCHES BACK IF NOT DONE.
FC44 39             RTS              RETURNS.
                    *
                    *
                    *
                    *       GETCHR INPUTS A CHARACTER INTO ACC A WITHOUT
                    *       MOVING THE CURSOR, AND BLINKS THE CURSOR.
                    *
FC45 DE 1C   GETCHR LDX     CSRPTR   LOADS CRT CURSOR POSITION.
FC47 63 00          COM     0,X      COMPLIMENT (FLASH POSITION).
FC49 CE 26F0        LDX     #9968    LOADS BLINK COUNT VALUE.
FC4C 09      GET1   DEX              COUNT GETS COUNT-1.
FC4D 27 F6          BEQ     GETCHR   RESETS CTR WHEN TIMED OUT.
FC4F 7D F001        TST     $F001    TESTS IF A CHAR TYPED IN.
FC52 2A F8          BPL     GET1     BRANCH IF CHAR NOT ENTERED.
FC54 DE 1C          LDX     CSRPTR   LOADS CURSOR POSITION.
FC56 A6 00          LDA A   0,X      TESTS IF BLINKED (SOLID).
FC58 2A 02          BPL     GET2     SKIPS IF NOT BLINKED.
FC5A 63 00          COM     0,X      CLEARS THE CHARACTER.
FC5C B6 F000 GET2   LDA A   $F000    LOADS A WITH CHAR.
FC5F 39             RTS              RETURNS TO CALLER.
                    *
                    *
                    *
                    *
                    *        EDITOR IS THE MAIN ENTRY POINT FOR EDITING.
                    *
FC60 DE 0C   EDITOR LDX     BUFADR   BUFFLO GETS THE
FC62 DF 20          STX     BUFFLO   VALUE OF BUFADR.
FC64 DE 0E          LDX     BUFEND   BUFFHI GETS THE
FC66 DF 22          STX     BUFFHI   VALUE OF BUFEND
FC68 8D C8   REEDIT BSR     HOME     ENTRY POINT FOR
FC6A 8D CC          BSR     CLEAR    RE-EDITING TEXT.
FC6C 97 32   EDITRD STA A   EDIT     TURNS ON EDIT MODE .
FC6E DE 1C   EDITIN LDX     CSRPTR   SETS SCNPTR TO CSRPTR.
FC70 DF 24          STX     SCNPTR
                    *
                    *
FC72 8D D1   EDREAD BSR     GETCHR   X GETS CSRPTR & A GETS CHR.
FC74 81 1B   ENDCHR CMP A   #$1B     TESTS FOR AN "ESC" CHAR.
FC76 26 04          BNE     ED1      SKIPS IF NOT EDIT END.
FC78 7F 0032        CLR     EDIT     TURNS OFFF EDIT FLAG.
FC7B 39             RTS              EXITS THE EDITOR.
FC7C 8D 02   ED1    BSR     INSERT   EDITS CHARACTER.
FC7E 20 F2          BRA     EDREAD   GOES FOR NEXT CHARACTER.
```

```
                  *       FOLLOWING IS THE MAIN EDITOR EXECUTION LOOP.
                  *
                  *
                  *
                  *
                  *
FC80 81 09        INSERT CMP A  #$09     TESTS FOR A CONTROL 'I'.
FC82 2D 16               BLT    DELETE   SKIPS TO DELETE COMMD.
FC84 2E 02               BGT    CR       SKIPS FOR NEXT TEST.
FC86 20 70               BRA    OVR1     SCROLLS UP ONE LINE.
                  *
                  *
FC88 81 0D        CR     CMP A  #$0D     TESTS FOR CARRIAGE RETURN.
FC8A 2D A6               BLT    HOME     SKIPS IF HOME CSR COMMD.
FC8C 2E 02               BGT    RTCSR    GOES TO  NEXT COMND TEST.
FC8E 86 60               LDA A  #$60     LOADS INTERNAL C. R.  VALUE.
                  *
                  *
FC90 81 12        RTCSR  CMP A  #$12     TESTS FOR RIGHT ARROW.
FC92 2D 2A               BLT    SUB32    SKIPS IF AN "UP ARROW".
FC94 2E 08               BGT    LFTCSR   SKIPS IF A "LEFT ARROW".
FC96 DE 1C        RTARRO LDX    CSRPTR   LOADS CURSOR POINTER.
FC98 20 1F               BRA    PUTCH1   STORES & INCREMENTS CSR.
                  *
                  *
FC9A 8D 3F        DELETE BSR    UNDR2    SCROLLS DOWN 1 LINE.
FC9C 20 6A               BRA    OVR2     CSRPTR GETS E1E0 (LAST LIN).
                  *
                  *
FC9E 81 14        LFTCSR CMP A  #$14     TESTS IF "LEFT ARROW".
FCA0 2D 26               BLT    ADD32    SKIPS IF "DOWN ARROW".
FCA2 2E 03               BGT    CLER     SKIPS FOR NEXT TEST.
FCA4 09                  DEX             SUB.  1 FROM CSRPTR.
FCA5 20 13               BRA    PUTCH2   STORES CURSER POINTER.
                  *
FCA7 81 1F        CLER   CMP A  #$1F     TESTS FOR CTRL BACK ARROW.
FCA9 2D 8D               BLT    CLEAR    GOES TO CLEAR SCREEN.
FCAB 27 62               BEQ    LFTJST   MOVES CSR TO LEFT OF SCREEN.
                  *
                  * ALL OTHER CHARACTERS FALL THRU TO PUTCHR.
                  *
```

```
                     *      PUTCHR DISPLAYS A CHARACTER ON THE CRT DISPLAY AND
                     *        INCREMENTS THE CURSER POINTER AS WELL AS CHECKING
                     *        AND HANDLING CARRIAGE RETURNS.
                     *
                     *
                     *
FCAD DE 1C           PUTCHR LDX      CSRPTR      LOADS OLD CSRPTR.
FCAF 81 0D                  CMP A    #$0D        TESTS FOR EXTERNAL C. R.
FCB1 27 04                  BEQ      CRLF1       SKIPS TO DO A C. R.  L. F.
FCB3 A7 00                  STA A    0,X         DISPLAYS CHAR ON SCREEN.
FCB5 81 60                  CMP A    #$60        TESTS FOR INTERNAL C. R.
FCB7 27 5D           CRLF1  BEQ      CRLF        SKIPS FOR CR.  LF.
FCB9 08              PUTCH1 INX                  INCREMENTS CSRPTR.
FCBA DF 1C           PUTCH2 STX      CSRPTR      SAVES NEW CSRPTR.
FCBC 20 12                  BRA      NDRFLO      TESTS FOR OVRFLO & UNDRFLO.


FCBE DE 1C           SUB32  LDX      CSRPTR      LOADS CURRENT CRSR POSITION.
FCC0 C6 20                  LDA B    #32         LOADS LOOP COUNT.
FCC2 09              SUB32A DEX                  DECRIMENTS CSRPTR.
FCC3 5A                     DEC B    -           DECRIMENTS LOOP COUNTR.
FCC4 26 FC                  BNE      SUB32A      SKIPS BACK IF NOT DONE.
FCC6 20 06                  BRA      ADD2        SKIPS TO CHECK UNDRFLO.


FCC8 C6 20           ADD32  LDA B    #32         LOADS LOOP COUNTER.
FCCA 08              ADD32A INX                  INCRE.  CSRPTR IN INDEX.
FCCB 5A                     DEC B    -           DECRIMENTS LOOP COUNTER.
FCCC 26 FC                  BNE      ADD32A      SKIPS BACK IF NOT DONE.
FCCE DF 1C           ADD2   STX      CSRPTR      SAVES CSRPTR.


                     * NDRFLO (UNDERFLOW) CHECKS FOR THE CURSOR GOING OFF THE
                     *   TOP OF THE SCREEN.  THE INDEX REG.  CONTAINS THE CURSOR
                     *   POINTER WHEN THE ROUTINE IS ENTERED.
                     *
                     *
FCD0 8C E000         NDRFLO CPX      #$E000      TESTS IF CSRPTR >= DFFF.
FCD3 2C 0B                  BGE      OVRFLO      SKIPS IF CSRPTR GREATER.
FCD5 D6 32                  LDA B    EDIT        TESTS IF EDITOR IS ON.
FCD7 27 02                  BEQ      UNDR2       SKIP TO EXIT IF EDITOR OFF.
FCD9 8D 6D                  BSR      MOVE2       MOVES LAST LINE TO BUFFHI.
FCDB BD FD73         UNDR2  JSR      SCRLDN      MOVES ALL LINES DOWN ONE.
FCDE 8D 3A                  BSR      MOVE3       MOVES BUFFLO TO TOP OF CRT.
```

```
                         *      OVERFLOW CHECKS FOR SCROLLING UP (CURSOR IS OFF
                         * THE BOTTON OF THE SCREEN); INDEX CONTAINS THE CURSOR
                         * POINTER UPON ENTRY.
                         *
                         *
FCE0 8C E200 OVRFLO CPX     #$E200    TESTS AND EXITS IF
FCE3 2B 12          BMI     OVREXT    CURSOR ON SCREEN.
FCE5 8D 11          BSR     OVR1      DOES OVR1 CHECKING.
FCE7 D6 32          LDA B   EDIT      TESTS IF EDIT IS ON.
FCE9 27 0C          BEQ     OVREXT    EXITS IF IT IS OFF.
FCEB DE 22          LDX     BUFFHI    LOADS HI TEXT POINTR.
FCED 9C 0E          CPX     BUFEND    TESTS IF PTRS NOT EQU.
FCEF 27 06          BEQ     OVREXT    EXITS IF NO TEXT.
FCF1 8D 44          BSR     MOVE1A    MOVES CHRS TO LAST LINE.
FCF3 DE 14          LDX     SRCADR    RESETS NEW BUFFHI
FCF5 DF 22          STX     BUFFHI    LOCATION.
FCF7 39      OVREXT RTS               EXITS BACK TO EDIT.
                         *
                         *
                         *
                         *      OVR1 DOES ACTUAL SCROLLING UP.
                         *
FCF8 D6 32   OVR1   LDA B   EDIT      TESTS IF EDIT IS ON.
FCFA 27 0A          BEQ     OVR1A     SKIPS IF EDIT OFF.
FCFC DE 20          LDX     BUFFLO    LOADS TEXT PTR LOW.
FCFE DF 16          STX     DSTADR    DESTINATION OF TEXT MOVE.
FD00 DE 24          LDX     SCNPTR    SOURCE FOR MOVE.
FD02 8D 34          BSR     MOVE1     MOVES LIN1 TO BUFFFLO.
FD04 DF 20          STX     BUFFLO    SAVES NEW BUFFLO PTR.
FD06 8D 58   OVR1A  BSR     SCRLUP    SCROLLS SCREEN UP 1.
FD08 CE E1E0 OVR2   LDX     #$E1E0    CURSOR GETS PUT ON
FD0B DF 1C          STX     CSRPTR    LAST LINE.
FD0D DF 16          STX     DSTADR    INIT DEST FOR NEXT MOVE.


                         *      FOLLOWING MOVE THE CURSOR.
                         *
FD0F D6 1D   LFTJST LDA B   CSRPTR+1  LOADS LO BYTE OF PTR.
FD11 C4 E0          AND B   #$E0      TRUNCATES TO LEFT OF LINE.
FD13 D7 1D          STA B   CSRPTR+1  SAVES L. J. ED PTR.
FD15 39             RTS               RETURNS TO EDITOR.
                         *
                         *
FD16 8D B0   CRLF   BSR     ADD32     LINE FEED.
FD18 20 F5          BRA     LFTJST    CARRIAGE RETURN.
```

```
                  *      MOVE INSTRUCTIONS MOVE FROM ONE BUFFER AREA TO
                  * ANOTHER BUFFER AREA.
                  *
                  *
                  *
                  *      MOVE3 CALCULATES THE SOURCE ADDRESS OF THE DATA IN
                  * BUFFLO (IF IT EXISTS) FOR MOVING TO THE FIRST LINE ON
                  * THE CRT.  MOVE 1 IS THEN ENTERED TO DO THE MOVING.
                  *
                  *
FD1A DE 24        MOVE3  LDX    SCNPTR   CSRPTR GETS E000 (HOME).
FD1C DF 16               STX    DSTADR   SETS MOVE ADDRESS.
FD1E DE 20               LDX    BUFFLO   LOADS LO BUFFR ADDR.
FD20 9C 0C               CPX    BUFADR   TESTS IF DATA STRING.
FD22 27 23               BEQ    MOVEXT   EXITS IF EMPTY.
FD24 09                  DEX             MOVES BACK FROM BLANK.
FD25 9C 0C        MV31   CPX    BUFADR   TESTS IF SRCADR = BUFFADR.
FD27 27 08               BEQ    MV32     MOVES IF START OF LINE.
FD29 09                  DEX             NEXT LOWER CHAR.
FD2A E6 00               LDA B  0,X      GETS SOURCE CHAR FOR TEST.
FD2C C1 60               CMP B  #$60     TESTS FOR "C.R.".
FD2E 26 F5               BNE    MV31     SKIPS BACK UNTIL "C.R.".
FD30 08                  INX             POINTS BACK TO FIRST CHAR.
FD31 DF 20        MV32   STX    BUFFLO   SAVES LO ADDRESS.
FD33 20 03               BRA    MOVE1    MOVES DATA.


                  *           MOVE1
                  *
                  *      MOVE1 MOVES A SET OF CHARACTERS FROM EITHER THE TOP
                  * LINE OF THE SCREEN TO BUFFLO OR FROM BUFFHI TO THE
                  * BOTTOM LINE OF THE SCREEN.  THE SOURCE ADDRESS IS PASSED
                  * IN THE INDEX REG., THE DESTINATION ADDRESS IN DSTADR,
                  * AND THE MOVE IS TERMINATED BY A "C.R." IN THE LINE OF
                  * TEXT BEING MOVED.
                  *
                  *
                  *
                  *
FD35 DE 14        MOVE   LDX    SRCADR   LOADS SOURCE ADDRESS INTO X.E
FD37 08           MOVE1A INX             POINTS TO NEXT SOURCE CHAR.
FD38 E6 00        MOVE1  LDA B  0,X      LOADS SOURCE CHARACTER.
FD3A DF 14               STX    SRCADR   SAVES THE SOURCE POINTER.
FD3C DE 16               LDX    DSTADR   LOADS DESTINATION ADDRESS.
FD3E E7 00               STA B  0,X      STORES CHAR. IN DESTINATION.
FD40 08                  INX             NEXT DESTINATION ADDRESS.
FD41 DF 16               STX    DSTADR   SAVES DESTINATION PTR.
FD43 C1 60               CMP B  #$60     TESTS IF MOVE FINSHED (CR).
FD45 26 EE               BNE    MOVE     SKIPS BACK IF NOT DONE.
FD47 39           MOVEXT RTS             RETURNS TO CALLER.
```

```
                    *          MOVE2 SUBROUTINE
                    *
                    *       THE MOVE2 SUBROUTINE MOVES THE LAST LINE ON THE
                    * SCREEN TO THE HIGH AREA OF THE BUFFER (BUFFHI) DURING
                    * SCROLLING.  THE TEXT IS TEMPORARILY STORED ON THE STACK
                    * DURING THE MOVE.  THE MOVE IS TERMIINATED BY A "C.R.".
                    * THE TEXT IS STORED AT BUFFHI ON DOWN.
                    *
                    *
                    *
                    *
FD48 8D BE          MOVE2  BSR    OVR2      X GETS ADDR OF LAST LINE.
FD4A 5F                    CLR B  -         SETS TERMINATION FOR
FD4B 37             MV21   PSH B  STACK     POPPING.
FD4C E6 00                 LDA B  0,X       LOADS SOURCE CHAR.
FD4E 08                    INX              POINTS TO NEXT CHAR.
FD4F C1 60                 CMP B  #$60      TESTS IF LINE TO "C.R."
FD51 26 F8                 BNE    MV21      MOVED TO STACK.
FD53 DE 22          MV22   LDX    BUFFHI    INIT. DESTINATION.
FD55 E7 00          MV23   STA B  0,X       STORES CHAR.
FD57 09                    DEX              POINTS TO NEXT LOCATION.
FD58 DF 22                 STX    BUFFHI    UPDATES BUFFER PTR.
FD5A 33                    PUL B  -         GETS NEXT CHAR.
FD5B C1 00                 CMP B  #00       TESTS IF ALL CHRS STORED.
FD5D 26 F6                 BNE    MV23      SKIPS BACK IF NOT STORED.
FD5F 39             MOVEX  RTS              RETURNS TO CALLER.


                    *      SCROLLUP MOVES ALL LINES UP 1, & CLEARS LAST LINE.
                    *
FD60 CE E000        SCRLUP LDX    #$E000    SETS CRT HOME POSITION.
FD63 E6 20          SCRP1  LDA B  $20,X     GETS CHAR FROM NEXT LINE.
FD65 E7 00                 STA B  00,X      STORES CHAR ON PREV. LINE.
FD67 08                    INX              POINTS TO NEXT LINE.
FD68 8C E1E0               CPX    #$E1E0    TESTS IF MOVE DONE.
FD6B 26 F6                 BNE    SCRP1     GOES BACK IF NOT DONE.
FD6D DF 1C                 STX    CSRPTR    SETS CSRPTR TO LAST LINE.
FD6F BD FC38               JSR    CLEAR     CLEARS LAST LINE.
FD72 39                    RTS              EXITS.
```

```
                        *       SCRLDOWN MOVES ALL LINES DOWN ONE AND
                        * CLEARS THE TOP LINE ON THE SCREEN.
                        *
FD73 CE E1DF SCRLDN LDX      #$E1DF    INITIALIZES THE POINTER.
FD76 E6 00   SCRD1  LDA B    0,X       LOADS DATA TO BE MOVED.
FD78 E7 20          STA B    $20,X     MOVES DATA DOWN ONE LINE.
FD7A DF 1C          STX      CSRPTR    SAVES CURSOR.
FD7C 09             DEX                POINTS TO NEXT BYTE.
FD7D 8C DFFF        CPX      #$DFFF    TESTS IF MOVE FINISHED.
FD80 26 F4          BNE      SCRD1     SKIPS BACK IF NOT DONE.
FD82 C6 60          LDA B    #$60      LOADS BLANK TO CLEAR LINE.
FD84 08      SCRD2  INX                POINTS TO NEXT CHARACTER.
FD85 E7 00          STA B    0,X       CLEARS BYTE ON LINE 1.
FD87 8C E01F        CPX      #$E01F    TESTS IF LINE 1 CLEARED.
FD8A 26 F8          BNE      SCRD2     SKIPS BACK IF NOT CLEARED.
FD8C 39             RTS                RETURNS.



                        *       OUTSTRING PRINTS OUT THE STRING BETWEEN THE
                        *       OUTBUF POINTER AND THE BUFEND POINTER.
                        *
FD8D DE 11   OUTSTR LDX      OUTBUF    BUFPTR GETS START OF TEXT.
FD8F A6 00   OUT1   LDA A    0,X       LOADS CHAR TO BE PUT OUT.
FD91 DF 1E          STX      BUFPTR    SAVES SOURCE POINTER.
FD93 BD FCAD        JSR      PUTCHR    PRINTS CHARACTER.
FD96 DE 1E          LDX      BUFPTR    RESTORES POINTER.
FD98 9C 0A          CPX      OUTEND    TESTS FOR END-OF-TEXT.
FD9A 27 03          BEQ      OUT2      EXITS IF END OF TEXT.
FD9C 08             INX                INCRE. PTR TO NEXT CHAR.
FD9D 20 F0          BRA      OUT1      GOES BACK FOR NEXT CHAR.
FD9F 39      OUT2   RTS                EXITS ROUTINE.
FDA0 01             NOP                UNUSED LOCATION.
                        *
                        *       END OF EDITOR PROGRAM.
                        *
```

```
*             THE MINI-ASSEMBLER
*
*
*     THE MINI-ASSEMBLER IS A FIXED-FIELD ONE INSTRUCTION
* PER LINE 2 PASS ASSEMBLER.  THE MINI-ASSSEMBLER FORMAT
* IS DESCRIBED ON PAGES 9-2 AND 9-3 OF THE SPHERE
* OPERATORS REFERENCE MANUAL.
*     THE TWO PASSES ARE REQUIRED TO FORM THE LABEL
* ADDRESSES.  THE SECOND PASS EQUATES THE ADDRESS FOR
* LABLES REFRENCED BEFORE THEY ARE DEFINED IN THE PROGRAM.
*
*
* ON ENTRY:
* SRCASM = ADDRESS OF SOURCE TEXT TO BE ASSEMBLED.
* BUFFLO = ADDRESSED OF OBJECT CODE PRODUCED.
*
* ON EXIT:
* PCVAL (PROGRAM COUNTER VALUE) = LAST LOCATION OF
* THE ASSEMBLED OBJECT PROGRAM.
*
*
*
*
* ALGORITHM:
*
*ASMBLR:  SET PASS COUNT TO ZERO; SET PCVAL TO DSTASM;
*ASM1A:  OPERAND VALUE FORMED IN "ONDVAL":
* A GETS CHAR IN X6 (OPERAND TYPE); X GETS X+7;
* IF CHAR X6 IS A "@" THEN ONDVAL GETS VALUE FROM SYMBOL
*   TABLE ELSE ONDVAL GETS VALUE FROM ASCBIN CONVERSION;
*SYMBL:  EQUATES SYMBOL (PC VALUE IS THE " " SYMBOL
*[LABEL]) TO A LABEL VALUE:
* SYMVAL GETS PCVAL;
* IF X(1) IS AN "=" THEN SYMVAL GETS ONDVAL;
* IF X(1) IS NOT A "=" OR A SPACE THEN IF SECOND PASS THEN
*   EXIT ELSE START SECOND PASS;
* LABEL ENTRY IN SYMBOL TABLE GETS SYMVAL;
*LDOP:  PUT OPERATION CODE INTO THE OBJECT CODE:
* CONVERT X(2)-X(3) INTO BINARY;
* SAVE PCVAL;
* P. C.  GETS P. C. +1;
*OPRND:  FORM OPERAND IN OBJECT CODE:
* FORM ONDVAL INTO PROPER SIZE BASED ON CODE IN X(6);
* STORE NEW OPERAND VALUE IN MEMORY;
* P. C.  GETS P. C. +1 OR 2;
* GET NEXT LINE OF SOURCE;
* GO TO ASM1A;
*
*
```

```
FDA1 7F 0004 ASMBLR CLR     AR3       INIT. PASS CTR TO FRST PASS. )
FDA4 DE 20   ASM1   LDX     BUFFLO    SETS PC CNTR TO START OF
FDA6 DF 40          STX     PCVAL     OBJECT CODE.
FDA8 DE 26          LDX     SRCASM    LOADS ADDR FOR FIRST LINE.
FDAA DF 02   ASM1A  STX     TMP1      SAVES ADDR OF CURRENT LINE.
FDAC A6 08          LDA A   8,X       LOADS SYMBOL (LABEL).
FDAE E6 07          LDA B   7,X       LOADS OPERAND TYPE CODE.
FDB0 C1 40          CMP B   #$'@      IF @, LOADS DATA IN SYMBOL
FDB2 27 6B          BEQ     INDADR    ADDRESS, GOES TO SYMBL.
FDB4 08             INX               SETS INDEX TO START OF
FDB5 08             INX               OPERAND NUMBER.
FDB6 08             INX
FDB7 08             INX
FDB8 08             INX
FDB9 08             INX
FDBA 08             INX
FDBB BD FF22        JSR     ASCBIN    CONVRTS # TO BINARY IN B-A.
FDBE D7 2A   ASM1B  STA B   ONDVAL    STORES OPERAND VALUE IN
FDC0 97 2B          STA A   ONDVAL+1  ONDVAL.
             *
             *
             *
             * FOLLOWING FORMS THE VALUE FOR THE LABEL.
             *
FDC2 DE 02   SYMBL  LDX     TMP1      LOADS ORIG LINE PTR INTO X.
FDC4 A6 00          LDA A   0,X       LOADS SYMBOL (LABEL).
FDC6 E6 01          LDA B   1,X       LOADS LABEL CONTROL CHAR.
FDC8 DE 40          LDX     PCVAL     LABEL VALUE GETS PCVAL.
FDCA DF 2C          STX     SYMVAL
FDCC C1 3D          CMP B   #$'=      TESTS IF LABL IS EQUATED.
FDCE 26 06          BNE     ASM2      SKIPS IF NOT EQUATED.
FDD0 DE 2A          LDX     ONDVAL    LABEL VALUE (SYMVAL) GETS
FDD2 DF 2C          STX     SYMVAL    THE OPERAND VALUE.
FDD4 20 0E          BRA     ASM3      CONTINUES EVALUATION.
FDD6 C1 20   ASM2   CMP B   #$'       TESTS FOR END-OF-PROGRAM.
FDD8 27 0A          BEQ     ASM3      SKIPS IF SPACE (NOT END).
FDDA 7D 0004        TST     AR3       TESTS IF SECOND PASS.
FDDD 27 01          BEQ     ASM2A     EXITS IF SECOND PASS.
FDDF 39             RTS
FDE0 D7 04   ASM2A  STA B   AR3       SETS CTR TO SECOND PASS.
FDE2 20 C0          BRA     ASM1      GOES BACK FOR SECOND PASS.
             *
             *
             *
             * FOLLOWING PUTS THE LABLE VALUE IN THE SYMBOL TABLE.
             *
FDE4 8D 41   ASM3   BSR     SYMPTR    X GETS SYMBL TABL ENTRY ADR. E
FDE6 96 2C          LDA A   SYMVAL    STORES THE LABEL
FDE8 A7 00          STA A   0,X       ADDRESS (SYMVAL) INTO THE
FDEA 96 2D          LDA A   SYMVAL+1  SYMBOL TABEL.
FDEC A7 01          STA A   1,X
```

```
                * FOLLOWING FORMS THE OPERATION CODE.
                *
FDEE DE 02      LDOP   LDX     TMP1      LOADS ORIG LINE POINTER.
FDF0 08                INX               SETS X TO POINT TO
FDF1 08                INX               THE OP CODE CHARS.
FDF2 08                INX
FDF3 A6 00             LDA A   0,X       GETS OP CODE CHAR INTO A.
FDF5 81 20             CMP A   #$'       TESTS IF OP CODE EXISTS.
FDF7 27 0A             BEQ     OPRND     SKIPS IF NONEXISTANT.
FDF9 BD FF22           JSR     ASCBIN    CONVRTS OP CODE TO BINARY.
FDFC DE 40             LDX     PCVAL     LOADS POINTR TO OBJECT CODE.
FDFE A7 00             STA A   0,X       STORS OP INTO OBJECT CODE.
FE00 08                INX               SETS TO NEXT OBJ CODE LOCTN.
FE01 DF 40             STX     PCVAL     SAVES P.C. POINTER.
                *
                *
                *
                * FOLLOWING STORES INTO THE OBJECT CODE THE SIZED OPERAND.
                *
FE03 DE 02      OPRND  LDX     TMP1      LOADS SOURCE LINE POINTER.
FE05 A6 06             LDA A   6,X       LOADS OPERAND SIZE CHAR.
FE07 DE 40             LDX     PCVAL     LOADS X WITH OBJ CODE PTR.
FE09 81 45             CMP A   #$'E      TESTS LENGTH TYPE.
FE0B 2E 31             BGT     RELTIV    SKIPS IF AN "R" OPERAND.
FE0D 27 21             BEQ     EXTEND    SKIPS IF AN "E" SIZE OPRND.
FE0F 81 44             CMP A   #$'D      TESTS IF SIZE CHR EXISTS.
FE11 27 22             BEQ     DIRECT    SKIPS IF "D"COMND EXISTS.
                *
                *
                *
                * FOLLOWING GETS THE NEXT LINE.
                *
FE13 DE 02      ASM4   LDX     TMP1      LOADS START OF LINE IN
FE15 08         ASM4A  INX               ORDER TO FIND NEXT LINE.
FE16 A6 00             LDA A   0,X       LOADS CHAR FROM SORCE LINE.
FE18 81 60             CMP A   #$60      TESTS FOR A CARRAGE RETURN.
FE1A 26 F9             BNE     ASM4A     SKIPS BAK UNTIL C.R. FOUND.
FE1C 08                INX               POINTS TO FIRST LINE CHAR.
FE1D 20 8B             BRA     ASM1A     GOES BACK TO ASSM. NEXT LINE
                *
                *
                *
                * THE FOLLOWING ARE SUBROUTINES USED BY THE MAIN CODE.
                *
                *
FE1F 8D 06      INDADR BSR     SYMPTR    GETS CONTENTS OF
FE21 EE 00             LDX     0,X       SYMBOL LOCATION.
FE23 DF 2A             STX     ONDVAL    STORES AS OPERAND.
FE25 20 9B             BRA     SYMBL     RETURNS TO FIX LABEL VALUE.
                *
FE27 48         SYMPTR ASL A   -         MULT LABEL BY 2 TO FORM
FE28 5F                CLR B   -         POINTR INTO SYMBOL TABLE.
FE29 97 01      LOADX  STA A   TMP+1     LOADS POINTER INTO THE
FE2B D7 00             STA B   TMP       SYMBOL TABLE INTO X.
FE2D DE 00             LDX     TMP       RETURNS TO CALLER.
FE2F 39                RTS               RETURNS.
0
```

```
FE30 D6 2A    EXTEND LDA B  ONDVAL    STORES HI BYTE OF OPERAND
FE32 E7 00           STA B  0,X       INTO OBJECT CODE.
FE34 08              INX              INC PC TO POINT TO NXT WD.
FE35 96 2B    DIRECT LDA A  ONDVAL+1  STORES LO BYTE OF OPERAND
FE37 A7 00           STA A  0,X       INTO OBJECT CODE.
FE39 08              INX              INC & SAVE P.C. TO POINT TO
FE3A DF 40           STX    PCVAL     NEXT BYTE.
FE3C 20 D5           BRA    ASM4      GOES TO WORK ON NEXT LINE.
              *
FE3E 08       RELTIV INX              INCRIMENT P.C. PTR TO POINT
FE3F DF 40           STX    PCVAL     TO NXT BYTE & SAVE P.C..
FE41 96 2B           LDA A  ONDVAL+1  LOADS LO BYTE OF OPERAND.
FE43 90 41           SUB A  PCVAL+1   FORMS RELATIVE OFFSET.
FE45 09              DEX              INSERTS RELATIVE BYTE INTO
FE46 A7 00           STA A  0,X       OBJECT CODE.
FE48 20 C9           BRA    ASM4      GOES TO ASSMBL NEXT LINE.
              *
              *      END OF THE ASSEMBLER PROGRAM.
              *
```

```
*             DEBUGGER
*
*     THE DEBUGGER FOR THE PDS SYSTEM WAS DESIGNED TO
* PROVIDE A VERSATILE TOOL FOR USE IN PROGRAM TESTING AND
* DEBUGGING.  IT ALLOWS FOR BREAKPOINTS, MINI-ASSEMBLER
* SYMBOL TABLE REFERENCING, STACK MANIPULATION AND
* INPUT IN EITHER HEXADECIMAL, OCTAL OR DECIMAL.
* THE DEBUGGER PRINTS A PROMT CHARACTER ">" ON EVERY NEW
* LINE.  AN INSTRUCTION CAN BE TYPED IN WHENEVER THE CURRSOR
* IS BLINKING, EXCEPT WHEN A NUMBER IS BEING TYPED IN.
* THE DEBUGGER CALLS THE EDITOR WHENEVER A NUMBER IS TO BE
* INPUT, SO CORRECTIONS CAN BE MADE IF THE WRONG DIGIT IS
* TYPED IN.  THE POINTER "PCVAL" POINTS TO THE CURRENTLY
* OPENED BYTE LOCATION.  THE DEBUGGER OPERATES ON WHATEVER
* BYTE IS POINTED TO BY PCVAL.  FOR FURTHER DETAILS SEE THE
* SECTION ON THE DEBUGGER IN THE OPERATORS REFERENCE MANUAL.
*
*     THE DEBUGGER IS IMPLEMENTED BY A SMALL ROUTINE TO SET
* UP ENTRY (DEBUG) AND A LARGE ROUTINE WHICH DOES A RANGE
* COMPARE TO FIND THE PROPER COMMAND AND THEN EXECUTES THE
* COMMAND (RUNBUG).  NOTE THAT SINCE COMMANDS ARE
* INTERPRETED BY RANGE, ANY KEY STRUCK WILL PRODUCE
* A COMMAND EXECUTION, SUCH AS A "," BEING INTERPRETED
* AS A "+" COMMAND.
*
*
*
* COMMANDS:
*
* "C.R." LINE - PRINTS ">" OUT ON A NEW LINE.
* " " CHANGE - THE SPACE COMND CHANGES CONTENTS FROM Y TO Z.
* "+" OPNNXT - OPENS NEXT LOCATION.
* "-" OPNPRE - OPENS PREVIOUS LOCATION.
* "B" BRKSET - SETS A BREAKPOINT AT THE OPENED LOCATION.
* "C" CLRBRK - CLEARS BRKPOINT.  MUST BE DONE BEFORE EXIT.
* "E" EXIT - PERFORM RTI - EXECUTE AT BRKPOINT LOCATION.
* "G" GOLOCN - STARTS EXECUTION AT OPENED LOCATION.
* "O" OPNLOC - OPENS LOCATION THAT IS TYPED IN AFTER "O".
* "R" OPNREG - OPENS THE TOP-OF-STACK LOCATION.
* "S" SETSTK - SETS THE STACK TO THE OPENED LOCATION.
* "T" OPNTBL - OPENS LOCATION IN SYMBOL TABLE OF NEXT CHR.
*
*
*
*
*
* SUBROUTINES:
*
* INPCHR - INPUTS A CHAR. INTO A AND PRINTS IT.
* INPNUM - INPUTS A NUMBER INTO B-A FROM THE KEYBOARD.
* PNTBYT - PRINTS ACC A AS 2 HEX DIGITS ON THE SCREEN.
* PNTDIG - PRINTS B-A AS 4 HEX DIGITS ON THE SCREEN.
* NEWLIN - PRINTS A C.R. AND A ">" ON THE SCREEN.
* DSPADR - PRINTS BYTE ADDRESS (XXXX) AND BYTE CONTENTS (YY)
*   AS >XXXX YY ON THE SCREEN.
*
```

0

```
FE4A                    ORG     $FE4A
                 *
FE4A BD FCBE BGPOP1 JSR     SUB32       MOVES CURSOR UP ONE LINE.
FE4D 31          POPLIN INS             CLEANS UP STACK FOR
FE4E 31                 INS             DISPLAY OF C. R. >.
                 *
                 *
FE4F 8D 74       DEBUG  BSR     NEWLIN  PRINTS "C. R. >".
FE51 8D 09       DBUG1  BSR     INPCHR  READS IN COMMAND.
FE53 BD FC96            JSR     RTARRO  INSERT BLANK.
FE56 DE 40              LDX     PCVAL   LOADS CURRENTLY OPENED LOC.
FE58 8D 09              BSR     RUNBUG  EXECUTES DEBUG COMMAND.
FE5A 20 F5              BRA     DBUG1   GOES BACK FOR NEXT COMND.
                 *
FE5C BD FC45 INPCHR JSR     GETCHR      READ IN CHAR INTO A.
FE5F BD FCAD            JSR     PUTCHR  DISPLAYS CHARACTER.
FE62 39                 RTS             RETURNS TO CALLER.
                 *
                 *
                 *
                 *
FE63 81 20       RUNBUG CMP A   #$'     TESTS FOR SPACE.
FE65 2D E3              BLT     BGPOP1  SKIPS IF A "C. R. " (LINE).
FE67 2E 09              BGT     OPNPRE  SKIPS FOR OTHER CMND TESTS.
FE69 BD FEF4 CHANGE JSR     INPNUM      INPUTS NEW BYTE CONTENTS.
FE6C DE 40              LDX     PCVAL   LOADS OPENED BYTE LOCATION.
FE6E A7 00              STA A   0,X     STORES NEW BYTE DATA.
FE70 20 DB              BRA     POPLIN  GOES TO INPUT NEXT COMMAND.
                 *
                 *
FE72 81 2D       OPNPRE CMP A   #$'-    TESTS FOR A "-" COMMAND.
FE74 2D 05              BLT     OPNNXT  SKIPS IF A "+" COMMAND.
FE76 2E 06              BGT     CLRBRK  SKIPS TO TEST FOR OTHER CMND
FE78 09                 DEX             FORMS PREV. LOCATION ADDR.
FE79 20 2B              BRA     DSPADR  GOES TO OPEN THE LOCATION.
                 *
                 *
FE7B 08          OPNNXT INX             FORMS NEXT LOCATION ADDR.
FE7C 20 28              BRA     DSPADR  GOES TO OPEN LOCATN BYTE.
                 *
                 *
FE7E 81 43       CLRBRK CMP A   #$'C    TESTS FOR A "C" COMMAND.
FE80 2D 0A              BLT     BRKSET  SKIPS IF A "B" COMMAND.
FE82 2E 78              BGT     EXIT    SKIPS FOR NEXT COMND TEST.
FE84 DE 30              LDX     BRKADR  GETS ADDRESS OF BREAKPOINT.
FE86 96 2E              LDA A   BRKSAV  LOADS ORIG BYTE CONTENTS.
FE88 A7 00              STA A   0,X     RESTORES BYTE DATA.
FE8A 20 1A              BRA     DSPADR  GOES TO OPEN THE LOCATION.
                 *
                 *
FE8C A6 00       BRKSET LDA A   0,X     LOADS DATA OF OPNED LOCATN.
FE8E 97 2E              STA A   BRKSAV  SAVES DATA OF OPNED BYTE.
FE90 DF 30              STX     BRKADR  SAVES ADDR. OF BREAKPOINT.
FE92 86 3F              LDA A   #$3F    LOADS SOFTWARE INTUP COMND.
FE94 A7 00              STA A   0,X     SETS AN SWI AT OPNED BYTE.
FE96 20 B5              BRA     POPLIN  GOES TO NEXT LINE FOR COMND.
```

```
FE98 30        OPNREG TSX             OPENS TOP-OF-STACK.
FE99 08               INX             PCVAL GETS STACK POINTER.
FE9A 08               INX             (CLEANS UP THE STACK).
FE9B 20 09            BRA    DSPADR   GOES TO DISPLAY THE T-O-S.
               *
               *
FE9D 81 53     SETSTK CMP A  #$'S     TESTS FOR AN "S" COMMAND.
FE9F 2D F7            BLT    OPNREG   SKIPS IF AN "R" COMMAND.
FEA1 2E 13            BGT    OPNTBL   SKIPS FOR THE "T" COMMAND.
FEA3 35               TXS             STACK POINTER GETS PCVAL.
FEA4 20 A9            BRA    DEBUG    RETURNS TO INPUT COMMAND.
               *
               *
               *
               *
               * FOLLOWING ARE SUBROUTINES USED BY DEBUGGER.
               *
               *
               * FOLLOWING DISPLAYS THE LOCATION ADDR. & CONTENTS.
               *
FEA6 DF 40     DSPADR STX    PCVAL    SAVES OPENED LOCATION ADDR.
FEA8 8D 1B     DSPAD1 BSR    NEWLIN   PRINTS A "C. R. " AND ">".
FEAA 8D 10            BSR    PNTDIG   PRINTS OUT "PCVAL" IN HEX.
FEAC BD FC96          JSR    RTARRO   PRINTS A SPACE.
FEAF DE 40            LDX    PCVAL    LOADS PTR. TO OPEND LOC.
FEB1 A6 00            LDA A  0,X      LOADS DATA FROM LOCATN.
FEB3 8D 17            BSR    PNTBYT   PRINTS DATA IN HEX FORMAT.
FEB5 39               RTS             RETURNS TO INPUT COMMAND.
               *
               *
FEB6 8D A4     OPNTBL BSR    INPCHR   LOADS A WITH SYMBOL (LABL).
FEB8 48               ASL A  -        ALIGNS ADDRESS FOR
FEB9 5F               CLR B  -        SYMBOL TABLE ENTRY.
FEBA 20 5C            BRA    OPNLC1   SAVES AND DISPLAYS ADDRESS.
               *
               *
FEBC 96 40     PNTDIG LDA A  PCVAL    PRINTS THE TWO HI HEX DIGITS
FEBE 8D 0C            BSR    PNTBYT   OF OPENED ADDRESS.
FEC0 96 41            LDA A  PCVAL+1  PRINTS OUT 2 LOW HEX DIGITS
FEC2 8D 08            BSR    PNTBYT   OF THE OPENED ADDRESS.
FEC4 39               RTS             RETURNS TO CALLING CODE.
               *
               *
FEC5 CE 0D3E   NEWLIN LDX    #$0D3E   LOADS "C. R. " & ">".
FEC8 DF 35            STX    IOBUFF   STORES FOR PRINTING.
FECA 20 1A            BRA    PNTBUF   SKIPS TO PRINT CHAR.
               *
               *
               * FOLLOWING PRINTS OUT 2 HEX DIGITS.
               *
FECC CE 0010   PNTBYT LDX    #16      LOADS 16 FOR BASE.
FECF DF 04            STX    ARB      STORES FOR CONVERSION.
FED1 5F               CLR B  -        CLEARS HI 2 DIGITS.
FED2 CE 0035          LDX    #IOBUFF  LOADS OUTPUT BUFF ADDRESS.
```

```
                  * FOLLOWING CONVERTS BYTE TO HEX WITH LEADING ZEROS.
                  *
FED5 D7 36        CONVRT STA B  IOBUFF+1 SETS BYTE FOR SECOND DIGIT.
FED7 BD FF64             JSR    BINASC   CONVERTS TO ASCII DIGITS.
FEDA 96 36               LDA A  IOBUFF+1 TESTS BOTH DIGITS CONVTD.
FEDC 26 08               BNE    PNTBUF   SKIPS IF BOTH DIGITS NOT 0.
FEDE 96 35               LDA A  IOBUFF   MOVES LOW DIGIT.
FEE0 97 36               STA A  IOBUFF+1
FEE2 86 30               LDA A  #$'0     HIGH DIGIT GETS A "0".
FEE4 97 35               STA A  IOBUFFF
                  *
FEE6 CE 0035      PNTBUF LDX    #IOBUFF  LOADS ADDR OF HEX DIGITS.
FEE9 DF 11               STX    OUTBUF   STORES AS OUTPUT BUFFER.
FEEB CE 0036             LDX    #IOBUFF+1  STORES ADDR. OF
FEEE DF 0A               STX    OUTEND   END OF DIGITS PRINTED.
FEF0 BD FD8D             JSR    OUTSTR   PRINTS OUT BYTES.
FEF3 39                  RTS             RETURNS TO CALLING PROGRAM.
                  *
                  *
                  *
                  * FOLLOWING INPUTS A 16 BIT NUMBER INTO THE BA REG.
                  *
FEF4 BD FC6E      INPNUM JSR    EDITIN   INPUTS A STRING OF DIGITS.
FEF7 DE 24               LDX    SCNPTR   LOADS ADDR OF FIRST DIGIT.
FEF9 8D 27               BSR    ASCBIN   CONVERTS TO BINARY IN BA.
FEFB 39                  RTS             RETURNS TO CALLER.
                  *
                  *
FEFC 81 47        EXIT   CMP A  #$'G     TESTS IF AN EXIT (E) COMMD.
FEFE 27 1E               BEQ    GOLOCN   SKIPS TO THE "GO" COMMAND.
FF00 2C 10               BGE    OPNLOC   SKIPS FOR NEXT COMMD TEST.
FF02 31                  INS             CLEARS UP THE STACK.
FF03 31                  INS
FF04 30                  TSX             INDEX GETS STACK POINTER.
FF05 E6 05               LDA B  5,X      LOADS RETURN ADDRESS.
FF07 A6 06               LDA A  6,X
FF09 80 01               SUB A  #1       SUB 1 FROM RETURN ADDR.
FF0B C2 00               SBC B  #0
FF0D E7 05               STA B  5,X      RESTORES ADDR. TO BRKPT
FF0F A7 06               STA A  6,X      LOCATION.
FF11 3B                  RTI             RETURNS FROM BREAKPOINT.
                  *
                  *
FF12 81 4F        OPNLOC CMP A  #'O      TESTS FOR AN "O" COMMAND.
FF14 2E 87               BGT    SETSTK   SKIPS IF AN "R", "S" ETC.
FF16 8D DC               BSR    INPNUM   LOADS A 16 BIT NUMBER.
FF18 D7 40        OPNLC1 STA B  PCVAL    STORES NEW OPENED
FF1A 97 41               STA A  PCVAL+1  LOCATION ADDRESS.
FF1C 20 8A               BRA    DSPAD1   DISPLAYS CONTENTS OF LOC.
                  *
                  *
FF1E 31           GOLOCN INS             CLEANS UP THE STACK.
FF1F 31                  INS
FF20 6E 00               JMP    0,X      JUMPS TO USERS PROGRAM.
                  *
                  *      END OF DEBUGGER
                  *
0
```

```
*       UTILITY PROGRAMS
*
*
*
*
*           ASCII TO BINARY CONVERSION.
*
*     THE ASCII TO BINARY ROUTINE CONVERTS FROM AN ASCII
* NUMBER STRING POINTED TO BY X TO AN UNSIGNED 16 BIT
* BINARY NUMBER IN BA (ACC B HAS THE HI BYTE, ACC A HAS
* THE LO BYTE).  THE ASCII STRING IS TERMINATED BY A NON
* HEXADECIMAL CHARACTER.  UPON EXITING, THE INDEX REGISTER
* WILL POINT TO THE NEXT CHARACTER AFTER THE NUMBER
* STRING.  THE BASE OF THE NUMBER STRING IS PASSED TO
* THE ROUTINE IN ARA (ARA IS THE ARITHMETIC REGESTER A
* LOCATED IN BYTES 06 AND 07 OF LOW MEMORY).  IF THE
* ROUTINE IS ENTERED WITH A KNOWN BASE, PUT THE BASE
* (BETWEEN 2 AND 16) IN ARA AND ENTER THE ROUTINE AT
* THE ENTRY POINT ENTR2.
*
*
*
*     CONVERSION FORMULA:
* ASCII NUMBBER STRING X[4],X[3],X[2],X[1] IN
*BASE Y;
* BINARY NUMBER =
*     X[4]*Y↑3+X[3]*Y↑2+X[2]*Y↑1+X[1]*Y↑0        OR
* BINARY NUMBER =
*     (((0*Y+X[4])*Y+X[3])*Y+X[2])*Y+X[1]
*WHERE ↑ IS THE EXPONENT OPERATOR,
* X IS A CHARACTER & Y IS THE BASE.
*
*
*
*     ALGORITHM:
*ASCBIN:  FORM THE BASE IN ARA BASED ON THE FIRST CHAR.
*  OF THE NUMBER STRING; INCREMENT CHAR. PTR. IN X;
*ENTR2:  NUMBER (IN BA) GETS 0;
*NXTCHR:  IF THE CURRENT CHAR. POINTED TO BY X IS NOT A
*  DIGIT THEN EXIT ELSE INCREMT CHARACTER PTR IN INDEX;
*  CONVERT DIGIT TO BINARY;
*  NUMBER GETS NUMBER * BASE;
*  NUMBER GETS NUMBER + DIGIT;
*  GO TO OPERATE ON THE NEXT DIGIT (NXTCHR);
*
```

```
FF22 A6 00    ASCBIN LDA A  0,X       GETS CHR TO FORM BASE.
FF24 81 2E           CMP A  #$'.      TESTS FOR DECML STRNG.
FF26 2D 06           BLT    OCT       SKIPS IF BASE 8 (*).
FF28 2E 09           BGT    HEX       SKIPS IF BASE 16.
FF2A 86 0A           LDA A  #10       LOADS BASE 10 FOR CONVERSN.
FF2C 20 02           BRA    ASC1      SKIPS TO INC. TEXT POINTR.
FF2E 86 08    OCT    LDA A  #8        LOADS BASE 8 FOR CONVERSION.
FF30 08       ASC1   INX              INCREMENT PTR TO NEXT CHAR.
FF31 20 02           BRA    ASC2      SKIPS TO SAVE BASE.
FF33 86 10    HEX    LDA A  #16       LOADS BASE 16 FOR CONVERN.
FF35 97 07    ASC2   STA A  AR0       SAVES BASE IN BASE#.
              *
              *
FF37 5F       ENTR2  CLR B  NUMBER    GETS 0.
FF38 37              PSH B  -         (LOW NUMBER ON STACK).
FF39 D7 06           STA B  AR1       CLEARS HI OF BASE.
FF3B A6 00    NXTCHR LDA A  0,X       GETS CHAR TO CONVERT.
FF3D 08              INX              INC TO NEXT CHARACTER.
FF3E 81 30           CMP A  #$'0      TESTS FOR END-OF-STRING.
FF40 2D 20           BLT    AEXIT     EXITS IF END.
FF42 80 30           SUB A  #$'0      FORMS B. C. D. NUMBER.
FF44 81 0A           CMP A  #10       TESTS IF DECIMAL DIGIT.
FF46 2D 0A           BLT    ASC3      SKIPS IF DECIMAL.
FF48 81 10           CMP A  #16       TESTS FOR END OF STRING.
FF4A 2F 16           BLE    AEXIT     EXITS IF NOT A HEX DIGIT.
FF4C 80 07           SUB A  #7        FORMS A HEX B. C. D. DIGIT.
FF4E 81 10           CMP A  #16       TESTS FOR END-OF-STRING.
FF50 2C 10           BGE    AEXIT     EXITS IF CHAR > "F".
FF52 97 08    ASC3   STA A  DIGIT     SAVES DIGIT FOR ADD.
              *
FF54 DF 00    CNVASC STX    TMP       SAVES INDEX REG FOR MULT.
FF56 32              PUL A  -         RESTORES LO OF "NUMBER".
FF57 8D 3A           BSR    MULT      NUMBER GETS NUMBER * BASE.
FF59 9B 08           ADD A  DIGIT     NUMBER GETS NUMBER + DIGIT.
FF5B C9 00           ADC B  #0
FF5D 36              PSH A  -         SAVES LO OF NUMBER.
FF5E DE 00           LDX    TMP       RESTORES STRING POINTER.
FF60 20 D9           BRA    NXTCHR    GOES TO CONVRT NEXT CHAR.
FF62 32       AEXIT  PUL A  -         RESTORES "NUMBER" IN BA.
FF63 39              RTS              RETURNS TO CALLING PROGRAM.
```

```
                    *           BINARY TO ASCII
                    *
                    *
                    *     THE BINARY TO ASCII CONVERSION ROUTINE CONVERTS
                    * A 16 BIT BINARY NUMBER IN THE BA REGISTR (REG B & REG A)
                    * TO A STRING OF ASCII DIGITS.  THE ASCII STRING CAN BE IN
                    * ANY BBASE FROM BASE 2 THROUGH BASE 41.  THE VALUE OF THE
                    * BASE IS LOCATED IN THE ARITHMETIC PSEUDO-REGISTER ARB
                    * (ARB IS LOOCATED IN BYTE AR3 [LOC 4] AND AR2 [LOC 5]).
                    * WHEN THE ROUTINE IS ENTERED, THE POINTER TO THE OUTPUT
                    * LOCATION IS PASSED IN THE INDEX REG.  WHEN THE ROUTINE
                    * EXITS, THE INDEX POINTS TO THE LAST DIGIT IN THE STRING
                    * PLUS ONE.
                    *     CONVERSION IS DONE BY THE METHOD OF REPEATED
                    * DIVISION.  THE LOW ORDER DIGIT IS FORMED FIRST.  THE
                    * DIGITS ARE THEN PLACED ON THE STACK UNTIL CONVERSION IS
                    * COMPLETED.  THE DIGITS ARE THEN POPPED OFF THE STACK
                    * AND PLACED IN THE OUTPUT STRING.  THE TOP-OF-STACK IS
                    * INITIALIZED TO ALL ONES TO TELL WHEN ALL THE DIGITS
                    * HAVE BEEN POPPED OFF THE STACK.  AFTER THE DIVISION, THE
                    * DIGIT (THE REMAINDER OF THE DIVISION OPERATION)
                    * IS LOCATED IN THE AR 0 PART OF ARA (BYTE 7).  WHEN
                    * THE QUOTIENT OF THE DIVISION IS 0, THEN THE CONVERSION
                    * IS COMPLEATED.
                    *
                    *
                    *
FF64 DF 00          BINASC STX    TMP      SAVES OUTPUT POINTER.
FF66 34                    DES             /SETS THE TOP-OF-STACK TO
FF67 30                    TSX             /ALL ONES TO TELL END OF
FF68 6F 00                 CLR    0,X      /CHAR STRING (LAST CHAR IS
FF6A 63 00                 COM    0,X      /PUT ON STACK FIRST).
FF6C DE 04          BIN1   LDX    ARB      RESTORES DEVISOR (BASE).
FF6E DF 06                 STX    ARA
FF70 8D 3D                 BSR    DIVIDE   * QUOTIENT IN BA GETS THE
                    * REMAINDER OF # TO BE CONVERTED; REMAINDER IN ARA GETS
                    * THE LOW ORDER DIGIT.
FF72 97 02                 STA A  TMP1     SAVES A OF BA.
FF74 96 07                 LDA A  AR0      LOAD DIGIT (REMAINDER).
FF76 36                    PSH A  -        STACK DIGIT (REVERSE ORDER). G
FF77 96 02                 LDA A  TMP1     RESTORES A OF BA.
FF79 4D                    TST A  -        /TESTS IF QUOTIENT IS = 0
FF7A 26 F0                 BNE    BIN1     /(SIGNIFYING THAT
FF7C 5D                    TST B  -        /THE CONVERRSION
FF7D 26 ED                 BNE    BIN1     /IS DONE).
                    *
FF7F DE 00          BINSTR LDX    TMP      RESTORES OUTPUT POINTER.
FF81 32             BIN3   PUL A  -        UNSTACK A DIGIT.
FF82 4D                    TST A  -        TESTS IF NEG (END?).
FF83 2A 01                 BPL    BIN4     SKIPS IF A DIGIT.
FF85 39                    RTS             EXITS FROM SUBROUTINE.
FF86 81 09          BIN4   CMP A  #9       TESTS IF RESULT IS HEX.
FF88 2F 02                 BLE    BIN5     SKIPS IF DIGIT NOT HEX.
FF8A 8B 07                 ADD A  #7       FORMS HEX VALUE OF DIGIT.
FF8C 8B 30          BIN5   ADD A  #$'0     FORMS DECIMAL CHARACTER.
FF8E A7 00                 STA A  0,X      OUTPUTS CHARACTER.
FF90 08                    INX             POINTS TO NEXT CHARACTER.
FF91 20 EE                 BRA    BIN3     GOES BACK FOR NEXT DIGIT.
```

```
                *           MULTIPLY ROUTINE
                *
                *     THE MULTIPLY ROUTINE MULTIPLIES TWO 16 BIT BINARY
                * NUMBERS TOGETHER TO PRODUCE A 16 BIT RESULT.  THE BA
                * REGISTERS AND ARA (BYTES 6 & 7) REGISTER ARE USED.
                * THE CONTENTS OF ARA ARE UNCHANGED UPON PROGRAM EXIT.
                *
                *
                *  BA GETS BA * ARA
                *
                *
                *     MULTIPLYING IS ACCOMPLISHED BY REPEATED ADDITIONS
                * OF ONE OF THE OPERATORS (OPERATOR ARA) INTO THE RESULT.
                * THE RESULT STARTS OUT WITH A ZERO VALUE AND IS SHIFTED
                * OVER ONE AFTER EACH ADDITION.  THE HIGHEST ORDER VALUE
                * IS ADDED IN FIRST AND TTHEN, GOING TO THE RIGHT,
                * (THUS SHIFTING THE ANSWER LEFT ONE TO BRING IN THE NEXT
                * RIGHTMOST DIGIT) GETTING THE NEXT LOWERMOST SIGNIFICANT
                * DIGIT.  THE NEXT RIGHTMOST BIT OF THE OTHER OPERAND
                * (THE ONE ORIGINALLY IN BA) IS TESTED, AND IF ONE,
                * ANOTHER ADDITION TAKES PLACE.  THIS IS REPEATED UNTIL
                * THE FINAL SUM IS FORMED.
                *
                *
                *
                *  MULTIPLY ALGORITHM:
                *
                *MULT:  STACK BA; BA GETS 0; SET COUNT VALUE TO 16;
                *MUL1:  SHIFT BA LEFT 1;
                *  SHIFT LEFT ORIG BA VALUE ON STACK INTO CARRY;
                *  IF CARRRY = 0 THEN GO TO MUL2
                *  BA GETS BA + ARA;
                *MUL2:  DECREMENT COUNT;
                *  IF COUNT # 0 THEN GO TO MUL1 ELSE EXIT.
                *
                *
                *
                *
FF93 36         MULT   PSH A  -       PUTS THE ORIGINAL CONTENTS
FF94 37                PSH B  -       OF BA ONTO THE STACK.
FF95 86 10             LDA A  #16     LOADS COUNT VALUE
FF97 36                PSH A  -       ONTO THE STACK.
FF98 4F                CLR A  -       BA GETS ZEROED.
FF99 5F                CLR B
FF9A 30                TSX            SET INDEX TO STACK.
FF9B 48         MUL1   ASL A  -       SHIFT LEFT BA.
FF9C 59                ROL B
FF9D 68 02             ASL    2,X     SHIFTS ORIG. BA OPERAND
FF9F 69 01             ROL    1,X     ONE LEFT INTO CARRY.
FFA1 24 04             BCC    MUL2    SKIPS ADDING IF CARRY = 0.
FFA3 9B 07             ADD A  AR0     BA GETS BA + ARA.
FFA5 D9 06             ADC B  AR1
FFA7 6A 00      MUL2   DEC    0,X     TESTS IF DONE.
FFA9 26 F0             BNE    MUL1    GOES BACK IF NOT DONE.
FFAB 31                INS            CLEANS UP THE STACK.
FFAC 31                INS
FFAD 31                INS
FFAE 39                RTS            EXITS ROUTINE.
```

```
                *           DIVIDE ROUTINE
                *
                *      THE DIVIDE SUBROUTINE DIVIDES THE 16 BIT NUMBER
                * IN THE BA REGESTERS BY THE 16 BIT NUMBER IN THE PSEUDO
                * REGESTER ARA (LOCATED IN BYTES 6 & 7).  UPON EXITING,
                * BA WILL CONTAIN THE QUOTIENT OF THE DIVISION AND ARA
                * WILL CONTAIN THE REMAINDER.  THE DIVIDEND BA IS DIVIDED
                * BY THE DIVISOR ARA (I.E.  BA/ARA ).
                *
                *
                *
                *
                *  DIVIDE ALGORITHM:
                *
                *DIVIDE:  X3,4 GETS BA (BA IS PUT ON THE STACK);
                *  X1,2 GETS THE 16 BIT ARA VALUE (ARA PUT ON THE STACK);
                *  COUNT GETS 1 + THE NUMBER OF NONSIGNIFICANT BITS IN
                * THE DIVISOR [LEFT JUSTIFY X1,2 TO FIRST 1 BIT, COUNT
                * GETS 1 + THE # OF LEADING ZEROS IN X1,2] [COUNT WILL BE
                * FROM 1 TO 17];
                *  BA GETS X3,4 [RESTORES BA];
                *  X3,4 GETS 0 [INITIALIZES THE QUOTIENT];
                *DIV3:  BA GETS BA - X1,2;
                *  IF THERE WAS A BORROW [I.E. DIVIDEND IN BA < DIVISOR
                * IN X1,2 (ARA)] THEN BA GETS BA + X1,2 [ORIGINAL BA
                * VALUE RESTORED] & CARRY CLEARED ELSE CARRY IS SET;
                *DIV5:  X3,4 [QUOTIENT] GETS CARRY LEFT SHIFTED IN;
                *  X1,2 [DIVISOR] GETS SHIFTED RIGHT ONE PLACE WITH ZERO
                * FILLED IN FROM THE LEFT SIDE;
                *  DECREMENT COUNT;
                *  EXIT IF DONE ELSE GO TO DIV3;
                *
                *
                *
                *
                *
FFAF 36         DIVIDE PSH A   -       LOADS DIVIDEND INTO X3,4.
FFB0 37                PSH B
FFB1 96 06             LDA A   AR1     LOADS DIVISOR FROM ARA.
FFB3 D6 07             LDA B   AR0
FFB5 37                PSH B   -       PUTS DIVISOR INTO X1,2.
FFB6 36                PSH A
FFB7 34                DES             SET UP SPACE FOR COUNT.
FFB8 30                TSX             INDEX GETS STACK POINTER.
FFB9 86 01             LDA A   #1      INITIALIZE COUNT.
FFBB 6D 01             TST     1,X     TESTS FOR HI DIVISR BIT ON.
FFBD 2B 0B             BMI     DIV2    SKIPS IF ON.
FFBF 4C         DIV1   INC A   -       COUNTS LEADING ZEROS.
FFC0 68 02             ASL     2,X     LEFT JUSTIFIES X1,2.
FFC2 69 01             ROL     1,X
FFC4 2B 04             BMI     DIV2    SKIPS IF NO LEADING ZERO.
FFC6 81 11             CMP A   #17     TESTS FOR ALL ZERO DIVISOR.
FFC8 26 F5             BNE     DIV1    GOES BACK IF BITS LEFT.
FFCA A7 00      DIV2   STA A   0,X     SETS COUNTER.
FFCC E6 03             LDA B   3,X     BA GETS ORIGINAL
FFCE A6 04             LDA A   4,X     DIVIDEND VALUE.
FFD0 6F 03             CLR     3,X     CLEARS X3,4 FOR FORMATION
FFD2 6F 04             CLR     4,X     OF THE QUOTIENT.
```

```
FFD4 A0 02   DIV3   SUB A  2,X      START OF DIVIDE LOOP.
FFD6 E2 01          SBC B  1,X
FFD8 24 07          BCC    DIV4     SKIP IF DIVIDEND < DIVISOR.
FFDA AB 02          ADD A  2,X      RESTORES DIVIDEND IN BA.
FFDC E9 01          ADC B  1,X
FFDE 0C             CLC             CLEARS THE CARRY.
FFDF 20 01          BRA    DIV5     SKIPS WITH CARRY CLEAR.
FFE1 0D      DIV4   SEC             SETS CARRY TO 1.
FFE2 69 04   DIV5   ROL    4,X      SHIFT CARRY INTO
FFE4 69 03          ROL    3,X      QUOTIENT X3,4
FFE6 64 01          LSR    1,X      SHIFTS DEVISOR X1,2
FFE8 66 02          ROR    2,X      RIGHT ONE.
FFEA 6A 00          DEC    0,X      DECREMENTS COUNTER.
FFEC 26 E6          BNE    DIV3     GOES BACK IF NOT DONE.
FFEE D7 06          STA B  AR1      STORES REMAINDER IN ARA.
FFF0 97 07          STA A  AR0
FFF2 31             INS             CLEANS UP THE STACK.
FFF3 31             INS
FFF4 31             INS
FFF5 33             PUL B  STORES   QUOTIENT IN BA.
FFF6 32             PUL A
FFF7 39             RTS             EXITS ROUTINE.
             *
             *
             *
FFF8 0104    IRQ    FDB    $0104    INTERRUPT REQUEST VECTOR.
FFFA FE4F    SWI    FDB    $FE4F    SOFTWARE INT. VECTOR ADDR.
FFFC 0108    NMI    FDB    $0108    NIN-MASKABLE-INT. VECT.
FFFE FC00    RST    FDB    $FC00    RESTART VECTOR ADDRESS.
             *
             *
             *
             *   END OF PDS SOURCE LISTING.
             *
             *
                    END
TOTAL ERRORS 00000
```

DONE ON PRO-LOG
1702 EPROM PROGRAMER

3-15-77
TUE

E12

```
*
ØØFF

L
FD00 DE 24 8D 34 DF 2Ø 8D 58 CE E1 EØ DF 1C DF 16 D6
  10 1D C4 EØ D7 1D 39 8D BØ 2Ø F5 DE 24 DF 16 DE 2Ø
  20 9C ØC 27 23 Ø9 9C ØC 27 Ø8 Ø9 E6 ØØ C1 6Ø 26 F5
  30 Ø8 DF 2Ø 2Ø Ø3 DE 14 Ø8 E6 ØØ DF 14 DE 16 E7 ØØ
  40 Ø8 DF 16 C1 6Ø 26 EE 39 8D BE 5F 37 E6 ØØ Ø8 C1
  50 6Ø 26 F8 DE 22 E7 ØØ Ø9 DF 22 33 C1 ØØ 26 F6 39
  60 CE EØ ØØ E6 2Ø E7 ØØ Ø8 8C E1 EØ 26 F6 DF 1C BD
  70 FC 38 39 CE E1 DF E6 ØØ E7 2Ø DF 1C Ø9 8C DF FF
  80 26 F4 C6 6Ø Ø8 E7 ØØ 8C EØ 1F 26 F8 39 DE 11 A6
  90 ØØ DF 1E BD FC AD DE 1E 9C ØA 27 Ø3 Ø8 2Ø FØ 39
  A0 Ø1 7F ØØ Ø4 DE 2Ø DF 4Ø DE 26 DF Ø2 A6 Ø8 E6 Ø7
  B0 C1 4Ø 27 6B Ø8 Ø8 Ø8 Ø8 Ø8 Ø8 Ø8 BD FF 22 D7 2A
  C0 97 2B DE Ø2 A6 ØØ E6 Ø1 DE 4Ø DF 2C C1 3D 26 Ø6
  D0 DE 2A DF 2C 2Ø ØE C1 2Ø 27 ØA 7D ØØ Ø4 27 Ø1 39
  E0 D7 Ø4 2Ø CØ 8D 41 96 2C A7 ØØ 96 2D A7 Ø1 DE Ø2
FDF0 Ø8 Ø8 Ø8 A6 ØØ 81 2Ø 27 ØA BD FF 22 DE 4Ø A7 ØØ
     /
```

OK 2nd

E06

```
*
ØØFF

L
FC00 8E Ø1 FF 3Ø DF 26 DF ØC 86 Ø7 B7 FØ Ø1 CE ØF FF
FC10 DF ØE DF 1A 8D 1C 8D 2Ø BD FD 16 BD FE 5C 81 Ø1
FC20 26 Ø3 BD FD A1 81 Ø5 26 Ø2 8D 35 81 Ø4 26 E9 7E
FC30 FE 4F CE EØ ØØ DF 1C 39 C6 6Ø CE E2 ØØ Ø9 E7 ØØ
  40 9C 1C 26 F9 39 DE 1C 63 ØØ CE 26 FØ Ø9 27 F6 7D
  50 FØ Ø1 2A F8 DE 1C A6 ØØ 2A Ø2 63 ØØ B6 FØ ØØ 39
  60 DE ØC DF 2Ø DE ØE DF 22 8D C8 8D CC 97 32 DE 1C
  70 DF 24 8D D1 81 1B 26 Ø4 7F ØØ 32 39 8D Ø2 2Ø F2
  80 81 Ø9 2D 16 2E Ø2 2Ø 7Ø 81 ØD 2D A6 2E Ø2 86 6Ø
  90 81 12 2D 2A 2E Ø8 DE 1C 2Ø 1F 8D 3F 2Ø 6A 81 14
  A0 2D 26 2E Ø3 Ø9 2Ø 13 81 1F 2D 8D 27 62 DE 1C 81
  B0 ØD 27 Ø4 A7 ØØ 81 6Ø 27 5D Ø8 DF 1C 2Ø 12 DE 1C
  C0 C6 2Ø Ø9 5A 26 FC 2Ø Ø6 C6 2Ø Ø8 5A 26 FC DF 1C
  D0 8C EØ ØØ 2C ØB D6 32 27 Ø2 8D 6D BD FD 73 8D 3A
  E0 8C E2 ØØ 2B 12 8D 11 D6 32 27 ØC DE 22 9C ØE 27
  F0 Ø6 8D 44 DE 14 DF 22 39 D6 32 27 ØA DE 2Ø DF 16
     /
```

OK 1st

OVER

E20

```
*
00FF

L
FE00 08 DF 40 DE 02 A6 06 DE 40 81 45 2E 31 27 21 81
  10 44 27 22 DE 02 08 A6 00 81 60 26 F9 08 20 8B 8D
  20 06 EE 00 DF 2A 20 9B 48 5F 97 01 D7 00 DE 00 39
  30 D6 2A E7 00 08 96 2B A7 00 08 DF 40 20 D5 08 DF
  40 40 96 2B 90 41 09 A7 00 20 C9 BD FC BE 31 31 8D
  50 74 8D 09 BD FC 96 DE 40 8D 09 20 F5 BD FC 45 BD
  60 FC AD 39 81 20 2D E3 2E 09 BD FE F4 DE 40 A7 00
  70 20 DB 81 2D 2D 05 2E 06 09 20 2B 08 20 28 81 43
  80 2D 0A 2E 78 DE 30 96 2E A7 00 20 1A A6 00 97 2E
  90 DF 30 86 3F A7 00 20 B5 30 08 08 20 09 81 53 2D
  A0 F7 2E 13 35 20 A9 DF 40 8D 1B 8D 10 BD FC 96 DE
  B0 40 A6 00 8D 17 39 8D A4 48 5F 20 5C 96 40 8D 0C
  C0 96 41 8D 08 39 CE 0D 3E DF 35 20 1A CE 00 10 DF
  D0 04 5F CE 00 35 D7 36 BD FF 64 96 36 26 08 96 35
  E0 97 36 86 30 97 35 CE 00 35 DF 11 CE 00 36 DF 0A
FEF0 BD FD 8D 39 BD FC 6E DE 24 8D 27 39 81 47 27 1E
/
```

OK 3rd

E33

```
*
00FF

L
FF00 2C 10 31 31 30 E6 05 A6 06 80 01 C2 00 E7 05 A7
  10 06 3B 81 4F 2E 87 8D DC D7 40 97 41 20 8A 31 31
  20 6E 00 A6 00 81 2E 2D 06 2E 09 86 0A 20 02 86 08
  30 08 20 02 86 10 97 07 5F 37 D7 06 A6 00 08 81 30
  40 2D 20 80 30 81 0A 2D 0A 81 10 2F 16 80 07 81 10
  50 2C 10 97 08 DF 00 32 8D 3A 9B 08 C9 00 36 DE 00
  60 20 D9 32 39 DF 00 34 30 6F 00 63 00 DE 04 DF 06
  70 8D 3D 97 02 96 07 36 96 02 4D 26 F0 5D 26 ED DE
  80 00 32 4D 2A 01 39 81 09 2F 02 8B 07 8B 30 A7 00
  90 08 20 EE 36 37 86 10 36 4F 5F 30 48 59 68 02 69
  A0 01 24 04 9B 07 D9 06 6A 00 26 F0 31 31 31 39 36
  B0 37 96 06 D6 07 37 36 34 30 86 01 6D 01 2B 0B 4C
  C0 68 02 69 01 2B 04 81 11 26 F5 A7 00 E6 03 A6 04
  D0 6F 03 6F 04 A0 02 E2 01 24 07 AB 02 E9 01 0C 20
  E0 01 0D 69 04 69 03 64 01 66 02 6A 00 26 E6 D7 06
FFF0 97 07 31 31 31 33 32 39 01 04 FE 4F 01 08 FC 00
/
```

OK 4th

FC 1st
FD 2nd

1 JMP INDEXED = 6E
0 JMP EXTENDED = $\boxed{7E}$
0 JSR INDEXED = AD
JSR EXTENDED = $\boxed{BD}$

8 BSR = 8D
10 RTS = 39

*
ØØFF V3A 6-14-77
E2O

L

FE00	Ø8	DF	4Ø	DE	Ø2	A6	Ø6	DE	4Ø	81	45	2E	31	27	21	81
10	44	27	22	DE	Ø2	Ø8	A6	ØØ	81	6Ø	26	F9	Ø8	2Ø	8B	8D
20	Ø6	EE	ØØ	DF	2A	2Ø	9B	48	5F	97	Ø1	D7	ØØ	DE	ØØ	39
30	D6	2A	E7	ØØ	Ø8	96	2B	A7	ØØ	Ø8	DF	4Ø	2Ø	D5	Ø8	DF
40	4Ø	96	2B	9Ø	41	Ø9	A7	ØØ	2Ø	C9	BD	FC	BE	31	31	8D
50	74	8D	Ø9	BD	FC	96	DE	4Ø	8D	Ø9	2Ø	F5	BD	FC	45	BD
60	FC	AD	39	81	2Ø	2D	E3	2E	Ø9	BD	FE	F4	DE	4Ø	A7	ØØ
70	2Ø	DB	81	2D	2D	Ø5	2E	Ø6	Ø9	2Ø	2B	Ø3	2Ø	28	81	43
80	2D	ØA	2E	78	DE	3Ø	96	2E	A7	ØØ	2Ø	1A	A6	ØØ	97	2E
90	DF	3Ø	86	3F	A7	ØØ	2Ø	B5	3Ø	Ø8	Ø8	2Ø	Ø9	81	53	2D
A0	F7	2E	13	35	2Ø	A9	DF	4Ø	8D	1B	8D	1Ø	BD	FC	96	DE
B0	4Ø	A6	ØØ	8D	17	39	8D	A4	48	5F	2Ø	5C	96	4Ø	8D	ØC
C0	96	41	8D	Ø8	39	CE	ØD	3E	DF	35	2Ø	1A	CE	ØØ	1Ø	DF
D0	Ø4	5F	CE	ØØ	35	D7	36	BD	FF	64	96	36	26	Ø8	96	35
E0	97	36	86	3Ø	97	35	CE	ØØ	35	DF	11	CE	ØØ	36	DF	ØA
F0	BD	FD	8D	39	BD	FC	6E	DE	24	8D	27	39	81	47	27	1E

/

END OF ASSEMBLER

START OF DEBUG

*
ØØFF V3A
E33

L

FF00	2C	1Ø	31	31	3Ø	E6	Ø5	A6	Ø6	8Ø	Ø1	C2	ØØ	E7	Ø5	A7
10	Ø6	3B	81	4F	2E	87	8D	DC	D7	4Ø	97	41	2Ø	8A	31	31
20	6E	ØØ	A6	ØØ	81	2E	2D	Ø6	2E	Ø9	86	ØA	2Ø	Ø2	86	Ø8
30	Ø8	2Ø	Ø2	86	1Ø	97	Ø7	5F	37	D7	Ø6	A6	ØØ	Ø8	81	3Ø
40	2D	2Ø	8Ø	3Ø	81	ØA	2D	ØA	81	1Ø	2F	16	8Ø	Ø7	81	1Ø
50	2C	1Ø	97	Ø8	DF	ØØ	32	8D	3A	9B	Ø8	C9	ØØ	36	DE	ØØ
60	2Ø	D9	32	39	DF	ØØ	34	3Ø	6F	ØØ	63	ØØ	DE	Ø4	DF	Ø6
70	8D	3D	97	Ø2	96	Ø7	36	96	Ø2	4D	26	FØ	5D	26	ED	DE
80	ØØ	32	4D	2A	Ø1	39	81	Ø9	2F	Ø2	8B	Ø7	8B	3Ø	A7	ØØ
90	Ø8	2Ø	EE	36	37	86	1Ø	36	4F	5F	3Ø	48	59	68	Ø2	69
A0	Ø1	24	Ø4	9B	Ø7	D9	Ø6	6A	ØØ	26	FØ	31	31	31	39	36
B0	37	96	Ø6	D6	Ø7	37	36	34	3Ø	86	Ø1	6D	Ø1	2B	ØB	4C
C0	68	Ø2	69	Ø1	2B	Ø4	81	11	26	F5	A7	ØØ	E6	Ø3	A6	Ø4
D0	6F	Ø3	6F	Ø4	AØ	Ø2	E2	Ø1	24	Ø7	AB	Ø2	E9	Ø1	ØC	2Ø
E0	Ø1	ØD	69	Ø4	69	Ø3	64	Ø1	66	Ø2	6A	ØØ	26	E6	D7	Ø6
F0	97	Ø7	31	31	31	33	32	39	Ø1	Ø4	FE	4F	Ø1	Ø8	FC	ØØ

/

END OF DEBUG

ASCII TO BINARY CONVERSION

BINARY TO ASCII CONVERT

MULTIPLY

DIVIDE

INTERRUPT VECTORS

IRQ SWI NMI RST

INTERRUPT (DEBUG) NON MASKABLE (COMMAND INTERPRET)

AS PROGRAMMED BY USER

3-15-77
TUE

E12

```
*
ØØFF

L
FDØØ DE 24 8D 34 DF 2Ø 8D 58 CE E1 EØ DF 1C DF 16 D6
  10 1D C4 EØ D7 1D 39 8D BØ 2Ø F5 DE 24 DF 16 DE 2Ø
  20 9C ØC 27 23 Ø9 9C ØC 27 Ø8 Ø9 E6 ØØ C1 6Ø 26 F5
  30 Ø8 DF 2Ø 2Ø Ø3 DE 14 Ø8 E6 ØØ DF 14 DE 16 E7 ØØ
  40 Ø8 DF 16 C1 6Ø 26 EE 39 8D BE 5F 37 E6 ØØ Ø8 C1
  50 6Ø 26 F8 DE 22 E7 ØØ Ø9 DF 22 33 C1 ØØ 26 F6 39
  60 CE EØ ØØ E6 2Ø E7 ØØ Ø8 8C E1 EØ 26 F6 DF 1C BD
  70 FC 38 39 CE E1 DF E6 ØØ E7 2Ø DF 1C Ø9 8C DF FF
  80 26 F4 C6 6Ø Ø8 E7 ØØ 8C EØ 1F 26 F8 39 DE 11 A6
  90 ØØ DF 1E BD FC AD DE 1E 9C ØA 27 Ø3 Ø8 2Ø FØ 39
  A0 Ø1 7F ØØ Ø4 DE 2Ø DF 4Ø DE 26 DF Ø2 A6 Ø8 E6 Ø7
  B0 C1 4Ø 27 6B Ø8 Ø8 Ø8 Ø8 Ø8 Ø8 Ø8 BD FF 22 D7 2A
  C0 97 2B DE Ø2 A6 ØØ E6 Ø1 DE 4Ø DF 2C C1 3D 26 Ø6
  D0 DE 2A DF 2C 2Ø ØE C1 2Ø 27 ØA 7D ØØ Ø4 27 Ø1 39
  E0 D7 Ø4 2Ø CØ 8D 41 96 2C A7 ØØ 96 2D A7 Ø1 DE Ø2
FDF0 Ø8 Ø8 Ø8 A6 ØØ 81 2Ø 27 ØA BD FF 22 DE 4Ø A7 ØØ
/
```

OK 2nd

EDITOR ↑

ASSEMBLER ↓

TO NEXT SHEET

BEGINNING

E06

```
*
ØØFF

L
FCØØ 8E Ø1 FF 3Ø DF 26 DF ØC 86 Ø7 B7 FØ Ø1 CE ØF FF
FC10 DF ØE DF 1A 8D 1C 8D 2Ø BD FD 16 BD FE 5C 81 Ø1
FC20 26 Ø3 BD FD A1 81 Ø5 26 Ø2 8D 35 81 Ø4 26 E9 7E
FC30 FE 4F CE EØ ØØ DF 1C 39 C6 6Ø CE E2 ØØ Ø9 E7 ØØ
  40 9C 1C 26 F9 39 DE 1C 63 ØØ CE 26 FØ Ø9 27 F6 7D
  50 FØ Ø1 2A F8 DE 1C A6 ØØ 2A Ø2 63 ØØ B6 FØ ØØ 39
  60 DE ØC DF 2Ø DE ØE DF 22 8D C8 8D CC 97 32 DE 1C
  70 DF 24 8D D1 81 1B 26 Ø4 7F ØØ 32 39 8D Ø2 2Ø F2
  80 81 Ø9 2D 16 2E Ø2 2Ø 7Ø 81 ØD 2D A6 2E Ø2 86 6Ø
  90 81 12 2D 2A 2E Ø8 DE 1C 2Ø 1F 8D 3F 2Ø 6A 81 14
  A0 2D 26 2E Ø3 Ø9 2Ø 13 81 1F 2D 8D 27 62 DE 1C 81
  B0 ØD 27 Ø4 A7 ØØ 81 6Ø 27 5D Ø8 DF 1C 2Ø 12 DE 1C
  C0 C6 2Ø Ø9 5A 26 FC 2Ø Ø6 C6 2Ø Ø8 5A 26 FC DF 1C
  D0 8C EØ ØØ 2C ØB D6 32 27 Ø2 8D 6D BD FD 73 8D 3A
  E0 8C E2 ØØ 2B 12 8D 11 D6 32 27 ØC DE 22 9C ØE 27
  F0 Ø6 8D 44 DE 14 DF 22 39 D6 32 27 ØA DE 2Ø DF 16
/
```

OK 1st

INITIALIZATION

EXECUTIVE
(COMMAND INTERPRETER)

EDITOR

```
*                V3A 6-14-77
ØØFF                 E2O

L
Ø8 DF 4Ø DE Ø2 A6 Ø6 DE 4Ø 81 45 2E 31 27 21 81
44 27 22 DE Ø2 Ø8 A6 ØØ 81 6Ø 26 F9 Ø8 2Ø 8B 8D
Ø6 EE ØØ DF 2A 2Ø 9B 48 5F 97 Ø1 D7 ØØ DE ØØ 39
D6 2A E7 ØØ Ø8 96 2B A7 ØØ Ø8 DF 4Ø 2Ø D5 Ø8 DF
4Ø 96 2B 9Ø 41 Ø9 A7 ØØ 2Ø C9 BD FC BE 31 31 8D
74 8D Ø9 BD FC 96 DE 4Ø 8D Ø9 2Ø F5 BD FC 45 BD
FC AD 39 81 2Ø 2D E3 2E Ø9 BD FE F4 DE 4Ø A7 ØØ
2Ø DB 81 2D 2D Ø5 2E Ø6 Ø9 2Ø 2B Ø8 2Ø 28 81 43
2D ØA 2E 78 DE 3Ø 96 2E A7 ØØ 2Ø 1A A6 ØØ 97 2E
DF 3Ø 86 3F A7 ØØ 2Ø B5 3Ø Ø8 Ø8 2Ø Ø9 81 53 2D
F7 2E 13 35 2Ø A9 DF 4Ø 8D 1B 8D 1Ø BD FC 96 DE
4Ø A6 ØØ 8D 17 39 8D A4 48 5F 2Ø 5C 96 4Ø 8D ØC
96 41 8D Ø8 39 CE ØD 3E DF 35 2Ø 1A CE ØØ 1Ø DF
Ø4 5F CE ØØ 35 D7 36 BD FF 64 96 36 26 Ø8 96 35
97 36 86 3Ø 97 35 CE ØØ 35 DF 11 CE ØØ 36 DF ØA
BD FD 8D 39 BD FC 6E DE 24 8D 27 39 81 47 27 1E
/
```

```
*                 V3A
ØØFF
                  E33
L
2C 1Ø 31 31 3Ø E6 Ø5 A6 Ø6 8Ø Ø1 C2 ØØ E7 Ø5 A7
Ø6 3B 81 4F 2E 87 8D DC D7 4Ø 97 41 2Ø 8A 31 31
6E ØØ A6 ØØ 81 2E 2D Ø6 2E Ø9 86 ØA 2Ø Ø2 86 Ø8
Ø8 2Ø Ø2 86 1Ø 97 Ø7 5F 37 D7 Ø6 A6 ØØ Ø8 81 3Ø
2D 2Ø 8Ø 3Ø 81 ØA 2D ØA 81 1Ø 2F 16 8Ø Ø7 81 1Ø
2C 1Ø 97 Ø8 DF ØØ 32 8D 3A 9B Ø8 C9 ØØ 36 DE ØØ
2Ø D9 32 39 DF ØØ 34 3Ø 6F ØØ 63 ØØ DE Ø4 DF Ø6
8D 3D 97 Ø2 96 Ø7 36 96 Ø2 4D 26 FØ 5D 26 ED DE
ØØ 32 4D 2A Ø1 39 81 Ø9 2F Ø2 8B Ø7 8B 3Ø A7 ØØ
Ø8 2Ø EE 36 37 86 1Ø 36 4F 5F 3Ø 48 59 68 Ø2 69
Ø1 24 Ø4 9B Ø7 D9 Ø6 6A ØØ 26 FØ 31 31 31 39 36
37 96 Ø6 D6 Ø7 37 36 34 3Ø 86 Ø1 6D Ø1 2B ØB 4C
68 Ø2 69 Ø1 2B Ø4 81 11 26 F5 A7 ØØ E6 Ø3 A6 Ø4
6F Ø3 6F Ø4 AØ Ø2 E2 Ø1 24 Ø7 AB Ø2 E9 Ø1 ØC 2Ø
Ø1 ØD 69 Ø4 69 Ø3 64 Ø1 66 Ø2 6A ØØ 26 E6 D7 Ø6
97 Ø7 31 31 31 33 32 39 Ø1 Ø4 FE 4F Ø1 Ø8 FC ØØ
/
```

V3A

4-21-77
NORMAL LOOP AFTER RESET
one LOOP w/ SCREEN ACTION

4C
4D
4D
4F
50
51
52
53

V3A

LOC	COMMAND	CYCLES	BYTES		VMA LINE	ADDR BUS	R/W LINE	DATA BUS
	LDS 8E IMMED	3	3	1	1	OP CODE ADDR	1	OP CODE
				2	1	" +1	1	OPERAND DATA HOB
				3	1	" +2	1	" LOB
	TSX 30 INHER	4	1	1	1	OP CODE ADDR	1	OP CODE
				2	1	" " + 1	1	OP CODE OF NEXT INSTRUCTION
				3	0	STACK POINTER	1	IRRELEVANT
				4	0	NEW INDEX REGISTER	1	IRRELEVANT
	STX DF w/26 DIRECT		2	1	1	OP CODE ADDR	1	OP CODE
				2	1	" " " +1	1	ADDRESS OF OPERAND
				3	0	ADDR OF OPERAND	1	IRRELEVANT DATA
				4	1	"	1	REGISTER DATA HOB
				5	1	" + 1	0	" " LOB
	STX DF w/ØC DIRECT		2	1	1	OP CODE ADDR	1	OP CODE
				2	1	" " " + 1	1	ADDR OF OPERAND
				3	0	ADDR OF OPERAND	1	IRRELEVANT DATA
				4	1	"	0	REGISTER DATA HOB
				5	1	" + 1	0	" " LOB
	LDA A 86 w/Ø7 IMMED		2	1	1	OP CODE ADDR	1	OP CODE
				2	1	" " " + 1	1	OPERAND DATA
	STA A B7 AT F001 EXTENDED		3	1	1	OP CODE ADDR	1	OP CODE
				2	1	" " " + 1	1	DESTINATION ADDR HOB
				3	1	" " " + 2	1	" " LOB
				4	0	OPERAND DESTINATION ADDR	1	IRRELEVANT DATA
				5	1	" " "	0	DATA FROM ACCUMULATOR

V3A

EDITOR: NORMAL LOOP
TESTING FOR CHARACTER
FROM KBD (FC52 BPL)
AND FLASHING CURSOR
IS "GET CHR" SUBROUTINE
STARTING AT FC45

FFFE & FFFF POINTER

FC00

INITIALIZATION

FC13

FC14, N=2

FC14 & 15 BSR HOME

FC32

FC37 SET SWITCHES AT

FC38 & 9

FC44

COMES OUT F 4 16 (100) INSTEAD OF F C 16 (1100) N=3

FC16 & 17 BSR CLEAR

FC18, 19, 1A JSR CRLF

FD16

2nd CHIP E12

FD19 BRANCH TO "LFT JST"

FC1B, 1C, D JSR INPCHR gets & display character

FD0F

FD15

FC1E

tests for
cont A,
Cont E
Cont D

IF NOT cont A, E or D
JUMP BNE's to

FE5C 3rd chip

JSR GETCHR

FE5F, 60, 61 3rd
JSR PUTCHR

FC45 1st CHIP

FC5F

FE62 RTS

FCAD 1st

FCBD BRA TO NORFLO

FCD0

FCE7

FDA1

FC60

FE4F

IC SOCKET
E6

BYRNE'S CHIP MODIFIED OF PDS V3A

E6 CHIP
FOR KBD PIA 1/2 AT DATA—F042
CONTROL—F043
W/PIA INIT WORD Ø4
INSTEAD OF 07

INSTEAD OF F000
F001

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	8E	01	FF	30	DF	26	DF	0C	86	04	87	F0	43	CE	0F	FF
1	DF	0E	DF	1A	8D	1C	8D	20	BD	FD	16	BD	FE	5C	81	01
2	26	03	BD	FD	A1	81	05	26	02	8D	35	81	04	26	E9	7E
3	FE	4F	CE	E0	00	DF	1C	39	C6	60	CE	E2	00	09	E7	00
4	9C	1C	26	F9	39	DE	1C	63	00	CE	26	F0	09	27	F6	7D
5	F0	43	2A	F8	DE	1C	A6	00	2A	02	63	00	B6	F0	42	39
6	DE	0C	DF	20	DE	0E	DF	22	8D	C8	8D	CC	97	32	DE	1C
7	DF	24	8D	D1	81	1B	26	04	7F	00	32	39	8D	02	20	F2
8	81	09	2D	16	2E	02	20	70	81	0D	2D	A6	2E	02	86	60
9	81	12	2D	2A	2E	08	DE	1C	20	1F	8D	3F	20	6A	81	14
A	2D	26	2E	03	09	20	13	81	1F	2D	8D	27	62	DE	1C	81
B	0D	27	04	A7	00	81	60	27	5D	08	DF	1C	20	12	DE	1C
C	C6	20	09	5A	26	FC	20	06	C6	20	08	5A	26	FC	DF	1C
D	8C	E0	00	2C	0B	D6	32	27	02	8D	6D	BD	FD	73	8D	3A
E	8C	E2	00	2B	12	8D	11	D6	32	27	0C	DE	22	9C	0E	27
F	06	8D	44	DE	14	DF	22	39	D6	32	27	0A	DE	20	DF	16

LISTED FROM
1702 ON SPHERE
4-7-79
USING JON BYRNE
MICRO BUG

PDS-V3N

2nd VINTAGE FIRMWARE.
(ALSO IS FOR KBD 2, BUT
BY CHANGING ADDRESS CALLED
IN KBD REFERENCES CAN
RUN w/ KBD 1)

V3N

```
                NAM     PDS-V3N
                OPT     O,NOG
*
*
*
*       PROGRAMMED BY ERIC JAMESON
*
*
*
*       COPYRIGHT 1976 SPHERE CORPORATION
*       940 N.  4TH EA.; NORTH SALT LAKE, UTAH  84054
*
*
*
*       SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
*       DISTRIBUTION AND USE OF THE PDS SOFTWARE.
*       NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
*       WRITTEN PERMISSION OF SPHERE CORP.
*
*
*
*
* THE PROGRAM DEVELOPMENT SYSTEM (PDS V3N) IS A SET OF
* PROGRAMS RDSIDING ON ERASABLE PROGRAMMABLE READ ONLY
* MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS
* SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE
* DEVELOPMENT OF COMPUTER PROGRAMS.
*    TOWARD THIS END, THE 4 PDS EPROMS CONTAIN A CURSOR
* BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING
* AID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16
* BIT MULTIPLY AND DIVIDE, ASCII-TO-BINARY, AND
* BINARY-TO-ASCII ROUTINES.
*
*
*
*    THE PDS-V3N PROM SET WAS WRITTEN IN ORDER TO RUN THE
* NEW KEYBOARD.  CHANGES WERE MADE IN THE EDITOR AND THE
* DEBUGGER.  AS THE NEW PROMS ARE A GREAT IMPROVEMENT OVER
* THE V3A PROMM SET, A VERSION KNOWN AS PDS-V3D WAS MADE
* WOULD RUN ON THE OLD KEYBOARDS.  THE ONLY DIFFERENCE
* BETWEEN THE V3D AND THE V3N PPROMS ARE THAT THE PIA
* ADDRESS IS CHANGED FROM F040 ON V3N TO F000 ON V3D.
* CHANGES WERE ALSO MADE IN THE DEBUGGER AND THE
* EDITOR.  THE EDITOR CHANGES WERE THAT INSERT AND DELETE
* ARE NOW AT THE TOP OF THE PAGE AND THAT THERE IS A REEDIT
* COMMAND IN THE EXEC (CTRL R) TO ALLOW RE-EDITING.  THE
* ENTRY TO THE DEBUGGER FROM THE BREAKPOINT INSTRUCTION
* HAS BEEN CHANGED SO THAT THE RETURN ADDRESS FOR THE
* BREAKPOINT IS NOW CALCULATED WHEN THE BREAKPOINT IS
* ENCOUNTERED AND GOES TO THE DEBUGGER.  IN ADDITION
* THERE ARE 2 NEW INSTRUCTIONS:  ↑J FOR DOING A JSR TO
* A ROUTINE AND ↑X TO EXIT BACK TO THE EXEC.
*
```

```
                 *                 MEMORY MAP
                 *
        0000     TMP    EQU    $00
        0002     TMP1   EQU    $02
        0004     ARB    EQU    $04       16 BIT ACC.  PSEUDO REG B.
        0004     AR3    EQU    $04       _HI BYTE OF ARB.
        0005     AR2    EQU    $05       _LO BYTE OF ARB.
        0006     ARA    EQU    $06       16 BIY ARITH PSEUDN REG A.
        0006     AR1    EQU    $06       _HI BYTE OF ARA.
        0007     AR0    EQU    $07       _LO BYTE OF ARA.
        0008     DIGIT  EQU    $08       BYTE USED BY ASCBIN FOR TMP.
        000A     OUTEND EQU    $0A       END OF OUTPUT BUFFER TEXT.
        000C     BUFADR EQU    $0C       START OF I/O BUFFER (PTR)
        000E     BUFEND EQU    $0E       PTR.  TO END OF I/O BUFFR.
        0011     OUTBUF EQU    $11       START OF OUTPUT BUFFER.
        0014     SRCADR EQU    $14       SOURCE FOR TEXT MOVES.
        0016     DSTADR EQU    $16       DEST.  ADDR.  FOR TEXT MOVE.
        001A     ENDMEM EQU    $1A       LAST ADDRES OF REAL MEMORY.
        001C     CSRPTR EQU    $1C       PTR TO CURSER ON SCREEN.
        001E     BUFPTR EQU    $1E       TEMP PTR USED BY OUTSTR.
        0020     BUFFLO EQU    $20       PTR TO END OF LOW EDIT TXT.
        0022     BUFFHI EQU    $22       PTR TO START OF HI TEXT.
        0024     SCNPTR EQU    $24       PTR.  TO BUFFRD TXT START.
        0026     SRCASM EQU    $26       PTR TO ASSMBLR SOURCE CODE.
        002A     ONDVAL EQU    $2A       HAS ASSMBLR OPERND VALUE.
        002C     SYMVAL EQU    $2C       VALUE PUT IN ASSM.  SYMTBL.
        002E     BRKSAV EQU    $2E       TEMP SAVE FOR BRKPT DATA.
        0030     BRKADR EQU    $30       ADDDRESS OF BREAKPOINT.
        0032     EDIT   EQU    $32       0 IF EDITOR IS NOT RUNNING.
        0035     IOBUFF EQU    $35       I-O BUFFER FOR DEBUGGER.
        0040     PCVAL  EQU    $40       PROGRAM COUNTER FOR ASSM.
                 *
        E01F     FRSTLN EQU    $E01F     RIGHTMOST CHAR OF LINE ONE.
        E1E0     LASTLN EQU    $E1E0     LEFT SIDE OF BOTTOM OF CRT.
        E1FF     LASTCH EQU    $E1FF     LAST CHAR ON CRT DISPLAY.
        F040     KBDPIA EQU    $F040     ADDRESS OF PIA FOR KBD/2.
                 *KEYBOARD PIA ADDRESS FOR OLD KEYBOARD (KBD/1A) IS F000.


        FE71     INPCHR EQU    $FE71     INPUTS A CHARACTER.
        FE64     DEBUG  EQU    $FE64     DEBUGGER ROUTINE.
        FF22     ASCBIN EQU    $FF22     ASCII TO BINARY ROUTINE.
```

```
                  *                 INITIALIZATION
                  *
                  *
                  *
                  *   THE INITIALIZATION ROUTINES SET UP THE INITIAL VALUES
                  * UPON SYSTEM RESET.
                  *
                  *
FC00                     ORG     $FC00
FC00 8E 01FF RESTRT LDS    #$1FF      SETS STACK POINTER
FC03 30                  TSX                MOVES STK PTR TO INDEX REG
FC04 DF 26               STX     SRCASM     SETS ASSEMBLR OUTPUT PTR
FC06 DF 0C               STX     BUFADR     INIT INPUT BUFFR ADDR.
FC08 86 1F               LDA A   #$1F       INTLZ. KEYBOARD PIA.
FC0A B7 F041             STA A   KBDPIA+1   PIA CONTROL REG. ADDRESS.
FC0D CE 0FFF             LDX     #$FFF      LAST LOCATN OF MEMORY.
FC10 DF 0E               STX     BUFEND     INIT END-OF-EDIT BUFFR.
FC12 DF 1A               STX     ENDMEM     INIT END-OF-MEM ADDR.
                  *
                  *
                  *
                  *
                  *
                  *                 COMMAND LANGUAGE
                  *
                  *    THIS EXECUTIVE ACCEPTS COMMANDS FROM THE KEYBOARD TO
                  * DETERMINE WHAT UTILITY IS TO BE RUN.  INVALID COMMANDS
                  * WILL SPACE THE CURSOR DOWN ONE LINE.  DO NOT SPACE OFF THE
                  * BOTTOM OF THE SCREEN.   V3D-5 GOES TO DEBUG IF GIVEN
                                            INVALID COMMANDS
                  *
                  *
FC14 8D 21        EXEC   BSR     HOME       CSRPTR IS HOMED.
FC16 8D 25               BSR     CLEAR      CLEARS SCREEN
FC18 8D 73        EXEC1  BSR     CR1        CRLF FOR NEW LINE.
FC1A BD FE71             JSR     INPCHR     GETS & DISPLAYS CHR.
FC1D 81 01               CMP A   #$01       TESTS IF ASSEMBLY COMMD.
FC1F 26 03               BNE     EXEC2      SKIPS IF OTHER COMMAND.
FC21 BD FDA1             JSR     ASMBLR     JUMPS TO ASSEMBL PRROGRM.
FC24 81 05        EXEC2  CMP A   #$05       TESTS IF CONTRL 'E'.
FC26 26 02               BNE     EXEC3      SKIPS IF NOT EDIT CMD.
FC28 8D 3D               BSR     EDITOR     JUMPS TO EDIT TEXT.
FC2A 81 12        EXEC3  CMP A   #$12       TESTS FOR A ↑R COMMAND.
FC2C 26 02               BNE     EXEC4      SKIPS IF NOT A REEDIT COMND.
FC2E 8D 3F               BSR     REEDIT     GOES TO REEDIT TEXT.
FC30 81 04        EXEC4  CMP A   #$04       TESTS FOR CONTRL 'D'.
FC32 26 E4               BNE     EXEC1      SKIPS BACK FOR A NEW COMND.
FC34 7E FE64             JMP     DEBUG      JUMPS TO DEBUGGER.
```

```
*                    THE EDITOR
*
*
*       THE EDITOR ALLOWS INPUT FROM THE KEYBOARD INTO A
* BUFFER MEMORY.  INPUT IS DISPLAYED ON THE SCREEN.  WHEN
* IT IS TYPED IN, THE SCREEN TEXT CAN THEN BE EDITED BY USE
* OF THE CURSOR.  WHEN THE SCREEN IS FULL OR EDITING IS
* FINISHED, THE DATA IS SCROLLED OFF THE SCREEN INTO THE
* EDIT BUFFER.  WHEN TEXT IS SCROLLED OFF THE TOP OF THE
* SCREEN, IT IS STORED FROM THE BUFFER ADDRESS POINTER
* (BUFADR) TO THE LOW BUFFER POINTER (BUFFLO).  BUFFLO
* POINTS TO THE END OF TEXT + ONE (I.E. IT POINTS TO THE
* FIRST UNUSED BYTE).  WHEN IT IS SCROLLED OFF THE BOTTOM
* OF THE SCREEN, IT IS STORED IN THE TOP OF THE EDIT BUFFER.
* THE TEXT GOES FROM THE HIGH BUFFER POINTER (BUFFHI) TO
* THE END OF BUFFER POINTER (BUFEND).  BUFFHI POINTS TO
* THE END OF TEXT - ONE (I.E. IT POINTS TO THE LAST UNUSED
* BYTE IN THE BUFFER).  WHEN THE TEXT IS SCROLLED UP
* OFF THE TOP OF THE SCREEN, TEXT IS TAKEN FROM THE HIGH
* AREA OF THE EDIT BUFFER AND DISPLAYED ON THE LAST LINE OF
* THE SCREEN.  WHEN TEXT IS SCROLLED DOWN OFF THE BOTTOM,
* TEXT, IF ANY EXISTS, IS MOVED FROM THE LOW EDIT BUFFER
* AREA TO THE TOP LINE OF THE SCREEN.
*
*
*
*
*
*
*
*
*
*
*                     POINTERS USED
*
* CSRPTR   POSITION OF CHARACTERS INSERTED ON THE SCREEN.
* SCNPTR   POSITION OF START OF EDITED TEXT ON SCREEN.
* BUFADR   START OF TEXT BUFFER IN MAIN MEMORY.
* BUFEND   END OF TEXT BUFFER IN MAIN MEMORY.
* BUFFLO   END OF TEXT SCROLLED OFF TOP OF SCREEN.
* BUFFHI   START OF TEXT SCROLED OFF BOTTOM OF SCREEN.
* BUFLEN   NOT CURRENTLY USED.
*
*
*
*
*
*                   EDITOR COMMANDS
*
* "UP ARROW"  MOVES CURSOR UP ONE LINE; CSRPTR GETS
*  CSRPTR-32; CALL NDRFLO.
*
* "DOWN ARROW"  MOVES CURSOR DOWN ONE LINE; CSRPTR GETS
*  CSRPTR+32; CALL OVRFLO.
*
* "RIGHT ARROW"  MOVES CURSOR ONE POSITION RIGHT; CSRPTR
*  GETS CSRPTR+1; CALL OVRFLO.
```

```
* "LEFT ARROW"  MOVES CURSOR ONE POSITION LEFT; CSRPTR
*  GETS CSRPTR-1; CALL NDRFLO.
*
* "CONTROL & LEFT ARROW (ON KEYBOARD)"  LEFT JUSTIFY CURSOR;
*  CSRPTR GETS CSRPTR TRUNCATED; CALL NDRFLO FOR SCNLOC CHK.
*
* "PUTCHR"  OUTPUTS CHARACTER; CSRPTR GETS
*  CSRPTR+1; GOES TO OVRFLO.
*
* "ENDCHR"  TERMINATION CHAR; CLEAR EDIT FLAG;
*  EXIT THE EDITOR.
*
* "HOME"  HOMES CURSOR POINTER; CSRPTR GETS E000; NDRFLO.
*
* "CLEAR"  CSRPTR TO END OF THE SCREEN GETS SPACES.
*
* "CTRL I"  INSERT A LINE AT THE LAST LINE ON THE SCREEN;
*  CALL OVR1 (SCROLLS UP ONE LINE); CSRPTR GETS E1E0.
*
* "CTRL D"  DELETE LAST LINE; SCROLL DOWN (UNDR2);
*  CSRPTR GETS E1E0.
*
*
*
*
* OVERFLOW CHECKS IF SCROLL UP IS NEEDED; IF IT IS, IT
*  SCROLLS UP AND MOVES DATA TO & FROM THE BUFFEERS.
*
* OVRFLO: IF CSRPTR < E200 THEN RETURN; IF EDIT IS ON THEN
* OVR1:    BUFFLO+ GETS SCNPTR TO 'C.R.';
* DSTADR GETS CSRPTR GETS E1E0 (LAST LINE ON SCREEN);
* IF EDIT IS ON AND BUFFHI < BUFEND THEN MOVE THE TEXT
* (THE STRING FROM BUFFHI TO 'C.R.') TO THE LAST LINE.
*
*
*
*
* UNDERFLOW CHECKS IF SCROLL DOWN IS NEEDED AND MOVES
*  DATA TO AND FROM THE BUFFERS.  CURSOR HAD BEEN
*  MOVED OFF THE TOP OF THE SCREEN AND IS NOW PUT AT THE
*  HOME POSITION ON THE SCREEN.
*
* NDRFLO:  IF CSRPTR > DFFF THEN RETURN (GO TO OVRFLO);
*  IF EDIT FLAG IS ON THEN MOVE LAST LINE TO BUFFHI
*   ON DOWN; SCRLDN; CSRPTR GETS E000; MOVE LINE FROM
*   BUFFLO TO FIRST LINE ON THE CRT.
*
*
*
*
*  NOTE:  DON'T SCROLL OFF SCREEN IN EXEC UNTIL AFTER
*    THE EDITOR HAS BEEN RUN.
*
*  NOTE:  EVERY LINE MUST HAVE A C.R. ON IT.
```

```
FC37 CE E000 HOME    LDX    #$E000    LOADS HOME POSITION.
FC3A DF 1C           STX    CSRPTR    STORES HOME IN CURSOR PTR.
FC3C 39              RTS              RETURNS TO CALLER.
                *
                *
FC3D C6 60   CLEAR   LDA B  #$60      LOADS BLANK (C. R. ).
FC3F CE E200         LDX    #LASTCH+1  LOADS END-OF-SCREEN PTR.
FC42 09      CLEAR1  DEX              DECREMENTS BLANKING PTR.
FC43 E7 00           STA B  0,X       BLANKS LOCATION.
FC45 9C 1C           CPX    CSRPTR    TESTS IF DONE.
FC47 26 F9           BNE    CLEAR1    BRANCHES BACK IF NOT DONE.
FC49 39              RTS              RETURNS.
                *
                *
                *
                *       GETCHR INPUTS A CHARACTER INTO ACC A WITHOUT
                *       MOVING THE CURSOR, AND BLINKS THE CURSOR.
                *
FC4A DE 1C   GETCHR  LDX    CSRPTR    LOADS CRT CURSOR POSITION.
FC4C 63 00           COM    0,X       COMPLIMENT (FLASH POSITION).
FC4E CE 26F0         LDX    #9968     LOADS BLINK COUNT VALUE.
FC51 09      GET1    DEX              COUNT GETS COUNT-1.
FC52 27 F6           BEQ    GETCHR    RESETS CTR WHEN TIMED OUT.
FC54 86 40           LDA A  #$40      LOADS MASK FOR CA2 FLAG.
FC56 B5 F041 ←FØØ1   BIT A  KBDPIA+1  TESTS IF A CHAR.  TYPED IN.
FC59 27 F6   on V3D  BEQ    GET1      BRANCH IF CHAR NOT ENTERED.
FC5B DE 1C           LDX    CSRPTR    LOADS CURSOR POSITION.
FC5D A6 00           LDA A  0,X       TESTS IF BLINKMD (SOLID).
FC5F 2A 02           BPL    GET2      SKIPS IF NOT BLINKED.
FC61 63 00           COM    0,X       CLEARS THE CHARACTER.
FC63 B6 F040 GET2    LDA A  KBDPIA    LOADS A WITH KEYBRD CHAR.
FC66 39              RTS              RETURNS TO CALLER.
                *       FØØØ on V3D
                *
                *
                *
                *         EDITOR IS THE MAIN ENTRY POINT FOR EDITING.
                *
FC67 DE 0C   EDITOR  LDX    BUFADR    BUFFLO GETS THE
FC69 DF 20           STX    BUFFLO    VALUE OF BUFADR.
FC6B DE 0E           LDX    BUFEND    BUFFHI GETS THE
FC6D DF 22           STX    BUFFHI    VALUE OF BUFEND.
FC6F 8D C6   REEDIT  BSR    HOME      ENTRY POINT FOR
FC71 8D CA           BSR    CLEAR     RE-EDITING TEXT.
FC73 97 32   EDITRD  STA A  EDIT      TURNS ON EDIT MODE.
FC75 DE 1C   EDITIN  LDX    CSRPTR    SETS SCNPTR TO CSRPTR.
FC77 DF 24           STX    SCNPTR
                *
                *
FC79 8D CF   EDREAD  BSR    GETCHR    X GETS CSRPTR & A GETS CHR.
FC7B 81 1B   ENDCHR  CMP A  #$1B      TESTS FOR AN "ESC" CHAR.
FC7D 26 04           BNE    ED1       SKIPS IF NOT EDIT END.
FC7F 7F 0032         CLR    EDIT      TURNS OFF EDIT FLAG.
FC82 39              RTS              EXITS THE EDITOR.
FC83 8D 0A   ED1     BSR    INSERT    EDITS CHARACTER.
FC85 20 F2           BRA    EDREAD    GOES FOR NEXT CHARACTER.
```

```
                  *
                  *
                  *       FOLLOWING IS THE MAIN EDITOR EXECUTION LOOP.
                  *
                  *
                  *
                  *
                  *
FC87 81 0D        CR     CMP A   #$0D     TESTS FOR CARRIAGE RETURN.
FC89 2D AC               BLT     HOME     SKIPS IF HOME CURSR COMND.
FC8B 2E 12               BGT     RTCSR    GOES TO NEXT COMND TEST.
FC8D 86 60        CR1    LDA A   #$60     LOADS INTERNAL C.R.  VALUE.
                  *
                  *
                  *
                  *
FC8F 81 09        INSERT CMP A   #$09     TESTS FOR A CONTROL 'I'.
FC91 2D 16               BLT     DELETE   SKIPS TO DELETE COMMD.
FC93 2E F2               BGT     CR       SKIPS FOR NEXT TEST.
FC95 D6 32        INSRT1 LDA B   EDIT     TESTS IF EDITOR IS ON.
FC97 27 03               BEQ     INSRT2   SKIP TO EXIT IF EDITOR OFF.
FC99 BD FD46             JSR     MOVE2    MOVES LAST LINE TO BUFFHI.
FC9C 7E FD74      INSRT2 JMP     SCRLDN   MOVES ALL LINES DOWN ONE.
                  *
                  *
                  *
FC9F 81 12        RTCSR  CMP A   #$12     TESTS FOR RIGHT ARROW.
FCA1 2D 28               BLT     SUB32    SKIPS IF AN "UP ARROW".
FCA3 2E 08               BGT     LFTCSR   SKIPS IF A "LEFT ARROW".
FCA5 DE 1C        RTARRO LDX     CSRPTR   LOADS CURSOR POINTER.
FCA7 20 1F               BRA     PUTCH1   STORES & INCREMENTS CSR.
                  *
                  *
                  *
FCA9 8D 67        DELETE BSR     OVR1A    SCROLLS UP ONE LINE.
FCAB 20 40               BRA     OVR3     MOVES NEW LAST SCREEN LINE.
                  *
                  *
                  *
FCAD 81 14        LFTCSR CMP A   #$14     TESTS IF "LEFT ARROW".
FCAF 2D 24               BLT     ADD32    SKIPS IF "DOWN ARROW".
FCB1 2E 03               BGT     CLER     SKIPS FOR NEXT TEST.
FCB3 09                  DEX              SUB. 1 FROM CSRPTR.
FCB4 20 25               BRA     ADD2     STORES CURSOR POINTER.
                  *
                  *
                  *
FCB6 81 1F        CLER   CMP A   #$1F     TESTS FOR CTRL BACK ARROW.
FCB8 2D 83               BLT     CLEAR    GOES TO CLEAR SCREEN.
FCBA 27 41               BEQ     LFTJST   MOVES CSR TO LEFT OF SCREEN.
                  *
                  *
                  * ALL OTHER CHARACTERS FALL THRU TO PUTCHR.
                  *
```

```
                  *       PUTCHR DISPLAYS A CHARACTER ON THE CRT DISPLAY AND
                  *         INCREMENTS THE CURSOR POINTER AS WELL AS CHECKING
                  *         AND HANDLING CARRIAGE RETURNS.
                  *
                  *
                  *
FCBC DE 1C        PUTCHR LDX     CSRPTR    LOADS OLD CSRPTR.
FCBE 81 0D               CMP A   #$0D      TESTS FOR EXTERNAL C.R.
FCC0 27 04               BEQ     CRLF1     SKIPS TO DO A C.R. L.F.
FCC2 A7 00               STA A   0,X       DISPLAYS CHAR ON SCREEN.
FCC4 81 60               CMP A   #$60      TESTS FOR INTERNAL C.R.
FCC6 27 4C        CRLF1  BEQ     CRLF      SKIPS FOR CR. LF.
FCC8 08           PUTCH1 INX               INCREMENTS CSRPTR.
FCC9 20 10               BRA     ADD2      TESTS FOR OVRFLO & UNDRFLO.



FCCB DE 1C        SUB32  LDX     CSRPTR    LOADS CURRENT CRSR POSITION.
FCCD C6 20               LDA B   #32       LOADS LOOP COUNT.
FCCF 09           SUB32A DEX               DECREMENTS CSRPTR.
FCD0 5A                  DEC B   -         DECREMENTS LOOP COUNTR.
FCD1 26 FC               BNE     SUB32A    SKIPS BACK IF NOT DONE.
FCD3 20 06               BRA     ADD2      SKIPS TO CHECK UNDRFLO.



FCD5 C6 20        ADD32  LDA B   #32       LOADS LOOP COUNTER.
FCD7 08           ADD32A INX               INCRE. CSRPTR IN INDEX.
FCD8 5A                  DEC B   -         DCEREMENTS LOOP COUNTER.
FCD9 26 FC               BNE     ADD32A    SKIPS BACK IF NOT DONE.
FCDB DF 1C        ADD2   STX     CSRPTR    SAVES CSRPTR.



                  * NDRFLO (UNDERFLOW) CHECKS FOR THE CURSOR GOING OFF THE
                  *  TOP OF THE SCREEN. THE INDEX REG. CONTAINS THE CURSOR
                  *  POINTER WHEN THE ROUTINE IS ENTERED.
                  *
                  *
FCDD 8C E000      NDRFLO CPX     ##E000    TESTS IF CSRPTR >= DFFF.
FCE0 2C 04               BGE     OVRFLO    SKIPS IF CSRPTR GREATER.
FCE2 8D B1               BSR     INSRT1    SCROLLS DOWN & MOVES LINE.
FCE4 8D 32               BSR     MOVE3     MOVES BUFFLO TO TOP OF CRT.
```

```
                  *
                  *
                  *      OVERFLOW CHECKS FOR SCROLLING UP (CURSOR IS OFF
                  * THE BOTTON OF THE SCREEN); INDEX CONTAINS THE CURSOR
                  * POINTER UPON ENTRY.
                  *
                  *
FCE6 8C E200 OVRFLO CPX    #$E200    TESTS AND EXITS IF
FCE9 2B 18          BMI    OVREXT    CURSOR ON SCREEN.
FCEB 8D 17          BSR    OVR1      DOES OVR1 CHECKING.
FCED D6 32   OVR3   LDA B  EDIT      TESTS IF EDIT IS ON.
FCEF 27 12          BEQ    OVREXT    EXITS IF IT IS OFF.
FCF1 DE 22          LDX    BUFFHI    LOADS HI TEXT POINTR.
FCF3 9C 0E          CPX    BUFEND    TESTS IF PTRS NOT EQU.
FCF5 27 0C          BEQ    OVREXT    EXITS IF NO TEXT.
FCF7 8D 3C          BSR    MOVE1A    MOVES CHRS TO LAST LINE.
FCF9 DE 14          LDX    SRCADR    RESETS NEW BUFFHI
FCFB DF 22          STX    BUFFHI    LOCATION.
                  *
                  *
                  *
                  *
                  *      FOLLOWING ROUTINE MOVES THE CURSER TO THE LEFT.
                  *
FCFD D6 1D   LFTJST LDA B  CSRPTR+1 LOADS LOW BYTE OF PTR.
FCFF C4 E0          AND B  #$E0      TRUNCATES TO LEFT OF LINE.
FD01 D7 1D          STA B  CSRPTR+1 SAVES L. J. ED PTR.
FD03 39      OVREXT RTS              RETURNS TO EDITOR.
                  *
                  *
                  *
                  *
                  *      OVR1 DOES ACTUAL SCROLLING UP.
                  *
FD04 D6 32   OVR1   LDA B  EDIT      TESTS IF EDIT IS ON.
FD06 27 0A          BEQ    OVR1A     SKIPS IF EDIT OFF.
FD08 DE 20          LDX    BUFFLO    LOADS TEXT PTR LOW.
FD0A DF 16          STX    DSTADR    DESTINATION OF TEXT MOVE.
FD0C DE 24          LDX    SCNPTR    SOURCE FOR MOVE.
FD0E 8D 26          BSR    MOVE1     MOVES LIN1 TO BUFFFLO.
FD10 DF 20          STX    BUFFLO    SAVES NEW BUFFLO PTR.
FD12 20 4B   OVR1A  BRA    SCRLUP    SCROLLS SCREEN UP 1.

                  *      FOLLOWING ROUTINE MOVES THE CURSOR.
                  *
FD14 8D BF   CRLF   BSR    ADD32     LINE FEED.
FD16 20 E5          BRA    LFTJST    CARRIAGE RETURN.
```

```
                *     MOVE INSTRUCTIONS MOVE FROM ONE BUFFER AREA TO
                * ANOTHER BUFFER AREA.
                *
                *
                *
                *     MOVE3 CALCULATES THE SOURCE ADDRESS OF THE DATA IN
                * BUFFLO (IF IT EXISTS) FOR MOVING TO THE FIRST LINE ON
                * THE CRT.  MOVE 1 IS THEN ENTERED TO DO THE MOVING.
                *
                *
FD18 DE 24      MOVE3   LDX     SCNPTR    CSRPTR GETS E000 (HOME).
FD1A DF 16              STX     DSTADR    SETS MOVE ADDRESS.
FD1C DE 20              LDX     BUFFLO    LOADS LO BUFFR ADDR.
FD1E 9C 0C              CPX     BUFADR    TESTS IF STRING EXISTS.
FD20 27 23              BEQ     MOVEXT    EXITS IF EMPTY.
FD22 09                 DEX               MOVES BACK FROM BLANK.
FD23 9C 0C      MV31    CPX     BUFADR    TESTS IF SRCADR = BUFFADR.
FD25 27 08              BEQ     MV32      MOVES IF START OF LINE.
FD27 09                 DEX               NEXT LOWER CHAR.
FD28 E6 00              LDA B   0,X       GETS SOURCE CHAR FOR TEST.
FD2A C1 60              CMP B   #$60      TESTS FOR "C. R. ".
FD2C 26 F5              BNE     MV31      SKIPS BACK UNTIL "C. R. ".
FD2E 08                 INX               POINTS BACK TO FIRST CHAR.
FD2F DF 20      MV32    STX     BUFFLO    SAVES LO ADDRESS.
FD31 20 03              BRA     MOVE1     MOVES DATA.


                *            MOVE1
                *
                *     MOVE1 MOVES A SET OF CHARACTERS FROM EITHER THE TOP
                * LINE OF THE SCREEN TO BUFFLO OR FROM BUFFHI TO THE
                * BOTTOM LINE OF THE SCREEN.  THE SOURCE ADDRESS IS PASSED
                * IN THE INDEX REG., THE DESTINATION ADDRESS IN DSTADR,
                * AND THE MOVE IS TERMINATED BY A "C. R. " IN THE LINE OF
                * TEXT BEING MOVED.
                *
                *
                *
                *
FD33 DE 14      MOVE    LDX     SRCADR    LOADS SOURCE ADDRESS INTO X.E
FD35 08         MOVE1A  INX               POINTS TO NEXT SOURCE CHAR.
FD36 E6 00      MOVE1   LDA B   0,X       LOADS SOURCE CHARACTER.
FD38 DF 14              STX     SRCADR    SAVES THE SOURCE POINTER.
FD3A DE 16              LDX     DSTADR    LOADS DESTINATION ADDRESS.
FD3C E7 00              STA B   0,X       STORES CHAR. IN DESTINATION.
FD3E 08                 INX               NEXT DESTINATION ADDRESS.
FD3F DF 16              STX     DSTADR    SAVES DESTINATION PTR.
FD41 C1 60              CMP B   #$60      TESTS IF MOVE FINSHED (CR).
FD43 26 EE              BNE     MOVE      SKIPS BACK IF NOT DONE.
FD45 39         MOVEXT  RTS               RETURNS TO CALLER.
```

```
                  *          MOVE2 SUBROUTINE
                  *
                  *      THE MOVE2 SUBROUTINE MOVES THE LAST LINE ON THE
                  * SCREEN TO THE HIGH AREA OF THE BUFFER (BUFFHI) DURING
                  * SCROLLING.  THE TEXT IS TEMPORARILY STORED ON THE STACK
                  * DURING THE MOVE.  THE MOVE IS TERMINATED BY A "C. R. ".
                  * THE TEXT IS STORED AT BUFFHI ON DOWN.
                  *
                  *
                  *
                  *
FD46 CE E1E0 MOVE2  LDX     #LASTLN   X GETS ADDR OF LAST LINE.
FD49 5F             CLR B   -         SETS TERMINATION FOR
FD4A 37      MV21   PSH B   STACK     POPPING.
FD4B E6 00          LDA B   0,X       LOADS SOURCE CHAR.
FD4D 08             INX               POINTS TO NEXT CHAR.
FD4E C1 60          CMP B   #$60      TESTS IF LINE TO "C. R. "
FD50 26 F8          BNE     MV21      MOVED TO STACK.
FD52 DE 22   MV22   LDX     BUFFHI    INIT. DESTINATION.
FD54 E7 00   MV23   STA B   0,X       STORES CHAR.
FD56 09             DEX               POINTS TO NEXT LOCATION.
FD57 DF 22          STX     BUFFHI    UPDATES BUFFER PTR.
FD59 33             PUL B   -         GETS NEXT CHAR.
FD5A C1 00          CMP B   #00       TESTS IF ALL CHRS STORED.
FD5C 26 F6          BNE     MV23      SKIPS BACK IF NOT STORED.
FD5E 39      MOVEX  RTS               RETURNS TO CALLER.


                  *         SCROLLUP MOVES ALL LINES UP 1, & CLEARS LAST LINE.
                  *
FD5F CE E000 SCRLUP LDX     #$E000    SETS CRT HOME POSITION.
FD62 E6 20   SCRP1  LDA B   $20,X     GETS CHAR FROM NEXT LINE.
FD64 E7 00          STA B   00,X      STORES CHAR ON PREV. LINE.
FD66 08             INX               POINTS TO NEXT LINE.
FD67 8C E1E0        CPX     #LASTLN   TESTS IF MOVE DONE.
FD6A 26 F6          BNE     SCRP1     GOES BACK IF NOT DONE.
FD6C DF 1C          STX     CSRPTR    SETS CSRPTR TO LAST LINE.
FD6E DF 16          STX     DSTADR    INIT DEST FOR NEXT MOVE.
FD70 BD FC3D        JSR     CLEAR     CLEARS LAST LINE.
FD73 39             RTS               EXITS.
```

```
                       *       SCRLDOWN MOVES ALL LINES DOWN ONE AND
                       * CLEARS THE TOP LINE ON THE SCREEN.
                       *
FD74 CE E1DF   SCRLDN LDX     #LASTLN-1  INITIALIZES THE POINTER.
FD77 E6 00     SCRD1  LDA B   0,X        LOADS DATA TO BE MOVED.
FD79 E7 20            STA B   $20,X      MOVES DATA DOWN ONE LINE.
FD7B DF 1C            STX     CSRPTR     SAVES CURSOR.
FD7D 09               DEX                POINTS TO NEXT BYTE.
FD7E 8C DFFF          CPX     #$DFFF     TESTS IF MOVE FINISHED.
FD81 26 F4            BNE     SCRD1      SKIPS BACK IF NOT DONE.
FD83 C6 60            LDA B   #$60       LOADS BLANK TO CLEAR LINE.
FD85 08        SCRD2  INX                POINTS TO NEXT CHARACTER.
FD86 E7 00            STA B   0,X        CLEARS BYTE ON LINE 1.
FD88 8C E01F          CPX     #FRSTLN    TESTS IF LINE 1 CLEARED.
FD8B 26 F8            BNE     SCRD2      SKIPS BACK IF NOT CLEARED.
FD8D 39               RTS                RETURNS.


                       *       OUTSTRING PRINTS OUT THE STRING BETWEEN THE
                       *       OUTBUF POINTER AND THE BUFEND POINTER.
                       *
FD8E DE 11     OUTSTR LDX     OUTBUF     BUFPTR GETS START OF TEXT.
FD90 A6 00     OUT1   LDA A   0,X        LOADS CHAR TO BE PUT OUT.
FD92 DF 1E            STX     BUFPTR     SAVES SOURCE POINTER.
FD94 BD FCBC          JSR     PUTCHR     PRINTS CHARACTER.
FD97 DE 1E            LDX     BUFPTR     RESTORES POINTER.
FD99 9C 0A            CPX     OUTEND     TESTS FOR END-OF-TEXT.
FD9B 27 03            BEQ     OUT2       EXITS IF END OF TEXT.
FD9D 08               INX                INCRE. PTR TO NEXT CHAR.
FD9E 20 F0            BRA     OUT1       GOES BACK FOR NEXT CHAR.
FDA0 39        OUT2   RTS                EXITS ROUTINE.
                       *
                       *       END OF EDITOR PROGRAM.
                       *
```

```
*           THE MINI-ASSEMBLER
*
*
*     THE MINI-ASSEMBLER IS A FIXED-FIELD ONE INSTRUCTION
* PER LINE 2 PASS ASSEMBLER.  THE MINI-ASSSEMBLER FORMAT
* IS DESCRIBED ON PAGES 9-2 AND 9-3 OF THE SPHERE
* OPERATORS REFERENCE MANUAL.
*     THE TWO PASSES ARE REQUIRED TO FORM THE LABEL
* ADDRESSES.  THE SECOND PASS EQUATES THE ADDRESS FOR
* LABELS REFRENCED BEFORE THEY ARE DEFINED IN THE PROGRAM.
*
*
* ON ENTRY:
* SRCASM = ADDRESS OF SOURCE TEXT TO BE ASSEMBLED.
* BUFFLO = ADDRESSED OF OBJECT CODE PRODUCED.
*
* ON EXIT:
* PCVAL (PROGRAM COUNTER VALUE) = LAST LOCATION OF
* THE ASSEMBLED OBJECT PROGRAM.
*
*
*
*
* ALGORITHM:
*
*ASMBLR:  SET PASS COUNT TO ZERO; SET PCVAL TO DSTASM;
*ASM1A:  OPERAND VALUE FORMED IN "ONDVAL":
* A GETS CHAR IN X6 (OPERAND TYPE); X GETS X+7;
* IF CHAR X6 IS A "@" THEN ONDVAL GETS VALUE FROM SYMBOL
*   TABLE ELSE ONDVAL GETS VALUE FROM ASCBIN CONVERSION;
*SYMBL:  EQUATES SYMBOL (PC VALUE IS THE " " SYMBOL
*[LABEL]) TO A LABEL VALUE:
* SYMVAL GETS PCVAL;
* IF X(1) IS AN "=" THEN SYMVAL GETS ONDVAL;
* IF X(1) IS NOT A "=" OR A SPACE THEN IF SECOND PASS THEN
*   EXIT ELSE START SECOND PASS;
* LABEL ENTRY IN SYMBOL TABLE GETS SYMVAL;
*LDOP:  PUT OPERATION CODE INTO THE OBJECT CODE:
* CONVERT X(2)-X(3) INTO BINARY;
* SAVE PCVAL;
* P. C.  GETS P. C. +1;
*OPRND:  FORM OPERAND IN OBJECT CODE:
* FORM ONDVAL INTO PROPER SIZE BASED ON CODE IN X(6);
* STORE NEW OPERAND VALUE IN MEMORY;
* P. C.  GETS P. C. +1 OR 2;
* GET NEXT LINE OF SOURCE;
* GO TO ASM1A;
*
*
```

```
FDA1 7F 0004 ASMBLR CLR     AR3       INIT. PASS CTR TO FRST PASS. )
FDA4 DE 20   ASM1   LDX     BUFFLO    SETS PC CNTR TO START OF
FDA6 DF 40          STX     PCVAL     OBJECT CODE.
FDA8 DE 26          LDX     SRCASM    LOADS ADDR FOR FIRST LINE.
FDAA DF 02   ASM1A  STX     TMP1      SAVES ADDR OF CURRENT LINE.
FDAC A6 08          LDA A   8,X       LOADS SYMBOL (LABEL).
FDAE E6 07          LDA B   7,X       LOADS OPERAND TYPE CODE.
FDB0 C1 40          CMP B   #$'@      IF @, LOADS DATA IN SYMBOL
FDB2 27 6B          BEQ     INDADR    ADDRESS, GOES TO SYMBL.
FDB4 08             INX               SETS INDEX TO START OF
FDB5 08             INX               OPERAND NUMBER.
FDB6 08             INX
FDB7 08             INX
FDB8 08             INX
FDB9 08             INX
FDBA 08             INX
FDBB BD FF22        JSR     ASCBIN    CONVRTS # TO BINARY IN B-A.
FDBE D7 2A   ASM1B  STA B   ONDVAL    STORES OPERAND VALUE IN
FDC0 97 2B          STA A   ONDVAL+1 ONDVAL.
             *
             *
             *
             * FOLLOWING FORMS THE VALUE FOR THE LABEL.
             *
FDC2 DE 02   SYMBL  LDX     TMP1      LOADS ORIG LINE PTR INTO X.
FDC4 A6 00          LDA A   0,X       LOADS SYMBOL (LABEL).
FDC6 E6 01          LDA B   1,X       LOADS LABEL CONTROL CHAR.
FDC8 DE 40          LDX     PCVAL     LABEL VALUE GETS PCVAL.
FDCA DF 2C          STX     SYMVAL    .
FDCC C1 3D          CMP B   #$'=      TESTS IF LABLE IS EQUATED.
FDCE 26 06          BNE     ASM2      SKIPS IF NOT EQUATED.
FDD0 DE 2A          LDX     ONDVAL    LABEL VALUE (SYMVAL) GETS
FDD2 DF 2C          STX     SYMVAL    THE OPERAND VALUE.
FDD4 20 0E          BRA     ASM3      CONTINUES EVALUATION.
FDD6 C1 20   ASM2   CMP B   #$'       TESTS FOR END-OF-PROGRAM.
FDD8 27 0A          BEQ     ASM3      SKIPS IF SPACE (NOT END).
FDDA 7D 0004        TST     AR3     . TESTS IF SECOND PASS.
FDDD 27 01          BEQ     ASM2A     EXITS IF SECOND PASS.
FDDF 39             RTS               EXITS THE ASSEMBLER.
FDE0 D7 04   ASM2A  STA B   AR3       SETS CTR TO SECOND PASS.
FDE2 20 C0          BRA     ASM1      GOES BACK FOR SECOND PASS.
             *
             *
             *
             * FOLLOWING PUTS THE LABEL VALUE IN THE SYMBOL TABLE.
             *
FDE4 8D 41   ASM3   BSR     SYMPTR    X GETS SYMBL TABL ENTRY ADR. E
FDE6 96 2C          LDA A   SYMVAL    STORES THE LABEL
FDE8 A7 00          STA A   0,X       ADDRESS (SYMVAL) INTO THE
FDEA 96 2D          LDA A   SYMVAL+1 SYMBOL TABEL.
FDEC A7 01          STA A   1,X
```

```
                    * FOLLOWING FORMS THE OPERATION CODE.
                    *
FDEE DE 02          LDOP   LDX      TMP1        LOADS ORIG LINE POINTER.
FDF0 08                    INX                  SETS X TO POINT TO
FDF1 08                    INX                  THE OP CODE CHARS.
FDF2 08                    INX
FDF3 A6 00                 LDA A    0,X         GETS OP CODE CHAR INTO A.
FDF5 81 20                 CMP A    #$'         TESTS IF OP CODE EXISTS.
FDF7 27 0A                 BEQ      OPRND       SKIPS IF NONEXISTANT.
FDF9 BD FF22               JSR      ASCBIN      CONVRTS OP CODE TO BINARY.
FDFC DE 40                 LDX      PCVAL       LOADS POINTR TO OBJECT CODE.
FDFE A7 00                 STA A    0,X         STORS OP INTO OBJECT CODE.
FE00 08                    INX                  SETS TO NEXT OBJ CODE LOCTN.
FE01 DF 40                 STX      PCVAL       SAVES P.C. POINTER.
                    *
                    *
                    *
                    * FOLLOWING STORES INTO THE OBJECT CODE THE SIZED OPERAND.
                    *
FE03 DE 02          OPRND  LDX      TMP1        LOADS SOURCE LINE POINTER.
FE05 A6 06                 LDA A    6,X         LOADS OPERAND SIZE CHAR.
FE07 DE 40                 LDX      PCVAL       LOADS X WITH OBJ CODE PTR.
FE09 81 45                 CMP A    #$'E        TESTS LENGTH TYPE.
FE0B 2E 31                 BGT      RELTIV      SKIPS IF AN "R" OPERAND.
FE0D 27 21                 BEQ      EXTEND      SKIPS IF AN "E" SIZE OPRND.
FE0F 81 44                 CMP A    #$'D        TESTS IF SIZE CHR EXISTS.
FE11 27 22                 BEQ      DIRECT      SKIPS IF "D"COMND EXISTS.
                    *
                    *
                    *
                    * FOLLOWING GETS THE NEXT LINE.
                    *
FE13 DE 02          ASM4   LDX      TMP1        LOADS START OF LINE IN
FE15 08             ASM4A  INX                  ORDER TO FIND NEXT LINE.
FE16 A6 00                 LDA A    0,X         LOADS CHAR FROM SORCE LINE.
FE18 81 60                 CMP A    #$60        TESTS FOR A CARRAGE RETURN.
FE1A 26 F9                 BNE      ASM4A       SKIPS BAK UNTIL C.R. FOUND.
FE1C 08                    INX                  POINTS TO FIRST LINE CHAR.
FE1D 20 8B                 BRA      ASM1A       GOES BACK TO ASSM. NEXT LINE
                    *
                    *
                    *
                    * THE FOLLOWING ARE SUBROUTINES USED BY THE MAIN CODE.
                    *
                    *
FE1F 8D 06          INDADR BSR      SYMPTR      GETS CONTENTS OF
FE21 EE 00                 LDX      0,X         SYMBOL LOCATION.
FE23 DF 2A                 STX      ONDVAL      STORES AS OPERAND.
FE25 20 9B                 BRA      SYMBL       RETURNS TO FIX LABEL VALUE.
                    *
FE27 48             SYMPTR ASL A    -           MULT LABEL BY 2 TO FORM
FE28 5F                    CLR B    -           POINTR INTO SYMBOL TABLE.
FE29 97 01          LOADX  STA A    TMP+1       LOADS POINTER INTO THE
FE2B D7 00                 STA B    TMP         SYMBOL TABLE INTO X.
FE2D DE 00                 LDX      TMP         RETURNS TO CALLER.
FE2F 39                    RTS                  RETURNS.
0
```

```
FE30 D6 2A   EXTEND LDA B  ONDVAL   STORES HI BYTE OF OPERAND
FE32 E7 00          STA B  0,X      INTO OBJECT CODE.
FE34 08             INX             INC PC TO POINT TO NXT WD.
FE35 96 2B   DIRECT LDA A  ONDVAL+1 STORES LO BYTE OF OPERAND
FE37 A7 00          STA A  0,X      INTO OBJECT CODE.
FE39 08             INX             INC & SAVE P.C. TO POINT TO
FE3A DF 40          STX    PCVAL    NEXT BYTE.
FE3C 20 D5          BRA    ASM4     GOES TO WORK ON NEXT LINE.
             *
FE3E 08      RELTIV INX             INCREMENT P.C. PTR TO POINT
FE3F DF 40          STX    PCVAL    TO NXT BYTE & SAVE P.C..
FE41 96 2B          LDA A  ONDVAL+1 LOADS LO BYTE OF OPERAND.
FE43 90 41          SUB A  PCVAL+1  FORMS RELATIVE OFFSET.
FE45 09             DEX             INSERTS RELATIVE BYTE INTO
FE46 A7 00          STA A  0,X      OBJECT CODE.
FE48 20 C9          BRA    ASM4     GOES TO ASSMBL NEXT LINE.
             *
             *      END OF THE ASSEMBLER PROGRAM.
             *
```

```
*             DEBUGGER
*
*      THE DEBUGGER FOR THE PDS SYSTEM WAS DESIGNED TO
* PROVIDE A VERSATILE TOOL FOR USE IN PROGRAM TESTING AND
* DEBUGGING.  IT ALLOWS FOR BREAKPOINTS, MINI-ASSEMBLER
* SYMBOL TABLE REFERENCING, STACK MANIPULATION AND
* INPUT IN EITHER HEXADECIMAL, OCTAL OR DECIMAL.
* THE DEBUGGER PRINTS A PROMPT CHARACTER ">" ON EVERY NEW
* LINE.  AN INSTRUCTION CAN BE TYPED IN WHENEVER THE CURSOR
* IS BLINKING, EXCEPT WHEN A NUMBER IS BEING TYPED IN.
* THE DEBUGGER CALLS THE EDITOR WHENEVER A NUMBER IS TO BE
* INPUT, SO CORRECTIONS CAN BE MADE IF THE WRONG DIGIT IS
* TYPED IN.  THE POINTER "PCVAL" POINTS TO THE CURRENTLY
* OPENED BYTE LOCATION.  THE DEBUGGER OPERATES ON WHATEVER
* BYTE IS POINTED TO BY PCVAL.  FOR FURTHER DETAILS SEE THE
* SECTION ON THE DEBUGGER IN THE OPERATORS REFERENCE MANUAL.
*
*      THE DEBUGGER IS IMPLEMENTED BY A SMALL ROUTINE TO SET
* UP ENTRY (DEBUG) AND A LARGE ROUTINE WHICH DOES A RANGE
* COMPARE TO FIND THE PROPER COMMAND AND THEN EXECUTES THE
* COMMAND (RUNBUG).  NOTE THAT SINCE COMMANDS ARE
* DIFFERENTIATED BY RANGE, ANY KEY STRUCK WILL PRODUCE
* A COMMAND EXECUTION, SUCH AS A "," BEING INTERPRETED
* AS A "+" COMMAND.
*
*
*
* COMMANDS:
*
* "C.R." LINE - PRINTS ">" OUT ON A NEW LINE.
* " " CHANGE - THE SPACE COMND CHANGES CONTENTS FROM Y TO Z.
* "+" OPNNXT - OPENS NEXT LOCATION.
* "-" OPNPRE - OPENS PREVIOUS LOCATION.
* "↑B" BRKSET - SETS A BREAKPOINT AT THE OPENED LOCATION.
* "↑C" CLRBRK - CLEARS BRKPOINT. MUST BE DONE BEFORE EXIT.
* "↑E" EXIT - PERFORM RTI - EXECUTE AT BRKPOINT LOCATION.
* "↑G" GOLOCN - STARTS EXECUTION AT OPENED LOCATION.
* "↑J" JUMP   - JUMP TO USERS SUBROUTINE.
* "↑O" OPNLOC - OPENS LOCATION THAT IS TYPED IN AFTER "O".
* "↑R" OPNREG - OPENS THE TOP-OF-STACK LOCATION.
* "↑S" SETSTK - SETS THE STACK TO THE OPENED LOCATION.
* "↑T" OPNTBL - OPENS LOCATION IN SYMBOL TABLE OF NEXT CHR.
* "↑X" GOEXEC - EXITS THE DEBUGGER - GOES BACK TO EXEC.
*
*
*
* SUBROUTINES:
*
* INPCHR - INPUTS A CHAR. INTO A AND PRINTS IT.
* INPNUM - INPUTS A NUMBER INTO B-A FROM THE KEYBOARD.
* PNTBYT - PRINTS ACC A AS 2 HEX DIGITS ON THE SCREEN.
* PNTDIG - PRINTS B-A AS 4 HEX DIGITS ON THE SCREEN.
* NEWLIN - PRINTS A C.R. AND A ">" ON THE SCREEN.
* DSPADR - PRINTS BYTE ADDRESS (XXXX) AND BYTE CONTENTS (YY)
*   AS >XXXX YY ON THE SCREEN.
*
```

```
      FE4A                    ORG     $FE4A
                       *
                       *
                       *
                       *       FOLLOWING IS LOCATION OF ENTRY OF THE BRKPT VECTOR.
                       *
      FE4A 30          BKENTR  TSX                     INDEX GETS STACK POINTER.
      FE4B E6 05               LDA B   5,X             LOADS HI RETURN ADDRESS.
      FE4D A6 06               LDA A   6,X             LOADS LOW BYTE OF ADDRESS.
      FE4F 80 01               SUB A   #1              SUB 1 FROM RETURN ADDRESS.
      FE51 C2 00               SBC B   #0              .
      FE53 E7 05               STA B   5,X             RESTORES RETURN ADDR.  TO
      FE55 A7 06               STA A   6,X             THE BREAKPOINT LOCATION.
      FE57 20 0B               BRA     DEBUG           GOES TO THE DEBUGGER.
                       *
                       *
      FE59 81 0D       LINE    CMP A   #$0D            TESTS FOR A C.R. (LINE).
      FE5B 2D 7D               BLT     JMPLCN          GOES TO 'JSR' (↑J) ROUTNE.
      FE5D 2E 4E               BGT     OPNREG          SKIPS FOR NEXT (↑R) TESTS.
      FE5F BD FCCB             JSR     SUB32           MOVES CURSOR UP ONE LINE.
      FE62 31          POPLIN  INS                     CLEANS UP STACK FOR
      FE63 31                  INS                     DISPLAY OF C.R. >.
                       *
                       *
      FE64 8D 76       DEBUG   BSR     NEWLIN          PRINTS "C.R.>".
      FE66 8D 09       DBUG1   BSR     INPCHR          READS IN COMMAND.
      FE68 BD FCA5             JSR     RTARRO          INSERT BLANK.
      FE6B DE 40               LDX     PCVAL           LOADS CURRENTLY OPENED LOC.
      FE6D 8D 09               BSR     RUNBUG          EXECUTES DEBUG COMMAND.
      FE6F 20 F5               BRA     DBUG1           GOES BACK FOR NEXT COMND.
                       *
                       *
      FE71 BD FC4A     INPCHR  JSR     GETCHR          READ IN CHAR INTO A.
      FE74 BD FCBC             JSR     PUTCHR          DISPLAYS CHARACTER.
      FE77 39                  RTS                     RETURNS TO CALLER.
                       *
                       *
                       *
      FE78 81 03       RUNBUG  CMP A   #$03            TESTS FOR A "↑C" COMMAND.
      FE7A 2D 25               BLT     BRKSET          SKIPS IF A "↑B" COMMAND.
      FE7C 2E 45               BGT     EXIT            SKIPS FOR NEXT COMND TEST.
      FE7E DE 3A       CLRBRK  LDX     BRKADR          GETS ADDRESS OF BREAKPOINT.
      FE80 96 2E               LDA A   BRKSAV          LOADS ORIG BYTE CONTENTS.
      FE82 A7 00               STA A   0,X             RESTORES BYTE DATA.
      FE84 20 66               BRA     DSPADR          GOES TO OPEN THE LOCATION.
                       *
                       *
      FE86 81 20       CHANGE  CMP A   #$'             TESTS OFR A SPACE COMND.
      FE88 2D 12               BLT     EXECTV          SKIPS TO EXIT BACK TO EXEC.
      FE8A 2E 09               BGT     OPNPRE          SKIPS FOR OTHER CMND TESTS.
      FE8C 8D 56       SPACE   BSR     INPNUM          INPUTS NEW BYTE CONTENTS.
      FE8E DE 40               LDX     PCVAL           LOADS OPENED BYTE LOCATION.
      FE90 A7 00               STA A   0,X             STORES NEW BYTE CONTENTS.
                       *
                       *
      FE92 08          OPNNXT  INX                     FORMS NEXT LOCATION ADDRES.
      FE93 20 57               BRA     DSPADR          GOES TO OPEN LOCATION BYTE.
0
```

```
FE95 81 2D    OPNPRE CMP A  #$'-      TESTS FOR A "-" COMMAND.
FE97 2D F9           BLT    OPNNXT    SKIPS FOR A "+" COMMAND.
FE99 09              DEX              FORMS PREV. LOCATION ADDR.
FE9A 20 50           BRA    DSPADR    GOES TO OPEN THE LOCATION.
                     *
                     *
FE9C 31       EXECTV INS              CLEANS UP THE STACK.
FE9D 31              INS              .
FE9E 7E FC14         JMP    EXEC      RETURNS TO THE EXECUTIVE.
                     *
                     *
FEA1 A6 00    BRKSET LDA A  0,X       LOADS DATA OF OPNED LOCATN.
FEA3 97 2E           STA A  BRKSAV    SAVES DATA OF OPNED BYTE.
FEA5 DF 30           STX    BRKADR    SAVES ADDR. OF BREAKPOINT.
FEA7 86 3F           LDA A  #$3F      LOADS SOFTWARE INTUP COMND.
FEA9 A7 00           STA A  0,X       SETS AN SWI AT OPNED BYTE.
FEAB 20 E5           BRA    POPLIN    GOES TO NEXT LINE FOR COMND.
                     *
                     *
FEAD 81 12    OPNREG CMP A  #$12      TESTS FOR "↑R" (STACK TOP).
FEAF 2D 0A           BLT    OPNLOC    GOES TO OPEN A LOCATION.
FEB1 2E 19           BGT    OPNTBL    SKIPS FOR NEXT TEST (↑T).
FEB3 30              TSX              OPENS TOP-OF-STACK.
FEB4 08              INX              PCVAL GETS STACK POINTER.
FEB5 08              INX              (CLEANS UP THE STACK).
FEB6 20 34           BRA    DSPADR    GOES TO DISPLAY THE T-O-S.
                     *
                     *
FEB8 35       SETSTK TXS              STACK POINTER GETS PCVAL.
FEB9 20 A9           BRA    DEBUG     RETURNS TO INPUT COMMAND.
                     *
                     *
FEBB 8D 27    OPNLOC BSR    INPNUM    LOADS A 16 BIT NUMBER.
FEBD D7 40    OPNLC1 STA B  PCVAL     STORES NEWLY OPENED
FEBF 97 41           STA A  PCVAL+1   LOCATION ADDRESS.
FEC1 20 2B           BRA    DSPAD1    DISPLAYS CONTNTS OF LOCATN.
                     *
                     *
FEC3 81 07    EXIT   CMP A  #$07      TESTS IF AN EXIT (↑E) COMD.
FEC5 27 11           BEQ    GOLOCN    SKIPS FOR THE "GO" COMMAND.
FEC7 2E 90           BGT    LINE      SKIPS FOR NEXT COMMD TEST.
FEC9 31              INS              CLEARS UP THE STACK.
FECA 31              INS
FECB 3B              RTI              RETURNS FROM BREAKPOINT.
                     *
                     *
FECC 81 14    OPNTBL CMP A  #$14      TESTS IF A "↑T" (TABLE).
FECE 2D E8           BLT    SETSTK    GOES TO SET STACK PTR (↑S).
FED0 2E B4           BGT    CHANGE    SKIPS FOR NEXT TEST (SPACE).
FED2 8D 9D           BSR    INPCHR    LOADS A WITH SYMBOL (LABL).
FED4 48              ASL A  .         ALIGNS ADDRESS FOR
FED5 5F              CLR B  .         SYMBOL TABLE ENTRY.
FED6 20 E5           BRA    OPNLC1    SAVES AND DISPLAYS ADDRESS.
                     *
                     *
FED8 31       GOLOCN INS              CLEANS UP THE STACK.
FED9 31              INS              .
FEDA 6E 00    JMPLCN JMP    0,X       JUMPS TO USERS PROGRAM.
```

SAME IN "DAVES" BUT DIFFERENT IN V3D-5

```
                      *
                      * FOLLOWING ARE SUBROUTINES USED BY THE DEBUGGER.
                      *
FEDC 86 0D    NEWLIN LDA A  #$0D     LOADS A CARRIAGE RETURN.
FEDE 8D 3E           BSR    PNTBF1   PRINTS A CARRIAGE RETURN.
FEE0 86 3E           LDA A  #'>      LOADS A PROMPT CHARACTER.
FEE2 20 3A           BRA    PNTBF1   DISPLAYS PROMPTER CHAR.
                      *
                      *
                      * FOLLOWING INPUTS A 16 BIT NUMBER INTO THEE BA REGISTER.
                      *
FEE4 BD FC75  INPNUM JSR    EDITIN   INPUTS A STRING OF DIGITS.
FEE7 DE 24           LDX    SCNPTR   LOADS ADDR. OF FIRST DIGIT.
FEE9 8D 37           BSR    ASCBIN   CONVERTS TO BINARY # IN BA.
FEEB 39              RTS             RETURNS TO CALLER.
                      *
                      *
                      * FOLLOWING DISPLAYS THE LOCATION ADDR. & CONTENTS.
                      *
FEEC DF 40    DSPADR STX    PCVAL    SAVES OPENED LOCATION ADDR.
FEEE 8D EC    DSPAD1 BSR    NEWLIN   PRINTS A "C.R." AND ">".
FEF0 8D 0A           BSR    PNTDIG   PRINTS OUT "PCVAL" IN HEX.
FEF2 BD FCA5         JSR    RTARRO   PRINTS A SPACE.
FEF5 DE 40           LDX    PCVAL    LOADS PTR. TO OPEND LOC.
FEF7 A6 00           LDA A  0,X      LOADS DATA FROM LOCATN.
FEF9 8D 07           BSR    PNTBYT   PRINTS DATA IN HEX FORMAT.
FEFB 39              RTS             RETURNS TO INPUT COMMAND.
                      *
                      *
FEFC 96 40    PNTDIG LDA A  PCVAL    PRINTS THE 2 HI HEX DIGITS
FEFE 8D 02           BSR    PNTBYT   OF OPENED ADDRESS.
FF00 96 41           LDA A  PCVAL+1  PRINTS OUT 2 LOW HEX DIGITS
                      *
                      *
                      * FOLLOWING PRINTS OUT 2 HEX DIGITS.
                      *
FF02 CE 0010  PNTBYT LDX    #16      LOADS 16 FOR BASE.
FF05 DF 04           STX    ARB      STORES FOR CONVERSION.
FF07 5F              CLR B  -        CLEARS HI 2 DIGITS.
FF08 CE 0035         LDX    #IOBUFF  LOADS OUTPUT BUFF ADDRESS.
                      *
                      *
                      * FOLLOWING CONVERTS BYTE TO HEX WITH LEADING ZEROS.
                      *
FF0B D7 36    CONVRT STA B  IOBUFF+1 CLERS BYTE FOR SECOND DIGIT..
FF0D BD FF64         JSR    BINASC   CONVERTS TO ASCII DIGITS.
FF10 96 35           LDA A  IOBUFF   LOADS HI DIGIT.
FF12 D6 36           LDA B  IOBUFF+1 TESTS BOTH DIGITS CONVTD.
FF14 26 04           BNE    PNTBUF   SSKIPS IF BOTH DIGITS CONVTD00304 FF16
97 36 FF16 97 36 STA A  IOBUFF+1 SETS UP LOW DIGIT.
FF18 86 30           LDA A  #$'0     HIGH DIGIT GETS A "0".
                      *
FF1A 8D 02    PNTBUF BSR    PNTBF1   PRINTS OUT HI DIGIT.
FF1C 96 36           LDA A  IOBUFF+1 LOADS LOW DIGI
FF1E BD FCBC  PNTBF1 JSR    PUTCHR   DISPLAYS CCHARACTER.
FF21 39              RTS             RETURNS TO CALLING PROGRAM.
                      *
                      *     END OF DEBUGGER
```

```
*       UTILITY PROGRAMS
*
*
*
*
*           ASCII TO BINARY CONVERSION.
*
*      THE ASCII TO BINARY ROUTINE CONVERTS FROM AN ASCII
* NUMBER STRING POINTED TO BY X TO AN UNSIGNED 16 BIT
* BINARY NUMBER IN BA (ACC B HAS THE HI BYTE, ACC A HAS
* THE LO BYTE).  THE ASCII STRING IS TERMINATED BY A NON
* HEXADECIMAL CHARACTER.  UPON EXITING, THE INDEX REGISTER
* WILL POINT TO THE NEXT CHARACTER AFTER THE NUMBER
* STRING.  THE BASE OF THE NUMBER STRING IS PASSED TO
* THE ROUTINE IN ARA (ARA IS THE ARITHMETIC REGESTER A
* LOCATED IN BYTES 06 AND 07 OF LOW MEMORY).  IF THE
* ROUTINE IS ENTERED WITH A KNOWN BASE, PUT THE BASE
* (BETWEEN 2 AND 16) IN ARA AND ENTER THE ROUTINE AT
* THE ENTRY POINT ENTR2.
*
*
*
*      CONVERSION FORMULA:
* ASCII NUMBBER STRING X[4],X[3],X[2],X[1] IN
*BASE Y;
* BINARY NUMBER =
*      X[4]*Y↑3+X[3]*Y↑2+X[2]*Y↑1+X[1]*Y↑0          OR
* BINARY NUMBER =
*      (((0*Y+X[4])*Y+X[3])*Y+X[2])*Y+X[1]
*WHERE ↑ IS THE EXPONENT OPERATOR,
* X IS A CHARACTER & Y IS THE BASE.
*
*
*
*      ALGORITHM:
*ASCBIN:  FORM THE BASE IN ARA BASED ON THE FIRST CHAR.
*  OF THE NUMBER STRING; INCREMENT CHAR. PTR.  IN X;
*ENTR2:  NUMBER (IN BA) GETS 0;
*NXTCHR:  IF THE CURRENT CHAR. POINTED TO BY X IS NOT A
*  DIGIT THEN EXIT ELSE INCREMT CHARACTER PTR IN INDEX;
*  CONVERT DIGIT TO BINARY;
*  NUMBER GETS NUMBER * BASE;
*  NUMBER GETS NUMBER + DIGIT;
*  GO TO OPERATE ON THE NEXT DIGIT (NXTCHR);
*
```

```
FF22 A6 00   ASCBIN LDA A  0,X      GETS CHR TO FORM BASE.
FF24 81 2E          CMP A  #$'.     TESTS FOR DECML STRNG.
FF26 2D 06          BLT    OCT      SKIPS IF BASE 8 (*).
FF28 2E 09          BGT    HEX      SKIPS IF BASE 16.
FF2A 86 0A          LDA A  #10      LOADS BASE 10 FOR CONVERSN.
FF2C 20 02          BRA    ASC1     SKIPS TO INC. TEXT POINTR.
FF2E 86 08   OCT    LDA A  #8       LOADS BASE 8 FOR CONVERSION.
FF30 08      ASC1   INX             INCREMENT PTR TO NEXT CHAR.
FF31 20 02          BRA    ASC2     SKIPS TO SAVE BASE.
FF33 86 10   HEX    LDA A  #16      LOADS BASE 16 FOR CONVERN.
FF35 97 07   ASC2   STA A  AR0      SAVES BASE IN BASE#.
             *
             *
FF37 5F      ENTR2  CLR B  NUMBER   GETS 0.
FF38 37             PSH B  -        (LOW NUMBER ON STACK).
FF39 D7 06          STA B  AR1      CLEARS HI OF BASE.
FF3B A6 00   NXTCHR LDA A  0,X      GETS CHAR TO CONVERT.
FF3D 08             INX             INC TO NEXT CHARACTER.
FF3E 81 30          CMP A  #$'0     TESTS FOR END-OF-STRING.
FF40 2D 20          BLT    AEXIT    EXITS IF END.
FF42 80 30          SUB A  #$'0     FORMS B. C. D. NUMBER.
FF44 81 0A          CMP A  #10      TESTS IF DECIMAL DIGIT.
FF46 2D 0A          BLT    ASC3     SKIPS IF DECIMAL.
FF48 81 10          CMP A  #16      TESTS FOR END OF STRING.
FF4A 2F 16          BLE    AEXIT    EXITS IF NOT A HEX DIGIT.
FF4C 80 07          SUB A  #7       FORMS A HEX B. C. D. DIGIT.
FF4E 81 10          CMP A  #16      TESTS FOR END-OF-STRING.
FF50 2C 10          BGE    AEXIT    EXITS IF CHAR > "F".
FF52 97 08   ASC3   STA A  DIGIT    SAVES DIGIT FOR ADD.
             *
FF54 DF 00   CNVASC STX    TMP      SAVES INDEX REG FOR MULT.
FF56 32             PUL A  -        RESTORES LO OF "NUMBER".
FF57 8D 3A          BSR    MULT     NUMBER GETS NUMBER * BASE.
FF59 9B 08          ADD A  DIGIT    NUMBER GETS NUMBER + DIGIT.
FF5B C9 00          ADC B  #0
FF5D 36             PSH A  -        SAVES LO OF NUMBER.
FF5E DE 00          LDX    TMP      RESTORES STRING POINTER.
FF60 20 D9          BRA    NXTCHR   GOES TO CONVRT NEXT CHAR.
FF62 32      AEXIT  PUL A  -        RESTORES "NUMBER" IN BA.
FF63 39             RTS             RETURNS TO CALLING PROGRAM.
```

```
                *            BINARY TO ASCII
                *
                *
                *     THE BINARY TO ASCII CONVERSION ROUTINE CONVERTS
                * A 16 BIT BINARY NUMBER IN THE BA REGISTR (REG B & REG A)
                * TO A STRING OF ASCII DIGITS.  THE ASCII STRING CAN BE IN
                * ANY BASE FROM BASE 2 THROUGH BASE 41.  THE VALUE OF THE
                * BASE IS LOCATED IN THE ARITHMETIC PSEUDO-REGISTER ARB
                * (ARB IS LOOCATED IN BYTE AR3 [LOC 4] AND AR2 [LOC 5]).
                * WHEN THE ROUTINE IS ENTERED, THE POINTER TO THE OUTPUT
                * LOCATION IS PASSED IN THE INDEX REG.  WHEN THE ROUTINE
                * EXITS, THE INDEX POINTS TO THE LAST DIGIT IN THE STRING
                * PLUS ONE.
                *     CONVERSION IS DONE BY THE METHOD OF REPEATED
                * DIVISION.  THE LOW ORDER DIGIT IS FORMED FIRST.  THE
                * DIGITS ARE THEN PLACED ON THE STACK UNTIL CONVERSION IS
                * COMPLETED.  THE DIGITS ARE THEN POPPED OFF THE STACK
                * AND PLACED IN THE OUTPUT STRING.  THE TOP-OF-STACK IS
                * INITIALIZED TO HEX FF TO TELL WHEN ALL THE DIGITS
                * HAVE BEEN POPPED OFF THE STACK.  AFTER THE DIVISION, THE
                * DIGIT (THE REMAINDER OF THE DIVISION OPERATION)
                * IS LOCATED IN THE AR 0 PART OF ARA (BYTE 7).  WHEN
                * THE QUOTIENT OF THE DIVISION IS 0, THEN THE CONVERSION
                * IS COMPLEATED.
                *
                *
                *
FF64 DF 00      BINASC STX    TMP      SAVES OUTPUT POINTER.
FF66 34                DES             /SETS THE TOP-OF-STACK TO
FF67 30                TSX             /ALL ONES TO TELL END OF
FF68 6F 00             CLR    0,X      /CHAR STRING (LAST CHAR IS
FF6A 63 00             COM    0,X      /PUT ON STACK FIRST).
FF6C DE 04      BIN1   LDX    ARB      RESTORES DIVISOR (BASE).
FF6E DF 06             STX    ARA
FF70 8D 3D             BSR    DIVIDE   * QUOTIENT IN BA GETS THE
                * REMAINDER OF # TO BE CONVERTED; REMAINDER IN ARA GETS
                * THE LOW ORDER DIGIT.
FF72 97 02             STA A  TMP1     SAVES A OF BA.
FF74 96 07             LDA A  AR0      LOAD DIGIT (REMAINDER).
FF76 36                PSH A  -        STACK DIGIT (REVERSE ORDER). G
FF77 96 02             LDA A  TMP1     RESTORES A OF BA.
FF79 4D                TST A  -        /TESTS IF QUOTIENT IS = 0
FF7A 26 F0             BNE    BIN1     /(SIGNIFYING THAT
FF7C 5D                TST B  -        /THE CONVERRSION
FF7D 26 ED             BNE    BIN1     /IS DONE).
                *
FF7F DE 00      BINSTR LDX    TMP      RESTORES OUTPUT POINTER.
FF81 32         BIN3   PUL A  -        UNSTACK A DIGIT.
FF82 4D                TST A  -        TESTS IF NEG (END?).
FF83 2A 01             BPL    BIN4     SKIPS IF A DIGIT.
FF85 39                RTS             EXITS FROM SUBROUTINE.
FF86 81 09      BIN4   CMP A  #9       TESTS IF RESULT IS HEX.
FF88 2F 02             BLE    BIN5     SKIPS IF DIGIT NOT HEX.
FF8A 8B 07             ADD A  #7       FORMS HEX VALUE OF DIGIT.
FF8C 8B 30      BIN5   ADD A  #$'0     FORMS DECIMAL CHARACTER.
FF8E A7 00             STA A  0,X      OUTPUTS CHARACTER.
FF90 08                INX             POINTS TO NEXT CHARACTER.
FF91 20 EE             BRA    BIN3     GOES BACK FOR NEXT DIGIT.
```

```
                *         MULTIPLY ROUTINE
                *
                *     THE MULTIPLY ROUTINE MULTIPLIES TWO 16 BIT BINARY
                * NUMBERS TOGETHER TO PRODUCE A 16 BIT RESULT.  THE BA
                * REGISTERS AND ARA (BYTES 6 & 7) REGISTER ARE USED.
                * THE CONTENTS OF ARA ARE UNCHANGED UPON PROGRAM EXIT.
                *
                *
                *  BA GETS BA * ARA
                *
                *
                *     MULTIPLYING IS ACCOMPLISHED BY REPEATED ADDITIONS
                * OF ONE OF THE OPERATORS (OPERATOR ARA) INTO THE RESULT.
                * THE RESULT STARTS OUT WITH A ZERO VALUE AND IS SHIFTED
                * OVER ONE AFTER EACH ADDITION.  THE HIGHEST ORDER VALUE
                * IS ADDED IN FIRST AND THEN, GOING TO THE RIGHT,
                * (THUS SHIFTING THE ANSWER LEFT ONE TO BRING IN THE NEXT
                * RIGHTMOST DIGIT) GETTING THE NEXT LOWERMOST SIGNIFICANT
                * DIGIT.  THE NEXT RIGHTMOST BIT OF THE OTHER OPERAND
                * (THE ONE ORIGINALLY IN BA) IS TESTED, AND IF ONE,
                * ANOTHER ADDITION TAKES PLACE.  THIS IS REPEATED UNTIL
                * THE FINAL SUM IS FORMED.
                *
                *
                *
                *  MULTIPLY ALGORITHM:
                *
                *MULT:  STACK BA; BA GETS 0; SET COUNT VALUE TO 16;
                *MUL1:  SHIFT BA LEFT 1;
                *  SHIFT LEFT ORIG BA VALUE ON STACK INTO CARRY;
                *  IF CARRY = 0 THEN GO TO MUL2
                *  BA GETS BA + ARA;
                *MUL2:  DECREMENT COUNT;
                *  IF COUNT # 0 THEN GO TO MUL1 ELSE EXIT.
                *
                *
                *
                *
FF93 36         MULT   PSH A   -       PUTS THE ORIGINAL CONTENTS
FF94 37                PSH B   -       OF BA ONTO THE STACK.
FF95 86 10             LDA A   #16     LOADS COUNT VALUE
FF97 36                PSH A   -       ONTO THE STACK.
FF98 4F                CLR A   -       BA GETS ZEROED.
FF99 5F                CLR B
FF9A 30                TSX             SET INDEX TO STACK.
FF9B 48         MUL1   ASL A   -       SHIFT LEFT BA.
FF9C 59                ROL B
FF9D 68 02             ASL     2,X     SHIFTS ORIG. BA OPERAND
FF9F 69 01             ROL     1,X     ONE LEFT INTO CARRY.
FFA1 24 04             BCC     MUL2    SKIPS ADDING IF CARRY = 0.
FFA3 9B 07             ADD A   AR0     BA GETS BA + ARA.
FFA5 D9 06             ADC B   AR1
FFA7 6A 00      MUL2   DEC     0,X     TESTS IF DONE.
FFA9 26 F0             BNE     MUL1    GOES BACK IF NOT DONE.
FFAB 31                INS             CLEANS UP THE STACK.
FFAC 31                INS
FFAD 31                INS
FFAE 39                RTS             EXITS ROUTINE.
```

```
                 *           DIVIDE ROUTINE
                 *
                 *       THE DIVIDE SUBROUTINE DIVIDES THE 16 BIT NUMBER
                 * IN THE BA REGISTERS BY THE 16 BIT NUMBER IN THE PSEUDO
                 * REGESTER ARA (LOCATED IN BYTES 6 & 7).  UPON EXITING,
                 * BA WILL CONTAIN THE QUOTIENT OF THE DIVISION AND ARA
                 * WILL CONTAIN THE REMAINDER.  THE DIVIDEND BA IS DIVIDED
                 * BY THE DIVISOR ARA (I.E.  BA/ARA ).
                 *
                 *
                 *
                 *
                 *  DIVIDE ALGORITHM:
                 *
                 *DIVIDE:  X3,4 GETS BA (BA IS PUT ON THE STACK);
                 *  X1,2 GETS THE 16 BIT ARA VALUE (ARA PUT ON THE STACK);
                 *  COUNT GETS 1 + THE NUMBER OF NONSIGNIFICANT BITS IN
                 * THE DIVISOR [LEFT JUSTIFY X1,2 TO FIRST 1 BIT, COUNT
                 * GETS 1 + THE # OF LEADING ZEROS IN X1,2] [COUNT WILL BE
                 * FROM 1 TO 17];
                 *  BA GETS X3,4 [RESTORES BA];
                 *  X3,4 GETS 0 [INITIALIZES THE QUOTIENT];
                 *DIV3:  BA GETS BA - X1,2;
                 *  IF THERE WAS A BORROW [I.E. DIVIDEND IN BA < DIVISOR
                 * IN X1,2 (ARA)] THEN BA GETS BA + X1,2 [ORIGINAL BA
                 * VALUE RESTORED] & CARRY CLEARED ELSE CARRY IS SET;
                 *DIV5:  X3,4 [QUOTIENT] GETS CARRY LEFT SHIFTED IN;
                 *  X1,2 [DIVISOR] GETS SHIFTED RIGHT ONE PLACE WITH ZERO
                 * FILLED IN FROM THE LEFT SIDE;
                 *  DECREMENT COUNT;
                 *  EXIT IF DONE ELSE GO TO DIV3;
                 *
                 *
                 *
                 *
                 *
FFAF 36          DIVIDE PSH A   -       LOADS DIVIDEND INTO X3,4.
FFB0 37                 PSH B
FFB1 96 06              LDA A   AR1     LOADS DIVISOR FROM ARA.
FFB3 D6 07              LDA B   AR0
FFB5 37                 PSH B   -       PUTS DIVISOR INTO X1,2.
FFB6 36                 PSH A
FFB7 34                 DES             SET UP SPACE FOR COUNT.
FFB8 30                 TSX             INDEX GETS STACK POINTER.
FFB9 86 01              LDA A   #1      INITIALIZE COUNT.
FFBB 6D 01              TST     1,X     TESTS FOR HI DIVISR BIT ON.
FFBD 2B 0B              BMI     DIV2    SKIPS IF ON.
FFBF 4C          DIV1   INC A   -       COUNTS LEADING ZEROS.
FFC0 68 02              ASL     2,X     LEFT JUSTIFIES X1,2.
FFC2 69 01              ROL     1,X
FFC4 2B 04              BMI     DIV2    SKIPS IF NO LEADING ZERO.
FFC6 81 11              CMP A   #17     TESTS FOR ALL ZERO DIVISOR.
FFC8 26 F5              BNE     DIV1    GOES BACK IF BITS LEFT.
FFCA A7 00       DIV2   STA A   0,X     SETS COUNTER.
FFCC E6 03              LDA B   3,X     BA GETS ORIGINAL
FFCE A6 04              LDA A   4,X     DIVIDEND VALUE.
FFD0 6F 03              CLR     3,X     CLEARS X3,4 FOR FORMATION
FFD2 6F 04              CLR     4,X     OF THE QUOTIENT.
```

```
00578 FFD4 A0 02    DIV3   SUB A   2,X      START OF DIVIDE LOOP.
00579 FFD6 E2 01           SBC B   1,X
00580 FFD8 24 07           BCC     DIV4     SKIP IF DIVIDEND < DIVISOR.
00581 FFDA AB 02           ADD A   2,X      RESTORES DIVIDEND IN BA.
00582 FFDC E9 01           ADC B   1,X
00583 FFDE 0C              CLC              CLEARS THE CARRY.
00584 FFDF 20 01           BRA     DIV5     SKIPS WITH CARRY CLEAR.
00585 FFE1 0D       DIV4   SEC              SETS CARRY TO 1.
00586 FFE2 69 04    DIV5   ROL     4,X      SHIFT CARRY INTO
00587 FFE4 69 03           ROL     3,X      QUOTIENT X3,4
00588 FFE6 64 01           LSR     1,X      SHIFTS DIVISOR X1,2
00589 FFE8 66 02           ROR     2,X      RIGHT ONE.
00590 FFEA 6A 00           DEC     0,X      DECREMENTS COUNTER.
00591 FFEC 26 E6           BNE     DIV3     GOES BACK IF NOT DONE.
00592 FFEE D7 06           STA B   AR1      STORES REMAINDER IN ARA.
00593 FFF0 97 07           STA A   AR0
00594 FFF2 31              INS              CLEANS UP THE STACK.
00595 FFF3 31              INS
00596 FFF4 31              INS
00597 FFF5 33              PUL B   STORES   QUOTIENT IN BA.
00598 FFF6 32              PUL A
00599 FFF7 39              RTS              EXITS ROUTINE.
00600               *
00601               *
00602               *
00603 FFF8 0104     IRQ    FDB     $0104    INTERRUPT REQUEST VECTOR.
00604 FFFA FE4A     SWI    FDB     BKENTR   SOFTWARE INT.  VECTOR ADDR.
00605 FFFC 0108     NMI    FDB     $0108    NON-MASKABLE-INT.  VECT.
00606 FFFE FC00     RST    FDB     $FC00    RESTART VECTOR ADDRESS.
00607               *
00608               *
00609               *
00610               *  END OF PDS SOURCE LISTING.
00611               *
00612               *
00613                      END
TOTAL ERRORS 00000
```

PDS SYS 2N

(SYS 2A EARLIER PARTIAL INFO)

CASSETTE DRIVER

FIRMWARE 5TH EPROM @ FBxx
1702

2NE FIXED IS LATEST REV OF THIS ITEM

SYS 2
CASS
DRIVER

EARLIEST DOCUMENT FOR
CASSETTE ROUTINES

```
            NAM     PDS-SYS2A
            OPT     O,NOG
*
*
*
*           PROGRAMMED BY ERIC JAMESON
*
*
*
*           COPYRIGHT 1976 SPHERE CORPORATION
*           940 N.   400 EA.;   NORTH SALT LAKE, UTAH  84054
*
*           P. O.  BOX 213;   BOUNTIFUL, UTAH  84010
*
*
*
*
*           SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
*           DISTRIBUTION AND USE OF THE PDS SOFTWARE.
*           NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
*           WRITTEN PERMISSION OF SPHERE CORP.
*
*
*
*
*
* THE PROGRAM DEVELOPMENT SYSTEM (PDS SYS2A) IS A SET OF
* PROGRAMS RESIDING ON ERASABLE PROGRAMMABLE READ ONLY
* MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS
* SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE
* DEVELOPMENT OF COMPUTER PROGRAMS.
*     TOWARD THIS END, THE 5 PDS EPROMS CONTAIN A CURSOR
* BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING
* AID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16
* BIT MULTIPLY AND DIVIDE, ASCHI-TO-BINARY, AND
* BINARY,TO-ASCII ROUTINES, AND ROUTINES TO DO
* INPUT AND OUTPUT TO THE AUDIO CASSETTE.
*
*
*
```

```
                 *                 MEMORY MAP
                 *
        0000     TMP    EQU    $00
        0002     TMP1   EQU    $02
        0004     ARB    EQU    $04       16 BIT ACC. PSEUDO REG B.
        0004     AR3    EQU    $04       _HI BXTE OF ARB.
        0005     AR2    EQU    $05       _LO BYTE OF ARB.
        0006     ARA    EQU    $06       16 BIY ARITH PSEUDO REG A.
        0006     AR1    EQU    $06       _HI BYTE OF ARA.
        0007     AR0    EQU    $07       _LO BYTE OF ARA.
        0008     DIGIT  EQU    $08       BYTE USED BY ASCBIN FOR TMP.
        0009     CSTATS EQU    $09       CASSETTE I-O STATUS BYTE.
        000A     OUTEND EQU    $0A       END OF OUTPUT BUFFER TEXT.
        000C     BUFADR EQU    $0C       START OF I/O BUFFER (PTR)
        000E     BUFEND EQU    $0E       PTR. TO END OF I/O BUFFR.
        0011     OUTBUF EQU    $11       START OF OUTPUT BUFFER.
        0013     CASNUM EQU    $13       PHYSICAL CASSETTE NUMBR.
        0014     SRCADR EQU    $14       SOURCE FOR TEXT MOVES.
        0016     DSTADR EQU    $16       DEST. ADDR. FOR TEXT MOVE.
        0018     ACIANO EQU    $18       SYS2A ACIA ADR (FDOS = 30).
        001A     ENDMEM EQU    $1A       LAST ADDRES OF REAL MEMORY.
        001C     CSRPTR EQU    $1C       PTR TO CURSER ON SCREEN.
        001E     BUFPTR EQU    $1E       TEMP PTR USED BY OUTSTR.
        0020     BUFFLO EQU    $20       PTR TO END OF LOW EDIT TXT.
        0022     BUFFHI EQU    $22       PTR TO START OF HI TEXT.
        0024     SCNPTR EQU    $24       PTR. TO BUFFRD TXT START.
        0026     SRCASM EQU    $26       PTR TO ASSMBLR SOURCE CODE.
        002A     ONDVAL EQU    $2A       HAS ASSMBLR OPERND VALUE.
        002C     SYMVAL EQU    $2C       VALUE PUT IN ASSM. SYMTBL.
        002E     BRKSAV EQU    $2E       TEMP SAVE FOR BRKPT DATA.
        0030     BRKADR EQU    $30       ADDDRESS OF BREAKPOINT.
        0032     EDIT   EQU    $32       0 IF EDITOR IS NOT RUNNING.
        0033     BLKNAM EQU    $33       CASSETTE BLOCK NAME.
        0035     IOBUFF EQU    $35       I-O BUFFER FOR DEBUGGER.
        003A     NOPRNT EQU    $3A       CASSETTE NAME PRINT FLAG.
        003B     BLKTYP EQU    $3B       CASSETTE BLOCK TYPE CODE.
        003C     BFRPTR EQU    $3C       ADDR. OF I-O BUFF. FOR CASS.
        003E     BFRSZE EQU    $3E       LENGTH OF CASS. BUFFER.
        0040     PCVAL  EQU    $40       PROGRAM COUNTER FOR ASSM.


                 *      FOLLOWING ARE VARIABLE VALUES.
                 *
        26F0     TIMEA  EQU    $26F0     TIMER COUNTER.
        0011     ON     EQU    $11       ACIA VALUE TO TURN ON CASS.
        0051     OFF    EQU    $51       ACIA VALUE TO STOP CASS.
        0003     ETX    EQU    $03       END-OF-TEXT.
        0016     SYN    EQU    $16       SYNCHRONISE.
        0017     ETB    EQU    $17       END-OF-TRANSMISSION-BLOCK.
        001B     ESC    EQU    $1B       ENCAPE TO NONSTANDARD HDR.
        0054     ERR4   EQU    $'T       SET  FOR TRAILER ERROR.
        0043     ERR5   EQU    $'C       SET  FOR CHECKSUM ERR.
        51E1     TIME   EQU    20833     TIME CNTR FOR 1/4 SEC.
        0008     TIMCNT EQU    8         TIMES FOR 2 SECONDS.
0
```

```
      *
FB00          ORG     $FB00
      *
      *
      *            CASSETTE I-O DRIVERS
      *
      *     THE CASSETTE DRIVERS LOAD AND DUMP A BLOCK OF DATA
      * TO AND FROM THE CASSETTE.  THEY HANDLE BOTH THE HEADER
      * AND TRAILER FORMATTING.
      *     THE DRIVERS ARE SET UP AR A SET OF SUBROUTINES
      * CALLED BY THE ECUTIVE OR THE USER'S PROGRAMS.
      *
      *   F060 = #2
      *       LOW MEMORY ADDRESSES USED BY THE DRIVERS ARE:
      *       CSTATS   AT   09     USED TO STORE INPUT ERR CODE.
      *       ACIANO   AT   18     PTR. TO THE ACIA CURRENTLY USED.
      *       BLKNAM   AT   33     2 CHAR. NAME OF BLOCK.
      *       NOPRNT   AT   3A     PRINT FLAG FOR BLOCK NAME.
      *       BFRPTR   AT   3C     START OF CASSETTE I-O BUFFER.
      *       BFRSZE   AT   3E     END OF CASSETTE DATA BUFFER.
      *
      *
      *     THE DRIVERS CAN BE RUN FROM THE PDS-SYS2A SOFTWARE
      * SYSTEM, WHICH CONTAIN THE CASSETTE LOAD AND DUMP
      * COMMANDS IN THE EXEC, OR FROM THE PDS-V3A, WHERE THE
      * DRIVERS ARE CALLED THROUGH THE DEBUGGER.
      *
      *     THE DRIVERS DISPLAY PERTINENT DATA ON THE SCREEN
      * WHEN WRITING OR READING FROM THE CASSETTES.  ON A WRITE,
      * THE CHARACTERS BEING WRITTEN ONTO THE CASSETTE ARE
      * DISPLAYED IN THE SECOND CHARACTER POSITION FROM THE
      * UPPER RIGHT HAND CORNER OF THE SCREEN.  ON A READ, THE
      * CHARACTERS BEING READ IN ARE DISPLAYED IN THE UPPER
      * RIGHT HAND CORNER OF THE SCREEN.  THE NAME OF THE BLOCK
      * CURRENTLY BEING READ OR SEARCHED OVER IS DISPLAYED ON THE
      * RIGHT HAND SIDE OF THE SECOND LINE.  THE ERROR CODE FOR
      * A READ IS DISPLAYED 1 POSITION TO THE LEFT OF THE NAME.
      * NO CHANGE IN CHARACTER MEANS THAT THE READ WAS O.K.
      * A "C" MEANS THERE WAS A CHECKSUM ERROR ON THE BLOCK.
      * A "T" MEANS THAT THE WRONG NUMBER OF BYTES WERE READ
      *  INTO THE BUFFER (TRAILER ERROR).  THIS WOULD OF COURSE
      *  IMPLY A CHECKSUM ERROR.  WHEN CSTATS IS A 0 UPON ENTRY TO
      * THE DRIVER ROUTINES, ~~THE ONLY DATA PRINTED OUT WILL BE
      * THE C AND T ERROR CODES.~~ the next thing
      * will be read in no matter what
      * the name is.
      *     TO USE THE DRIVERS WITH THE PDS-V3A SOFTWARE, WRITE
      * THE PROGRAM:  JSR $FBXX, JMP $FE4F (THE DEBUGGER), WHERE
      * XX IS 00 FOR INITIALIZATION, 2D FOR WRITE AND 8F FOR
      * A READ.  THE LOW MEMORY LOCATIONS ARE INITIALIZED BEFORE
      * JUMPING TO THE ROUTINES.  SETTING THE NOPRNT FLAG WILL
      * HALT THE DISPLAY OF CHARACTERS ON THE SCREEN DURING
      * CASSETTE READ AND WRITE, except for T & C error
      * codes.
      *     NOTE:  ACIANO IS LOCATED AT 18 IN PDS-SYS2A
      *                AND AT LOCATION 30 ON THE FDOS.
0
```

PDS - SYS 2A operating notes

tape format

SYN
SYN
SYN } syncs on these 2 chars
ESC }

HI } 16 bit block length
LO }

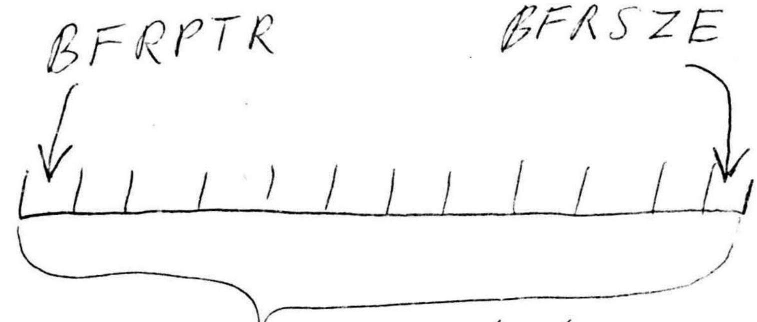
first char } NAME
second char }

DATA

ETB
CHECKSUM
ETX

checks on
ETB (T error)
checksum (C error)

buffer in memory:



outputs data to cassette.

on input, BFRSZE is a value returned by the loader and points to the last character read in

addresses

initialize — FB00
read in a block - FB8F
write out a block - FB2D
Read in a char - FB7C
write out a char - FB60

char passed in accum A

TURN ON - FB75
TURN OFF - FBAE

read & write a block automatically / turn on & off the cassette.

---

as all routines are accessed as subroutines, it is best to write up a simple JSR program:

```
800  BD   } JSR to
     FB   } cassette
802  XX   } routines
     7E   }
     FE   } jump back
     4F   } to debugger
```

INIT 00
WRITE 2D
READ 0F

XX can be set to any desired routine, such as 00 for initialize cassette

---

these routines can be used to do I-O to a teletype or modem ~~[illegible]~~ if a different initialization routine (different values) are used.

It is to be stressed that the reliability depends on the adjustment of the trimmer on the SIM board on reading in data. One way to check this is to read in a long string of the same character from the cassette & adjust until the character reads in perfectly. The best character to use is the 'U'.

---

sample routine for doing above

```
A: JSR $FB7C    Cass. in
   JSR $FCAD    display char on CRT
   TST $F001  } tests
   BPL A      } keyboard
```

skips back until a char. is typed on the keyboard.

INTERMEDIATE UPDATE DOCUMENTS
FOR CASSETTE ROUTINES

THESE TWO SHEETS ARE REPLACEMENT SHEETS FOR PAGES 5 AND 6 FOR THE DOCUMENTATION OF THE SIM BOARD

```
*
*      THE ADDRESSES OF THE ROUTINES FOR CONTROLLING
*      THE CASSETTE ARE AS FOLLOWS:
*
*      INTLZ  - FB00 : INITIALIZES THE ACIA FOR USE.
*      WRTBLK - FB2D : WRITES A  FORMATTED BLOCK TO CASSETTE
*      RDBLK  - FB91 : READS IN A FORMATTED BLOCK OF TAPE
*      WRTMOD - FB2F : WRITES A BLOCK TO A RUNNING CASSETTE
*      RDMOD  - FB93 : READS A BLOCK FROM A RUNNING TAPE.
*      CASOUT - FB62 : WRITES OUT ONTO TAPE THE BYTE IN A.
*      CASIN  - FB7E : READS IN A BYTE INTO ACCUM. A
*      TURNON - FB77 : TURNS ON THE CASSETTE DRIVE.
*      TRNOFF - FBB0 : TURNS OFF THE CASSETTE DRIVE WHEN THE
*                      POINTR TO ACIA IS PASSED IN THE X REG.
*
*     NOTE THAT ALL THE ABOVE ROUTINES ARE SUBROUTINES THAT
*      ARE ENTERED BY A JSR OR BSR CALL.
*
*
*
*
*             USING THE CASSETTE DRIVERS.
*
*      IF THE EPROMS ON THE CPU BOARD ARE THE SYS2N OR THE
* SYS2D PROMS, THE DRIVERS ARE USED BY THE I (INITIALIZE),
* L (LOAD FROM CASSETTE) AND S (STORE ONTO CASSETTE)
* COMMANDS THAT ARE A PART OF THE SYS-2 EXECUTIVE.
*
*      IF THE CPU PROMS BEING USED ARE THE PDS-V3A OR THE
* PDS-V3N OR V3D PROM SETS, THEN THE USER MUST SET UP THE
* POINTERS AND CALL THE ROUTINES HIMSELF, USING THE
* DEBUGGER ON THE CPU PROMS.  (NOTE THAT THE V3A AND V3N
* DEBUGGERS VARY, FOR INSTANCE THE V3A GO COMMAND IS A 'G'
* WHILE THE V3N GO COMMAND IS A 'CNTL G'.  REFER TO THE
* USERS MANUAL FOR DETAILES.)
*      THE FIRST THING TO DO IS OPEN THE LOW MEMORY LOCATIONS
* USED BY THE DRIVERS AS FLAGS AND POINTERS AND INITIALIZE
* THE LOCATIONS.  THE ACIANO, BLKNAM, BFRPTR AND BFRSZE
* SHOULD NOW BE GIVEN VALUES.
*      ACIANO WOULD POINT TO F050 FOR THE FIRST CASSETTE AND
* TO F060 FOR THE SECOND CASSETTE DRIVE.
*      A SIMPLE PROGRAM TO CALL THE ROUTINES SHOULD NOW BE
* WRITTEN.  OPEN LOCATION 900.  TYPE IN THROUGH THE DEBUGGER
* THE INSTRUCTIONS JSR BDXX, JMP FE4F.  XX IS THE SECOND BYTE
* OF THE DESIRED DRIVER ROUTINE. I.E. 00 FOR INITIALLIZATION
* OR 91 FOR READING A BLOCK OR 2D FOR WRITING A BLOCK.  THUS,
* TO INITIALIZE AN ACIA, ACIANO WOULD BE LOADED WITH
* THE ADDRESS OF THE ACIA AND THE USER WOULD THEN JUMP TO THE
* ROUTINE AT 900 BY OPENING 900 AND JUMPING TO IT WITH
* THE 'G' COMMAND.  THE ROUTINE WOULD BE AS FOLLOWS:
*
*CALL 900  BD      JSR
*     901  FB      HI BYTE OF ROUTINE AADDRESS
*     902  00      LOW BYTE OF DRIVER ADDRESS.
*     903  7E      JMP
*     904  FE      ADDRESS OF THE DEBUGGER ON
*     905  4F      THE PDS V3A PROM SET
```

```
*      THE ACIA WOULD NOW BE INITIALIZED AND CONTROL WOULD
* HAVE RETURNED TO THE DEBUGGER.  TO READ IN A BLOCK,
* THE NAME WOULD BE PUT IN BLKNAM AND LOCATION 902 WOULD
* BE CHANGED TO 91.  THE USER WOULD THEN  JUMP TO 900.
*
*      TO TEST THE CASSETTE, SET BFRPTR TO E060 AND BFRSZE
* TO E0DF.  THIS WILL ALLOW THE USER TO WRITE OUT DATA
* FROM THE FOURTH, FIFTH, SIXTH AND SEVENTH LINE OF THE
* CRT DISPLAY AND THEN READ IT BACK ONTO THE DISPLAY.  DATA
* CAN BE TYPED ONTO THE SCREEN BY OPENING A LOCATION WITH
* THE DEBUGGER AND THEN MOVING THE CURSOR AROUND THE SCREEN
* TO CHANGE THE CHARACTERS.  THIS IS POSSIBLE BECAUSE THE
* ROUTINE TO INPUT AN ADDRESS CALLS THE EDITOR FOR INPUT.
*
*      THE DRIVER ROUTINES CAN ALSO BE USED TO PERFORM
* I-O WITH A MODEM OR TELETYPE.  THE MAIN HARDWARE
* DIFFERENCE BETWEEN THE CASSETTE AND MODEM/TTY IS THAT THE
* CASSETTE HAS A DIVIDE BY 16 CLOCK AND IS UNIDIRECTIONAL
* WHILE THE MODEM/TTY HAVE A DIVIDE BY 64 CLOCK AND CAN BE
* BIDIRECTIONAL.  BECAUSE OF THE CLOCK CHANGE THE ACIA MUST
* BE TURNED ON WITH A DIFFERENT VALUE BEFORE THE READ BLOCK
* OR WRITE BLOCK ROUTINES ARE ENTERED.  TO TURN ON THE ACIA
* STORE THE VALUE 'B2' INTO LOCATION F050 OR F060.  ONCE
* IT IS TURNED ON, EITHER CASIN OR CASOUT MAY BE CALLED
* REPEATEDLY OR WRITE MODEM BLOCK (WRTMOD) OR READ MODEM
* (RDMOD) CAN BE CALLED ONCE.  WRTMOD & RDMOD ARE THE SAME
* AS WRTBLK & RDBLK EXCEPT THEY DO NOT TURN ON THE ACIA.  A
* PROGRAM TO READ IN A BLOCK OF DATA FROM A MODEM OR TTY
* WOULD THUS BE (PLACED BEFORE THE 'CALL' ROUTINE):
*      8FA  DE 38  LDX    ACIANO   LOADS ACIA POINTTER.
*      8FC  86 B2  LDA A  #$B2     LOADS STARTUP VALUE.
*      8FE  A7 00  STA A  0,X      PUTS START CODE INTO ACIA.
* LOCATION 902 WOULD NOW BE 2F FOR WRITING AND 93 TO READ.
*      TO READ IN A CHARACTER FROM THE TELETYPE TURN ON THE
* ACIA AND GO TO THE FOLLOWING ROUTINE:
*      TTYIN  JSR    CASIN   READS IN A CHAR FROM KEYBOARD.
*             JMP    CASOUT  TYPES OUT CHAR ON PRINTER.
* THE RS232 SHOULD HAVE BEEN STRAPPED TO HALF DUPLEX.
*
*
*      IT IS TO BE STRESSED THAT  THE RELIABILITY OF THE
* CASSETTE CONTROLLER DEPENDS ON THE ADJUSTMENT OF THE
* TRIMMER ON THE SIM BOARD.  IF THE TRIMMER IS OUT OF
* ADJUSTMENT THE DATA WILL NOT READ IN PROPERLY.  BESIDES
* USING THE OSCILLOSCOPE TO ADJUST THE TRIMMER, IT CAN ALSO
* BE ADJUSTED BY READING IN A STRING OF SINGLE CHARACTERS
* FROM  THE CASSETTE AND ADJUSTING IT UNTIL THE CHARACTERS
* SYNC IN PROPERLY.  THE BEST CHARACTER TO USE IS A STRING
* OF 'U'S.  TO READ IN THE STRING FOR TESTING USE, WRITE A
* LOOP TO GET A CHARACTER FROM THE CASSETTE AND THEN
* DISPLAY THAT CHARACTER.  A SAMPLE ROUTINE TO DO THIS IS:
*    A  JSR  $FB7E    LOADS A WITH CASSETTE CHAR.
*       JSR  $FCAD    PDS-V3A PUTCHR ROUTINE.
*       TST  $F001    TESTS KEYBOARD FOR A KEY.
*       BPL  A        SKIPS BACK IF NO INPUT.
* THE ABOVE ROUTINE WOULD INPUT CHARACCTERS UNTIL A KEY
* ON THE KEYBOARD WAS DEPRESSED.
0
```

THIS IS EARLY VERSION
HAS BUG IN IT

SYS 2 NF COMES NEXT
IN CHRONOLOGY

```
                NAM     PDS-SYS2N
                OPT     O,NOG
*
*               PDS SYSTEM 2N  CASSETTE DRIVERS
*
*               PROGRAMMED BY ERIC JAMESON
*
*
*
*               COPYRIGHT 1976 SPHERE CORPORATION
*               940 N.  400 EA.;  NORTH SALT LAKE, UTAH  84054
*
*               P. O. BOX 213;  BOUNTIFUL, UTAH  84010
*
*
*
*
*               SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
*               DISTRIBUTION AND USE OF THE PDS SOFTWARE.
*               NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
*               WRITTEN PERMISSION OF SPHERE CORP.
*
*
*
*
*
* THE PROGRAM DEVELOPMENT SYSTEM (PDS SYS2N) IS A SET OF
* PROGRAMS RESIDING ON ERASABLE PROGRAMMABLE READ ONLY
* MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS
* SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE
* DEVELOPMENT OF COMPUTER PROGRAMS.
*      TOWARD THIS END, THE 5 PDS EPROMS CONTAIN A CURSOR
* BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING
* AID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16
* BIT MULTIPLY AND DIVIDE, ASCII-TO-BINARY, AND
* BINARY-TO-ASCII ROUTINES, AND ROUTINES TO DO
* INPUT AND OUTPUT TO THE AUDIO CASSETTE.
*
*
*
*      THE SYS2N SOFTWARE IS AN UPGRADE OF THE PDS V3A
* (V3N & V3D) SOFTWARE DESIGNED TO RUN WITH THE CASSETTE
* SYSTEM.  THERE ARE TWO VERSIONS OF THE SYS2 SOFTWARE:
* SYS2N WHICH RUNS WITH THE NEW KEYBOARD AND THE SYS2A
* WHICH RUNS WITH THE ORIGIONAL (KBD/1A) KEYBOARD.  THE
* MAIN DIFFERENCE BETWEEN THE V3A AND SYS2 VERSIONS ARE
* THAT A FIFTH EPROM HAS BEEN ADDED AND THAT THE MINI-
* ASSEMBLER HAS BEEN DELEATED AND REPLACED WITH A SET OF
* COMMANDS TO DO LOADING AND DUMPING OF CASSETTE TAPES.
```

```
                *               MEMORY MAP
                *
        0000    TMP     EQU     $00
        0002    TMP1    EQU     $02
        0004    ARB     EQU     $04        16 BIT ACC. PSEUDO REG B.
        0004    AR3     EQU     $04        _HI BYTE OF ARB.
        0005    AR2     EQU     $05        _LO BYTE OF ARB.
        0006    ARA     EQU     $06        16 BIT ARITH PSEUDO REG A.
        0006    AR1     EQU     $06        _HI BYTE OF ARA.
        0007    AR0     EQU     $07        _LO BYTE OF ARA.
        0008    DIGIT   EQU     $08        BYTE USED BY ASCBIN FOR TMP.
        0009    CSTATS  EQU     $09        CASSETTE I-O STATUS BYTE.
        000A    OUTEND  EQU     $0A        END OF OUTPUT BUFFER TEXT.
        000C    BUFADR  EQU     $0C        START OF I/O BUFFER (PTR)
        000E    BUFEND  EQU     $0E        PTR. TO END OF I/O BUFFR.
        0011    OUTBUF  EQU     $11        START OF OUTPUT BUFFER.
        0013    CASNUM  EQU     $13        PHYSICAL CASSETTE NUMBR.
        0014    SRCADR  EQU     $14        SOURCE FOR TEXT MOVES.
        0016    DSTADR  EQU     $16        DEST. ADDR. FOR TEXT MOVE.
        001A    ENDMEM  EQU     $1A        LAST ADDRES OF REAL MEMORY.
        001C    CSRPTR  EQU     $1C        PTR TO CURSER ON SCREEN.
        001E    BUFPTR  EQU     $1E        TEMP PTR USED BY OUTSTR.
        0020    BUFFLO  EQU     $20        PTR TO END OF LOW EDIT TXT.
        0022    BUFFHI  EQU     $22        PTR TO START OF HI TEXT.
        0024    SCNPTR  EQU     $24        PTR. TO BUFFRD TXT START.
        0026    SRCASM  EQU     $26        PTR TO ASSMBLR SOURCE CODE.
        002A    ONDVAL  EQU     $2A        HAS ASSMBLR OPERND VALUE.
        002C    SYMVAL  EQU     $2C        VALUE PUT IN ASSM. SYMTBL.
        002E    BRKSAV  EQU     $2E        TEMP SAVE FOR BRKPT DATA.
        0030    BRKADR  EQU     $30        ADDDRESS OF BREAKPOINT.
        0032    EDIT    EQU     $32        0 IF EDITOR IS NOT RUNNING.
        0033    BLKNAM  EQU     $33        CASSETTE BLOCK NAME.
        0035    IOBUFF  EQU     $35        I-O BUFFER FOR DEBUGGER.
        0038    ACIANO  EQU     $38        SYS2N CASSETTE ACIA ADDR.
        003A    NOPRNT  EQU     $3A        CASSETTE NAME PRINT FLAG.
        003B    BLKTYP  EQU     $3B        CASSETTE BLOCK TYPE CODE.
        003C    BFRPTR  EQU     $3C        ADDR. OF I-O BUFF. FOR CASS.
        003E    BFRSZE  EQU     $3E        LENGTH OF CASS. BUFFER.
        0040    PCVAL   EQU     $40        PROGRAM COUNTER FOR ASSM.


                *       FOLLOWING ARE VARIABLE VALUES.
                *
        26F0    TIMEA   EQU     $26F0      TIMER COUNTER.
        00B1    ON      EQU     $B1        ACIA VALUE TO TURN ON CASS.
        0051    OFF     EQU     $51        ACIA VALUE TO STOP CASS.
        0003    ETX     EQU     $03        END-OF-TEXT.
        0016    SYN     EQU     $16        SYNCHRONISE.
        0017    ETB     EQU     $17        END-OF-TRANSMISSION-BLOCK.
        001B    ESC     EQU     $1B        EXCAPE TO NONSTANDARD HDR.
        0054    ERR4    EQU     $'T        SET  FOR TRAILER ERROR.
        0043    ERR5    EQU     $'C        SET  FOR CHECKSUM ERR.
        5161    TIME    EQU     20833      TIME CNTR FOR 1/4 SEC.
        0009    TIMCNT  EQU     9          TIMES FOR 2 & 1/4 SECONDS.
0
```

PDS SYSTEM 2N CASSETTE DRIVERS

```
                *
FB00                ORG     $FB00
                *
                *
                *          CASSETTE I-O DRIVERS
                *
                *      THE CASSETTE DRIVERS LOAD AND DUMP A BLOCK OF DATA
                * TO AND FROM THE CASSETTE.  THEY HANDLE BOTH THE HEADER
                * AND TRAILER FORMATTING.
                *      THE DRIVERS ARE SET UP AR A SET OF SUBROUTINES
                * CALLED BY THE EXECUTIVE OR THE USER'S PROGRAMS.
                *
                *
                *
                *       LOW MEMORY ADDRESSES USED BY THE DRIVERS ARE:
                *
                *       CSTATS  AT  09    USED TO STORE INPUT ERR CODE.
                *       BLKNAM  AT  33    2 CHAR. NAME OF BLOCK.
                *       ACIANO  AT  38    PTR. TO THE ACIA CURRENTLY USED.
                *       NOPRNT  AT  3A    PRINT FLAG FOR BLOCK NAME.
                *       BFRPTR  AT  3C    START OF CASSETTE I-O BUFFER.
                *       BFRSZE  AT  3E    END OF CASSETTE DATA BUFFER.
                *
                *
                *      THE DRIVERS CAN BE RUN FROM THE PDS-SYS2N SOFTWARE
                * SYSTEM, WHICH CONTAIN THE CASSETTE LOAD AND DUMP
                * COMMANDS IN THE EXEC, OR FROM THE PDS-V3A, WHERE THE
                * DRIVERS ARE CALLED THROUGH THE DEBUGGER.
                *
                *      THE DRIVERS DISPLAY PERTINENT DATA ON THE SCREEN
                * WHEN WRITING OR READING FROM THE CASSETTES.  ON A WRITE,
                * THE CHARACTERS BEING WRITTEN ONTO THE CASSETTE ARE
                * DISPLAYED IN THE SECOND CHARACTER POSITION FROM THE
                * UPPER RIGHT HAND CORNER OF THE SCREEN.  ON A READ, THE
                * CHARACTERS BEING READ IN ARE DISPLAYED IN THE UPPER
                * RIGHT HAND CORNER OF THE SCREEN.  THE NAME OF THE BLOCK
                * CURRENTLY BEING READ OR SEARCHED OVER IS DISPLAYED ON THE
                * RIGHT HAND SIDE OF THE SECOND LINE.  THE ERROR CODE FOR
                * A READ IS DISPLAYED ON THE RIGHT SIDE OF THE THIRD LINE.
                * NO CHANGE IN CHARACTER MEANS THAT THE READ WAS O.K.
                * A "C" MEANS THERE WAS A CHECKSUM ERROR ON THE BLOCK.
                * A "T" MEANS THAT THE WRONG NUMBER OF BYTES WERE READ
                *  INTO THE BUFFER (TRAILER ERROR).  THIS WOULD OF COURSE
                *  IMPLY A CHECKSUM ERROR ALONG WITH THE TRAILER ERROR.
                *
                *      THE CSTATS (CASSETTE STATUS) BYTE WILL CONTAIN A 0
                * IF THE BLOCK READ IN WAS O.K.   IF IT WAS A BAD READ, IT
                * WILL CONTAIN A 54 FOR TRAILER ERRROR OR A 43 FOR A
                * CHECKSUM ERROR CODE UPON EXIT FROM THE ROUTINE.
                *
                *      SETTING THE NOPRNT (NO PRINTING) FLAG TO A 0 WILL
                * STOP THE DISPLAY OF CHARACTERS ON THE SCREEN DURING
                * CASSETTE READ AND WRITE, EXCEPT FOR THE T & C ERROR
                * CODES, WHICH ARE ALWAYS DISPLAYED WHEN THEY OCCUR.
```

```
*      IF THE FIRST BYTE OF BLKNAM (BLOCK NAME) IS A 0
* WHEN THE READ BLOCK ROUTINE IS ENTERED, THE NEXT BLOCK
* WILL BE READ FROM TAPE NO MATTER WHAT THE NAME OF THE
* TAPE BLOCK IS.  ON THE SYS2N EXEC, A CONTROL SPACE
* CHARACTER CAN BE TYPED IN AS THE FIRST CHARACTER OF THE
* NAME IN THE LOAD BLOCK COMMAND.  THUS, A (CNTL L)(CNTL
* SPACE)(X) WOULD READ IN THE NEXT BLOCK ON THE TAPE.
*
*
*
*      THE TAPE FORMAT IS:
*        SYN
*        SYN
*        SYN
*        ESC
*        HI BYTE OF 16 BIT BLOCK LENGTH
*        LOW BYTE OF BLOCK LENGTH
*        FIRST CHAR OF BLOCK NAME
*        SECOND CHARACTER OF NAME
*        DATA
*        .
*        .
*        .
*        DATA
*        ETB
*        CHECKSUM
*        CHECKSUM
*        CHECKSUM
*        CHECKSUM
*
*
*
*      THE CHECKSUM IS CALCULATED FROM THE DATA, WHICH
* IS READ IN FROM THE CASSETTE BUFFER RESIDING IN MEMORY.
*
*
*      THE FORMAT FOR THE CASSETTE BUFFER IS:
*
*        BFRPTR                          BFRSZE
*        |                                    |
*        v                                    v
*       |C|A|S|S|E|T|T|E| | | |B|U|F|F|E|R|
*
*
*      WHEN DATA IS READ IN, THE READ ROUTINE SETS 'BFRSZE'
* TO POINT TO THE LAST CHARACTER READ INTO THE BUFFER.
* NOTE THAT THERE IS NO OVERFLOW CHECK WHEN DATA IS READ
* INTO MEMORY.  ON OUTPUT TO THE CASSETTE, THE BLOCK
* LENGTH IS CALCULATED FROM THE BFRPTR AND BFRSZE POINTERS.
*
*
*      THE READ BLOCK AND WRITE BLOCK ROUTINES AUTOMATICALLY
* TURN ON AND OFF THE CASSETTE.
```

```
*
*     THE ADDRESSES OF THE ROUTINES FOR CONTROLLING
*     THE CASSETTE ARE AS FOLLOWS:
*
*     INTLZ  - FB00 : INITIALIZES THE ACIA FOR USE.
*     WRTBLK - FB2D : WRITES A  FORMATTED BLOCK TO CASSETTE
*     RDBLK  - FB91 : READS IN A FORMATTED BLOCK OF TAPE
*     WRTMOD - FB2F : WRITES A BLOCK TO A RUNNING CASSETTE
*     RDMOD  - FB93 : READS A BLOCK FROM A RUNNING TAPE.
*     CASOUT - FB62 : WRITES OUT ONTO TAPE THE BYTE IN A.
*     CASIN  - FB7E : READS IN A BYTE INTO ACCUM. A
*     TURNON - FB77 : TURNS ON THE CASSETTE DRIVE.
*     TRNOFF - FBB0 : TURNS OFF THE CASSETTE DRIVE WHEN THE
*                     POINTR TO ACIA IS PASSED IN THE X REG.
*
*    NOTE THAT ALL THE ABOVE ROUTINES ARE SUBROUTINES THAT
*     ARE ENTERED BY A JSR OR BSR CALL.
*
*
*
*
*            USING THE CASSETTE DRIVERS.
*
*     IF THE EPROMS ON THE CPU BOARD ARE THE SYS2N OR THE
* SYS2D PROMS, THE DRIVERS ARE USED BY THE I (INITIALIZE),
* L (LOAD FROM CASSETTE) AND S (STORE ONTO CASSETTE)
* COMMANDS THAT ARE A PART OF THE SYS-2 EXECUTIVE.
*
*     IF THE CPU PROMS BEING USED ARE THE PDS-V3A OR THE
* PDS-V3N OR V3D PROM SETS, THEN THE USER MUST SET UP THE
* POINTERS AND CALL THE ROUTINES HIMSELF, USING THE
* DEBUGGER ON THE CPU PROMS.  (NOTE THAT THE V3A AND V3N
* DEBUGGERS VARY, FOR INSTANCE THE V3A GO COMMAND IS A 'G'
* WHILE THE V3N GO COMMAND IS A 'CNTL G'.  REFER TO THE
* USERS MANUAL FOR DETAILES.)
*     THE FIRST THING TO DO IS OPEN THE LOW MEMORY LOCATIONS
* USED BY THE DRIVERS AS FLAGS AND POINTERS AND INITIALIZE
* THE LOCATIONS.  THE ACIANO, BLKNAM, BFRPTR AND BFRSZE
* SHOULD NOW BE GIVEN VALUES.
*     ACIANO WOULD POINT TO F050 FOR THE FIRST CASSETTE AND
* TO F060 FOR THE SECOND CASSETTE DRIVE.
*     A SIMPLE PROGRAM TO CALL THE ROUTINES SHOULD NOW BE
* WRITTEN.  OPEN LOCATION 900.  TYPE IN THROUGH THE DEBUGGER
* THE INSTRUCTIONS JSR BDXX, JMP FE4F.  XX IS THE SECOND BYTE
* OF THE DESIRED DRIVER ROUTINE, I.E. 00 FOR INITIALLIZATION
* OR 91 FOR READING A BLOCK OR 2D FOR WRITING A BLOCK.  THUS,
* TO INITIALIZE AN ACIA, ACIANO WOULD BE LOADED WITH
* THE ADDRESS OF THE ACIA AND THE USER WOULD THEN JUMP TO THE
* ROUTINE AT 900 BY OPENING 900 AND JUMPING TO IT WITH
* THE 'G' COMMAND.  THE ROUTINE WOULD BE AS FOLLOWS:
*
*CALL 900  BD     JSR
*     901  FB     HI BYTE OF ROUTINE AADDRESS
*     902  00     LOW BYTE OF DRIVER ADDRESS.
*     903  7E     JMP
*     904  FE     ADDRESS OF THE DEBUGGER ON
*     905  4F     THE PDS V3A PROM SET
0
```

```
*     THE ACIA WOULD NOW BE INITIALIZED AND CONTROL WOULD
* HAVE RETURNED TO THE DEBUGGER.  TO READ IN A BLOCK,
* THE NAME WOULD BE PUT IN BLKNAM AND LOCATION 902 WOULD
* BE CHANGED TO 91.  THE USER WOULD THEN  JUMP TO 900.
*
*     TO TEST THE CASSETTE, SET BFRPTR TO E060 AND BFRSZE
* TO E0DF.  THIS WILL ALLOW THE USER TO WRITE OUT DATA
* FROM THE FOURTH, FIFTH, SIXTH AND SEVENTH LINE OF THE
* CRT DISPLAY AND THEN READ IT BACK ONTO THE DISPLAY.  DATA
* CAN BE TYPED ONTO THE SCREEN BY OPENING A LOCATION WITH
* THE DEBUGGER AND THEN MOVING THE CURSOR AROUND THE SCREEN
* TO CHANGE THE CHARACTERS.  THIS IS POSSIBLE BECAUSE THE
* ROUTINE TO INPUT AN ADDRESS CALLS THE EDITOR FOR INPUT.
*
*     THE DRIVER ROUTINES CAN ALSO BE USED TO PERFORM
* I-O WITH A MODEM OR TELETYPE.  THE MAIN HARDWARE
* DIFFERENCE BETWEEN THE CASSETTE AND MODEM/TTY IS THAT THE
* CASSETTE HAS A DIVIDE BY 16 CLOCK AND IS UNIDIRECTIONAL
* WHILE THE MODEM/TTY HAVE A DIVIDE BY 64 CLOCK AND CAN BE
* BIDIRECTIONAL.  BECAUSE OF THE CLOCK CHANGE THE ACIA MUST
* BE TURNED ON WITH A DIFFERENT VALUE BEFORE THE READ BLOCK
* OR WRITE BLOCK ROUTINES ARE ENTERED.  TO TURN ON THE ACIA
* STORE THE VALUE 'B2' INTO LOCATION F050 OR F060.  ONCE
* IT IS TURNED ON, EITHER CASIN OR CASOUT MAY BE CALLED
* REPEATEDLY OR WRITE MODEM BLOCK (WRTMOD) OR READ MODEM
* (RDMOD) CAN BE CALLED ONCE.  WRTMOD & RDMOD ARE THE SAME
* AS WRTBLK & RDBLK EXCEPT THEY DO NOT TURN ON THE ACIA.  A
* PROGRAM TO READ IN A BLOCK OF DATA FROM A MODEM OR TTY
* WOULD THUS BE (PLACED BEFORE THE 'CALL' ROUTINE):
*      8FA  DE 38  LDX    ACIANO    LOADS ACIA POINTTER.
*      8FC  86 B2  LDA A  #$B2      LOADS STARTUP VALUE.
*      8FE  A7 00  STA A  0,X       PUTS START CODE INTO ACIA.
* LOCATION 902 WOULD NOW BE 2F FOR WRITING AND 93 TO READ.
*     TO READ IN A CHARACTER FROM THE TELETYPE TURN ON THE
* ACIA AND GO TO THE FOLLOWING ROUTINE:
*     TTYIN  JSR    CASIN    READS IN A CHAR FROM KEYBOARD.
*            JMP    CASOUT   TYPES OUT CHAR ON PRINTER.
* THE RS232 SHOULD HAVE BEEN STRAPPED TO HALF DUPLEX.
*
*
*     IT IS TO BE STRESSED THAT  THE RELIABILITY OF THE
* CASSETTE CONTROLLER DEPENDS ON THE ADJUSTMENT OF THE
* TRIMMER ON THE SIM BOARD.  IF THE TRIMMER IS OUT OF
* ADJUSTMENT THE DATA WILL NOT READ IN PROPERLY.  BESIDES
* USING THE OSCILLOSCOPE TO ADJUST THE TRIMMER, IT CAN ALSO
* BE ADJUSTED BY READING IN A STRING OF SINGLE CHARACTERS
* FROM  THE CASSETTE AND ADJUSTING IT UNTIL THE CHARACTERS
* SYNC IN PROPERLY.  THE BEST CHARACTER TO USE IS A STRING
* OF 'U'S.  TO READ IN THE STRING FOR TESTING USE, WRITE A
* LOOP TO GET A CHARACTER FROM THE CASSETTE AND THEN
* DISPLAY THAT CHARACTER.  A SAMPLE ROUTINE TO DO THIS IS:
*    A  JSR  $FB7E   LOADS A WITH CASSETTE CHAR.
*       JSR  $FCAD   PDS-V3A PUTCHR ROUTINE.
*       TST  $F001   TESTS KEYBOARD FOR.A KEY.
*       BPL  A       SKIPS BACK IF NO INPUT.
* THE ABOVE ROUTINE WOULD INPUT CHARACCTERS UNTIL A KEY
* ON THE KEYBOARD WAS DEPRESSED.
0
```

B2 will produce a signel of eight data bits and 2 stop bits. If the TTY operates on a different code, look up the proper initialization value in the ACIA section of the chip description appendex.

```
                  *       INTLZ INITIALIZES THE ACIA CONTROLLER FOR
                  *       A SPECIFIC TAPE UNIT AT MOUNT TIME.
                  *
FB00 DE 38        INTLZ  LDX     ACIANO   X GETS ACIA ADDRESS.
FB02 86 13               LDA A   #$13     RESETS THE ACIA.
FB04 A7 00               STA A   0,X      .
FB06 86 51               LDA A   #OFF     SETS ACIA TO
FB08 A7 00               STA A   0,X      /16 2 STOPS BIT FORMAT.
FB0A 39                  RTS

                  *       WRTHDR FORMATS THE HEADER ON THE TAPE.
                  *
FB0B 86 16        WRTHDR LDA A   #SYN     PUTS SYNC CHARS ONTO TAPE.
FB0D 8D 53               BSR     CASOUT   .
FB0F 8D 51               BSR     CASOUT   .
FB11 8D 4F               BSR     CASOUT   .
FB13 86 1B               LDA A   #ESC     .
FB15 8D 4B               BSR     CASOUT   .
FB17 96 3E               LDA A   BFRSZE   FOLLOWING OUTPUTS LENGTH.
FB19 D6 3F               LDA B   BFRSZE+1 LOADS LO BYTE OF END PTR.
FB1B D0 3D               SUB B   BFRPTR+1 SUBS LO BYTE OF BEGIN PTR.
FB1D 92 3C               SBC A   BFRPTR   SUBS HI BYTE OF START PTR.
FB1F 8D 41               BSR     CASOUT   OUTPUTS HI LENGTH BYTE.
FB21 17                  TBA              LOADS LO BYTE OF LENGTH.
FB22 8D 3E               BSR     CASOUT   OUTPUTS LOW LENGTH BYTE.
FB24 96 33               LDA A   BLKNAM   PUTS OUT NAME OF BLOCK.
FB26 8D 3A               BSR     CASOUT   .
FB28 96 34               LDA A   BLKNAM+1 .
FB2A 8D 36               BSR     CASOUT   PUTS OUT LAST OF NAME.
FB2C 39                  RTS              RETURNS BACK TO WRTBLK.

                  *       WRTBLK WRITES OUT A BLOCK OF DATA TO THE CASSETTE.
                  *
FB2D 8D 48        WRTBLK BSR     TURNON   TURNS ON THE CASSETTE.
                  *        FOLLOWING WAITES FOR CASSETTE TO GET UP TO SPEED.
FB2F C6 09        WRTMOD LDA B   #TIMCNT  LOADS TIME LOOP COUNTER.
FB31 CE 5161      TIME1  LDX     #TIME    MASTER TIME LOOP (1/4 SEC).
FB34 09           TIME2  DEX              COUNTS CYCLES OF LOOP.
FB35 26 FD               BNE     TIME2    TESTS FOR FIRST TIME OUT.
FB37 5A                  DEC B   -        COUNTS TIMES IN LOOP.
FB38 26 F7               BNE     TIME1    SKIPS BACK UNTIL DONE.
                  *        THE TIME LOOP IS NOW FINISHED.
FB3A 8D CF               BSR     WRTHDR   WRITES HEADER ON THE TAPE.
FB3C 8D 10               BSR     WRTBFR   WRITES OUT BUFFER DATA.
                  *        FOLLOWING WRITES THE TRAILER OUT ONTO THE TAPE.
FB3E 86 17        WRTTLR LDA A   #ETB     OUTPUTS END-OF-BLOCK CHAR.
FB40 8D 20               BSR     CASOUT   ETB IS DISPLAYED AS A "W".
FB42 17                  TBA              A GETS CHECKSUM FROM B.
FB43 8D 1D               BSR     CASOUT   OUTPUTS THE CHECKSUM.
FB45 8D 1B               BSR     CASOUT   OUTPUTS TRAILER FILLER
FB47 8D 19               BSR     CASOUT   BYTES.
FB49 8D 17               BSR     CASOUT   .
                  *        END OF TRAILER WRITING ROUTINE.
FB4B 8D 63               BSR     TRNOFF   HALTS CASSETTE DRIVE.
FB4D 39                  RTS              -
```

```
                    *       WRTBFR WRITES OUT THE CONTENTS  OF THE
                    *       BUFFER ONTO THE CASSETTE TAPE.
                    *
                    *
FB4E 5F             WRTBFR CLR B    -           INIT CHECKSUM COUNT.
FB4F DE 3C                 LDX     BFRPTR      ACIA CONTROL MASK.
FB51 A6 00          WBFR1  LDA A   0,X         LOADS CHAR.  FROM BUFFER.
FB53 DF 00                 STX     TMP         SAVES BUFFER PTR.
FB55 8D 0B                 BSR     CASOUT      PUTS CHAR ONTO CASSETTE.
FB57 1B                    ABA                 A GETS A+B.
FB58 16                    TAB                 B GETS A+B.
FB59 DE 00                 LDX     TMP         RESTORES PTR.  INTO BUFFR.
FB5B 9C 3E                 CPX     BFRSZE      TESTS IF BUFFER EMPTY.
FB5D 27 17                 BEQ     CSOEXT      EXITS WHEN EMPTY.
FB5F 08                    INX                 INC POINTER.
FB60 26 EF                 BNE     WBFR1       SKIPS BACK IF CHARS.  LEFT.
```

```
                    *       CASOUT TAKES THE CHAR IN A AND PUTS IT OUT
                    *       ONTO THE CASSETTE TAPE.
                    *
                    *
FB62 36             CASOUT PSH A   .           SAVES CHAR TO READ OUT.
FB63 DE 38                 LDX     ACIANO      X GETS PHYSICAL ACIA ADDR.
FB65 86 02                 LDA A   #2          LOADS CONTROL TEST BITS.
FB67 A5 00          CAS01  BIT A   0,X         TESTS IF ACIA BUFFER EMPTY.
FB69 27 FC                 BEQ     CAS01       LOOPS BACK UNTIL READY.
FB6B 32                    PUL A   .           GETS ORIG CHAR.
FB6C A7 01                 STA A   1,X         STORS CHAR INTO ACIA BUFFER.
FB6E 7D 003A               TST     NOPRNT      TESTS IF PRINTOUT ALLOWED.
FB71 27 03                 BEQ     CSOEXT      SKIPS PRINTING IF A 0.
FB73 B7 E01E               STA A   $E01E       DISPLAYS CHAR ON SCREEN.
FB76 39             CSOEXT RTS
```

```
                    *        TURNON TURNS ON THE CASSETTE DRIVE.
                    *
                    *
FB77 DE 38          TURNON LDX     ACIANO      LOADS CASSETTE ACIA ADDRESS.
FB79 86 B1                 LDA A   #ON         .
FB7B A7 00                 STA A   0,X         TURNS ACIA ON.
FB7D 39                    RTS
```

```
                    *       CASIN READS IN A CHARACTER FROM THE CASSETTE TAPE
                    *       INTO THE A ACCUMULATOR.
                    *
                    *
FB7E DE 38          CASIN   LDX     ACIANO    X GETS THE ACIA ADDRESS.
FB80 86 01                  LDA A   #1        LOADS TEST BITS.
FB82 A5 00          CASIN1  BIT A   0,X       TESTS IF ACCIA BUFFR FULL.
FB84 27 FC                  BEQ     CASIN1    SKIPS BACK IF NOT IN YET.
FB86 A6 01                  LDA A   1,X       LOADS IN CHAR FROM TAPE.
FB88 7D 003A                TST     NOPRNT    TESTS IF PRINT IS OFF.
FB8B 27 03                  BEQ     CINEXT    SKIPS DISPLAYING IF 0.
FB8D B7 E01F                STA A   $E01F     DISPLAYS CHR ON TV.
FB90 39             CINEXT  RTS               .



                    *       RDBLK READS IN A BLOCK FROM THE CASSETTE
                    *       TAPE INTO BUFFER MEMORY.
                    *
                    *
FB91 8D E4          RDBLK   BSR     TURNON    TURNS ON TAPE DRIVE.
FB93 8D 20          RDMOD   BSR     RDHDR     READS IN THE HEADER.
FB95 8D 55                  BSR     RDBFR     READS IN DATA INTO BUFFER.
                    *        FOLLOWING READS IN TRAILER AND CHECKS CHECKSUM.
FB97 4F             RDTLR   CLR A   -         LOADS A 0 FOR A GOOD READ.
FB98 97 09                  STA A   CSTATS    SETS STATUS BYTE TO NO ERR.
FB9A 8D E2                  BSR     CASIN     INPUTS END-OF-BLOCK CHAR.
FB9C 81 17                  CMP A   #ETB      ETB DISPLAYS AS A "W".
FB9E 27 04                  BEQ     RTLR1     SKIPS IF NO ETB ERROR.
FBA0 86 54                  LDA A   #ERR4     LOADS TRAILER ERROR CODE.
FBA2 20 07                  BRA     RTLR2     SKIPS TO STORE ERROR CODE.
FBA4 8D D8          RTLR1   BSR     CASIN     READS IN CHECKSUM.
FBA6 11                     CBA               TESTS CHECKSUM.
FBA7 27 07                  BEQ     TRNOFF    SKIPS IF OK.
FBA9 86 43                  LDA A   #ERR5     SETS CHECKSUM ERROR CODE.
FBAB 97 09          RTLR2   STA A   CSTATS    SETS ERROR STATUS BYTE.
FBAD B7 E05F                STA A   $E05F     DISP.  ERR CODE ON SCREEN.
                    *        END OF TRAILER READ IN.
                    *        FOLLOWING TURNS OFF THE CASSETTE DRIVE.
FBB0 86 51          TRNOFF  LDA A   #OFF      LOADS COMMAND TO TURN
FBB2 A7 00                  STA A   0,X       OFF ACIA CASSETTE DRIVE.
FBB4 39                     RTS               .
```

```
                      *       RDHDR FINDS THE START OF THE BLOCK ON THE TAPE,
                      *       FIGURES THE BUFFER END AND CHECKS THE NAME.
                      *
                      *
FBB5 8D C7    RDHDR   BSR     CASIN      FOLLOWING FINDS THE
FBB7 81 16            CMP A   #SYN       START OF THE HEADER.
FBB9 26 FA            BNE     RDHDR      .
FBBB 8D C1            BSR     CASIN      TESTS FOR HEADR CHAR.
FBBD 81 1B            CMP A   #ESC       START-OF-HEADER.
FBBF 26 F4            BNE     RDHDR      GOES BACK IF NOT GOOD HDR.
FBC1 8D BB            BSR     CASIN      READS IN HI BYTE OF LEN.
FBC3 16               TAB                SAVES HI LENGTH BYTE.
FBC4 8D B8            BSR     CASIN      INPUTS LO SIZE (LEN) BYTE.
FBC6 9B 3D            ADD A   BFRPTR+1   FORMS POINTER TO THE
FBC8 D9 3C            ADC B   BFRPTR     TOP BYTE OF THE BUFFER.
FBCA 97 3F            STA A   BFRSZE+1   SAVES THE HI BUFF. PTR.
FBCC D7 3E            STA B   BFRSZE     TO THE CASSETTE BUFFER.
FBCE 8D AE            BSR     CASIN      READS IN BLOCK NAME.
FBD0 36               PSH A   -          SAVES FIRST CHAR OF NAME.
FBD1 8D AB            BSR     CASIN      READ IN SECOND CHAR INTO A.
FBD3 D6 3A            LDA B   NOPRNT     TESTS IF PRINT IS OFF.
FBD5 33               PUL B   -          RESTORES FIRST NAME CHAR.
FBD6 27 06            BEQ     RHDR1      SKIPS IF PRINT FLAG IS 0.
FBD8 F7 E03E          STA B   $E03E      DISPLAYS BLOCK NAME ON
FBDB B7 E03F          STA A   $E03F      THE CRT SCREEN.
FBDE 7D 0033  RHDR1   TST     BLKNAM     TESTS IF NAME IS CHECKED.
FBE1 27 08            BEQ     RHDR2      SKIPS IF NO NAME CHECK.
FBE3 D1 33            CMP B   BLKNAM     TESTS FIRST CHAR OF NAME.
FBE5 26 CE            BNE     RDHDR      SKIPS BACK IF BAD NAME
FBE7 91 34            CMP A   BLKNAM+1   TESTS SECOND NAME CHAR.
FBE9 26 CA            BNE     RDHDR      SKIPS BACK IF BAD NAME.
FBEB 39       RHDR2   RTS                .
```

```
                      *       RDBFR READS DATA INTO THE MEMORY BUFFER FROM
                      *       THE CASSETTE.
                      *
                      *
FBEC 5F       RDBFR   CLR B   -          INIT B FOR CHECKSUM.
FBED DE 3C            LDX     BFRPTR     LOADS START OF BUFFER.
FBEF DF 00    RBFR1   STX     TMP        .
FBF1 8D 8B            BSR     CASIN      A GETS CHAR READ IN.
FBF3 DE 00            LDX     TMP        X GETS BUFFER PTR.
FBF5 A7 00            STA A   0,X        STORS CHAR INTO BUFFER.
FBF7 1B               ABA                A GETS A+B.
FBF8 16               TAB                B GETS A.
FBF9 9C 3E            CPX     BFRSZE     TESTS IF BUFFER FULL.
FBFB 27 EE            BEQ     RHDR2      SKIPS TO EXIT IF ALL IS IN.
FBFD 08               INX                INC TO NEXT CHAR POSITIN.
FBFE 26 EF            BNE     RBFR1      GOES BACK IF ANY LEFT.
```

```
         NAM     PDS-SYS2N
         OPT     O,NOG
*
*        PDS SYSTEM 2N  CASSETTE DRIVERS  (SYS2NF)
*
*        PROGRAMMED BY ERIC JAMESON
*
*
*
*        COPYRIGHT 1976 SPHERE CORPORATION
*        940 N.   400 EA.;   NORTH SALT LAKE, UTAH  84054
*
*        P. O.  BOX 213;  BOUNTIFUL, UTAH  84010
*
*
*
*
*        SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
*        DISTRIBUTION AND USE OF THE PDS SOFTWARE.
*        NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
*        WRITTEN PERMISSION OF SPHERE CORP.
*
*
*
*
*
* THE PROGRAM DEVELOPMENT SYSTEM (PDS SYS2N) IS A SET OF
* PROGRAMS RESIDING ON ERASABLE PROGRAMMABLE READ ONLY
* MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS
* SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE
* DEVELOPMENT OF COMPUTER PROGRAMS.
*      TOWARD THIS END, THE 5 PDS EPROMS CONTAIN A CURSOR
* BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING
* AID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16
* BIT MULTIPLY AND DIVIDE, ASCII-TO-BINARY, AND
* BINARY-TO-ASCII ROUTINES, AND ROUTINES TO DO
* INPUT AND OUTPUT TO THE AUDIO CASSETTE.
*
*
*
*      THE SYS2N SOFTWARE IS AN UPGRADE OF THE PDS V3A
* (V3N & V3D) SOFTWARE DESIGNED TO RUN WITH THE CASSETTE
* SYSTEM.  THERE ARE TWO VERSIONS OF THE SYS2 SOFTWARE,
* SYS2N WHICH RUNS WITH THE NEW KEYBOARD AND THE SYS2A
* WHICH RUNS WITH THE ORIGIONAL (KBD/1A) KEYBOARD.  THE
* MAIN DIFFERENCE BETWEEN THE V3A AND SYS2 VERSIONS ARE
* THAT A FIFTH EPROM HAS BEEN ADDED AND THAT THE MINI-
* ASSEMBLER HAS BEEN DELEATED AND REPLACED WITH A SET OF
* COMMANDS TO DO LOADING AND DUMPING OF CASSETTE TAPES
```

The SYS2NF prom is a version of the SYS2N cassette prom with a software bug fixed. This bug would cause the next block of a multiblock read to be skipped if the checksum on the preceeding block was a 16 and the tape had not been previously used. The change is on page 1Ø where the ESC test and branch now branches to RDHDR1 instead of RDHDR.

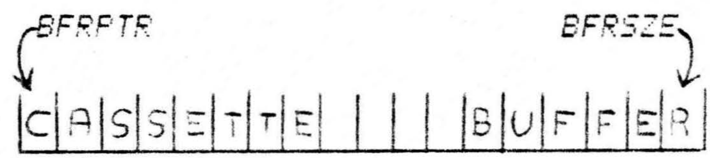
```
                *               MEMORY MAP
                *
        0000    TMP     EQU     $00
        0002    TMP1    EQU     $02
        0004    ARB     EQU     $04         16 BIT ACC. PSEUDO REG B.
        0004    AR3     EQU     $04         _HI BYTE OF ARB.
        0005    AR2     EQU     $05         _LO BYTE OF ARB.
        0006    ARA     EQU     $06         16 BIT ARITH PSEUDO REG A.
        0006    AR1     EQU     $06         _HI BYTE OF ARA.
        0007    AR0     EQU     $07         _LO BYTE OF ARA.
        0008    DIGIT   EQU     $08         BYTE USED BY ASCBIN FOR TMP.
        0009    CSTATS  EQU     $09         CASSETTE I-O STATUS BYTE.
        000A    OUTEND  EQU     $0A         END OF OUTPUT BUFFER TEXT.
        000C    BUFADR  EQU     $0C         START OF I/O BUFFER (PTR)
        000E    BUFEND  EQU     $0E         PTR. TO END OF I/O BUFFR.
        0011    OUTBUF  EQU     $11         START OF OUTPUT BUFFER.
        0013    CASNUM  EQU     $13         PHYSICAL CASSETTE NUMBR.
        0014    SRCADR  EQU     $14         SOURCE FOR TEXT MOVES.
        0016    DSTADR  EQU     $16         DEST. ADDR. FOR TEXT MOVE.
        001A    ENDMEM  EQU     $1A         LAST ADDRES OF REAL MEMORY.
        001C    CSRPTR  EQU     $1C         PTR TO CURSER ON SCREEN.
        001E    BUFPTR  EQU     $1E         TEMP PTR USED BY OUTSTR.
        0020    BUFFLO  EQU     $20         PTR TO END OF LOW EDIT TXT.
        0022    BUFFHI  EQU     $22         PTR TO START OF HI TEXT.
        0024    SCNPTR  EQU     $24         PTR. TO BUFFRD TXT START.
        0026    SRCASM  EQU     $26         PTR TO ASSMBLR SOURCE CODE.
        002A    ONDVAL  EQU     $2A         HAS ASSMBLR OPERND VALUE.
        002C    SYMVAL  EQU     $2C         VALUE PUT IN ASSM. SYMTBL.
        002E    BRKSAV  EQU     $2E         TEMP SAVE FOR BRKPT DATA.
        0030    BRKADR  EQU     $30         ADDDRESS OF BREAKPOINT.
        0032    EDIT    EQU     $32         0 IF EDITOR IS NOT RUNNING.
        0033    BLKNAM  EQU     $33         CASSETTE BLOCK NAME.
        0035    IOBUFF  EQU     $35         I-O BUFFER FOR DEBUGGER.
        0038    ACIANO  EQU     $38         SYS2N CASSETTE ACIA ADDR.
        003A    NOPRNT  EQU     $3A         CASSETTE NAME PRINT FLAG.
        003B    BLKTYP  EQU     $3B         CASSETTE BLOCK TYPE CODE.
        003C    BFRPTR  EQU     $3C         ADDR. OF I-O BUFF. FOR CASS.
        003E    BFRSZE  EQU     $3E         LENGTH OF CASS. BUFFER.
        0040    PCVAL   EQU     $40         PROGRAM COUNTER FOR ASSM.


                *       FOLLOWING ARE VARIABLE VALUES.
                *
        26F0    TIMEA   EQU     $26F0       TIMER COUNTER.
        00B1    ON      EQU     $B1         ACIA VALUE TO TURN ON CASS.
        0051    OFF     EQU     $51         ACIA VALUE TO STOP CASS.
        0003    ETX     EQU     $03         END-OF-TEXT.
        0016    SYN     EQU     $16         SYNCHRONISE.
        0017    ETB     EQU     $17         END-OF-TRANSMISSION-BLOCK.
        001B    ESC     EQU     $1B         EXCAPE TO NONSTANDARD HDR.
        0054    ERR4    EQU     $'T         SET FOR TRAILER ERROR.
        0043    ERR5    EQU     $'C         SET FOR CHECKSUM ERR.
        5161    TIME    EQU     20833       TIME CNTR FOR 1/4 SEC.
        0009    TIMCNT  EQU     9           TIMES FOR 2 & 1/4 SECONDS.
0
```

PDS SYSTEM 2N CASSETTE DRIVERS

```
                        *
FB00                            ORG     $FB00
                        *
                        *
                        *          CASSETTE I-O DRIVERS
                        *
                        *     THE CASSETTE DRIVERS LOAD AND DUMP A BLOCK OF DATA
                        * TO AND FROM THE CASSETTE.  THEY HANDLE BOTH THE HEADER
                        * AND TRAILER FORMATTING.
                        *     THE DRIVERS ARE SET UP AR A SET OF SUBROUTINES
                        * CALLED BY THE EXECUTIVE OR THE USER'S PROGRAMS.
                        *
                        *
                        *
                        *      LOW MEMORY ADDRESSES USED BY THE DRIVERS ARE:
                        *
                        *      CSTATS  AT  09  . USED TO STORE INPUT ERR CODE.
                        *      BLKNAM  AT  33    2 CHAR. NAME OF BLOCK.
                        *      ACIANO  AT  38    PTR. TO THE ACIA CURRENTLY USED.
                        *      NOPRNT  AT  3A    PRINT FLAG FOR BLOCK NAME.
                        *      BFRPTR  AT  3C    START OF CASSETTE I-O BUFFER.
                        *      BFRSZE  AT  3E    END OF CASSETTE DATA BUFFER.
                        *
                        *
                        *     THE DRIVERS CAN BE RUN FROM THE PDS-SYS2N SOFTWARE
                        * SYSTEM, WHICH CONTAIN THE CASSETTE LOAD AND DUMP
                        * COMMANDS IN THE EXEC, OR FROM THE PDS-V3A, WHERE THE
                        * DRIVERS ARE CALLED THROUGH THE DEBUGGER.
                        *
                        *     THE DRIVERS DISPLAY PERTINENT DATA ON THE SCREEN
                        * WHEN WRITING OR READING FROM THE CASSETTES.  ON A WRITE,
                        * THE CHARACTERS BEING WRITTEN ONTO THE CASSETTE ARE
                        * DISPLAYED IN THE SECOND CHARACTER POSITION FROM THE
                        * UPPER RIGHT HAND CORNER OF THE SCREEN.  ON A READ, THE
                        * CHARACTERS BEING READ IN ARE DISPLAYED IN THE UPPER
                        * RIGHT HAND CORNER OF THE SCREEN.  THE NAME OF THE BLOCK
                        * CURRENTLY BEING READ OR SEARCHED OVER IS DISPLAYED ON THE
                        * RIGHT HAND SIDE OF THE SECOND LINE.  THE ERROR CODE FOR
                        * A READ IS DISPLAYED ON THE RIGHT SIDE OF THE THIRD LINE.
                        * NO CHANGE IN CHARACTER MEANS THAT THE READ WAS O.K.
                        * A "C" MEANS THERE WAS A CHECKSUM ERROR ON THE BLOCK.
                        * A "T" MEANS THAT THE WRONG NUMBER OF BYTES WERE READ
                        *  INTO THE BUFFER (TRAILER ERROR).  THIS WOULD OF COURSE
                        *  IMPLY A CHECKSUM ERROR ALONG WITH THE TRAILER ERROR.
                        *
                        *     THE CSTATS (CASSETTE STATUS) BYTE WILL CONTAIN A 0
                        * IF THE BLOCK READ IN WAS O.K.  IF IT WAS A BAD READ, IT
                        * WILL CONTAIN A 54 FOR TRAILER ERRROR OR A 43 FOR A
                        * CHECKSUM ERROR CODE UPON EXIT FROM THE ROUTINE.
                        *
                        *     SETTING THE NOPRNT (NO PRINTING) FLAG TO A 0 WILL
                        * STOP THE DISPLAY OF CHARACTERS ON THE SCREEN DURING
                        * CASSETTE READ AND WRITE, EXCEPT FOR THE T & C ERROR
                        * CODES, WHICH ARE ALWAYS DISPLAYED WHEN THEY OCCUR.
```

```
*     IF THE FIRST BYTE OF BLKNAM (BLOCK NAME) IS A 0
* WHEN THE READ BLOCK ROUTINE IS ENTERED, THE NEXT BLOCK
* WILL BE READ FROM TAPE NO MATTER WHAT THE NAME OF THE
* TAPE BLOCK IS.  ON THE SYS2N EXEC, A CONTROL SPACE
* CHARACTER CAN BE TYPED IN AS THE FIRST CHARACTER OF THE
* NAME IN THE LOAD BLOCK COMMAND.  THUS, A (CNTL L)(CNTL
* SPACE)(X) WOULD READ IN THE NEXT BLOCK ON THE TAPE.
*
*
*
*     THE TAPE FORMAT IS:
*       SYN
*       SYN
*       SYN
*       ESC
*       HI BYTE OF 16 BIT BLOCK LENGTH
*       LOW BYTE OF BLOCK LENGTH
*       FIRST CHAR OF BLOCK NAME
*       SECOND CHARACTER OF NAME
*       DATA
*       .
*       .
*       .
*       DATA
*       ETB
*       CHECKSUM
*       CHECKSUM
*       CHECKSUM
*       CHECKSUM
*
*
*
*     THE CHECKSUM IS CALCULATED FROM THE DATA, WHICH
* IS READ IN FROM THE CASSETTE BUFFER RESIDING IN MEMORY.
*
*
*     THE FORMAT FOR THE CASSETTE BUFFER IS:
*
```




```
*
*     WHEN DATA IS READ IN, THE READ ROUTINE SETS 'BFRSZE'
* TO POINT TO THE LAST CHARACTER READ INTO THE BUFFER.
* NOTE THAT THERE IS NO OVERFLOW CHECK WHEN DATA IS READ
* INTO MEMORY.  ON OUTPUT TO THE CASSETTE, THE BLOCK
* LENGTH IS CALCULATED FROM THE BFRPTR AND BFRSZE POINTERS.
*
*
*     THE READ BLOCK AND WRITE BLOCK ROUTINES AUTOMATICALLY
* TURN ON AND OFF THE CASSETTE.
```

```
*
*       THE ADDRESSES OF THE ROUTINES FOR CONTROLLING
*       THE CASSETTE ARE AS FOLLOWS:
*
*       INTLZ  - FB00 : INITIALIZES THE ACIA FOR USE.
*       WRTBLK - FB2D : WRITES A  FORMATTED BLOCK TO CASSETTE
*       RDBLK  - FB91 : READS IN A FORMATTED BLOCK OF TAPE
*       WRTMOD - FB2F : WRITES A BLOCK TO A RUNNING CASSETTE
*       RDMOD  - FB93 : READS A BLOCK FROM A RUNNING TAPE.
*       CASOUT - FB62 : WRITES OUT ONTO TAPE THE BYTE IN A.
*       CASIN  - FB7E : READS IN A BYTE INTO ACCUM. A
*       TURNON - FB77 : TURNS ON THE CASSETTE DRIVE.
*       TRNOFF - FBB0 : TURNS OFF THE CASSETTE DRIVE WHEN THE
*                       POINTR TO ACIA IS PASSED IN THE X REG.
*
*      NOTE THAT ALL THE ABOVE ROUTINES ARE SUBROUTINES THAT
*       ARE ENTERED BY A JSR OR BSR CALL.
*
*
*
*
*                USING THE CASSETTE DRIVERS.
*
*       IF THE EPROMS ON THE CPU BOARD ARE THE SYS2N OR THE
* SYS2D PROMS, THE DRIVERS ARE USED BY THE I (INITIALIZE),
* L (LOAD FROM CASSETTE) AND S (STORE ONTO CASSETTE)
* COMMANDS THAT ARE A PART OF THE SYS-2 EXECUTIVE.
*
*       IF THE CPU PROMS BEING USED ARE THE PDS-V3A OR THE
* PDS-V3N OR V3D PROM SETS, THEN THE USER MUST SET UP THE
* POINTERS AND CALL THE ROUTINES HIMSELF, USING THE
* DEBUGGER ON THE CPU PROMS.  (NOTE THAT THE V3A AND V3N
* DEBUGGERS VARY, FOR INSTANCE THE V3A GO COMMAND IS A 'G'
* WHILE THE V3N GO COMMAND IS A 'CNTL G'.  REFER TO THE
* USERS MANUAL FOR DETAILES.)
*       THE FIRST THING TO DO IS OPEN THE LOW MEMORY LOCATIONS
* USED BY THE DRIVERS AS FLAGS AND POINTERS AND INITIALIZE
* THE LOCATIONS.  THE ACIANO, BLKNAM, BFRPTR AND BFRSZE
* SHOULD NOW BE GIVEN VALUES.
*       ACIANO WOULD POINT TO F050 FOR THE FIRST CASSETTE AND
* TO F060 FOR THE SECOND CASSETTE DRIVE.
*       A SIMPLE PROGRAM TO CALL THE ROUTINES SHOULD NOW BE
* WRITTEN.  OPEN LOCATION 900.  TYPE IN THROUGH THE DEBUGGER
* THE INSTRUCTIONS JSR BD FB XX, 7E JMP FE4F.  XX IS THE SECOND BYTE xx
* OF THE DESIRED DRIVER ROUTINE, ADDRESS I.E. 00 FOR INITIALLIZATION
* OR 91 FOR READING A BLOCK OR 2D FOR WRITING A BLOCK.  THUS,
* TO INITIALIZE AN ACIA, ACIANO WOULD BE LOADED WITH
* THE ADDRESS OF THE ACIA AND THE USER WOULD THEN JUMP TO THE
* ROUTINE AT 900 BY OPENING 900 AND JUMPING TO IT WITH
* THE 'G' COMMAND.  THE ROUTINE WOULD BE AS FOLLOWS:
*
*CALL 900  BD      JSR
*      901  FB      HI BYTE OF THE DRIVER ROUTINE ADDRESS
*      902  00      LOW BYTE OF THE DRIVER ROUTINE ADDRESS.
*      903  7E      JMP
*      904  FE      ADDRESS OF THE DEBUGGER ON
*      905  4F      THE PDS V3A PROM SET
0
```

```
*      THE ACIA WOULD NOW BE INITIALIZED AND CONTROL WOULD
* HAVE RETURNED TO THE DEBUGGER.  TO READ IN A BLOCK,
* THE NAME WOULD BE PUT IN BLKNAM AND LOCATION 902 WOULD
* BE CHANGED TO 91.  THE USER WOULD THEN  JUMP TO 900.
*
*      TO TEST THE CASSETTE, SET BFRPTR TO E060 AND BFRSZE
* TO E0DF.  THIS WILL ALLOW THE USER TO WRITE OUT DATA
* FROM THE FOURTH, FIFTH, SIXTH AND SEVENTH LINE OF THE
* CRT DISPLAY AND THEN READ IT BACK ONTO THE DISPLAY.  DATA
* CAN BE TYPED ONTO THE SCREEN BY OPENING A LOCATION WITH
* THE DEBUGGER AND THEN MOVING THE CURSOR AROUND THE SCREEN
* TO CHANGE THE CHARACTERS.  THIS IS POSSIBLE BECAUSE THE
* ROUTINE TO INPUT AN ADDRESS CALLS THE EDITOR FOR INPUT.
*
*      THE DRIVER ROUTINES CAN ALSO BE USED TO PERFORM
* I-O WITH A MODEM OR TELETYPE.  THE MAIN HARDWARE
* DIFFERENCE BETWEEN THE CASSETTE AND MODEM/TTY IS THAT THE
* CASSETTE HAS A DIVIDE BY 16 CLOCK AND IS UNIDIRECTIONAL
* WHILE THE MODEM/TTY HAVE A DIVIDE BY 64 CLOCK AND CAN BE
* BIDIRECTIONAL.  BECAUSE OF THE CLOCK CHANGE THE ACIA MUST
* BE TURNED ON WITH A DIFFERENT VALUE BEFORE THE READ BLOCK
* OR WRITE BLOCK ROUTINES ARE ENTERED.  TO TURN ON THE ACIA
* STORE THE VALUE 'B2' INTO LOCATION F050 OR F060.  ONCE
* IT IS TURNED ON, EITHER CASIN OR CASOUT MAY BE CALLED
* REPEATEDLY OR WRITE MODEM BLOCK (WRTMOD) OR READ MODEM
* (RDMOD) CAN BE CALLED ONCE.  WRTMOD & RDMOD ARE THE SAME
* AS WRTBLK & RDBLK EXCEPT THEY DO NOT TURN ON THE ACIA.  A
* PROGRAM TO READ IN A BLOCK OF DATA FROM A MODEM OR TTY
* WOULD THUS BE (PLACED BEFORE THE 'CALL' ROUTINE):
*      8FA  DE 38  LDX    ACIANO   LOADS ACIA POINTTER.
*      8FC  86 B2  LDA A  #$B2     LOADS STARTUP VALUE.
*      8FE  A7 00  STA A  0,X      PUTS START CODE INTO ACIA.
* LOCATION 902 WOULD NOW BE 2F FOR WRITING AND 93 TO READ.
*      TO READ IN A CHARACTER FROM THE TELETYPE TURN ON THE
* ACIA AND GO TO THE FOLLOWING ROUTINE:
*      TTYIN  JSR    CASIN   READS IN A CHAR FROM KEYBOARD.
*             JMP    CASOUT  TYPES OUT CHAR ON PRINTER.
* THE RS232 SHOULD HAVE BEEN STRAPPED TO HALF DUPLEX.
*
*
*      IT IS TO BE STRESSED THAT  THE RELIABILITY OF THE
* CASSETTE CONTROLLER DEPENDS ON THE ADJUSTMENT OF THE
* TRIMMER ON THE SIM BOARD.  IF THE TRIMMER IS OUT OF
* ADJUSTMENT THE DATA WILL NOT READ IN PROPERLY.  BESIDES
* USING THE OSCILLOSCOPE TO ADJUST THE TRIMMER, IT CAN ALSO
* BE ADJUSTED BY READING IN A STRING OF SINGLE CHARACTERS
* FROM  THE CASSETTE AND ADJUSTING IT UNTIL THE CHARACTERS
* SYNC IN PROPERLY.  THE BEST CHARACTER TO USE IS A STRING
* OF 'U'S.  TO READ IN THE STRING FOR TESTING USE, WRITE A
* LOOP TO GET A CHARACTER FROM THE CASSETTE AND THEN
* DISPLAY THAT CHARACTER.  A SAMPLE ROUTINE TO DO THIS IS:
*    A   JSR   $FB7E    LOADS A WITH CASSETTE CHAR.
*        JSR   $FCAD    PDS-V3A PUTCHR ROUTINE.
*        TST   $F001    TESTS KEYBOARD FOR A KEY.
*        BPL   A        SKIPS BACK IF NO INPUT.
* THE ABOVE ROUTINE WOULD INPUT CHARACCTERS UNTIL A KEY
* ON THE KEYBOARD WAS DEPRESSED.
0
```

B2 will produce a signel of eight data bits and 2 stop bits. If the TTY operates on a different code, look up the proper initialization value in the ACIA section of the chip description appendex.

```
                   *       INTLZ INITIALIZES THE ACIA CONTROLLER FOR
                   *       A SPECIFIC TAPE UNIT AT MOUNT TIME.
                   *
FB00 DE 38         INTLZ   LDX     ACIANO    X GETS ACIA ADDRESS.
FB02 86 13                 LDA A   #$13      RESETS THE ACIA.
FB04 A7 00                 STA A   0,X       .
FB06 86 51                 LDA A   #OFF      SETS ACIA TO
FB08 A7 00                 STA A   0,X       /16 2 STOPS BIT FORMAT.
FB0A 39                    RTS

                   *       WRTHDR FORMATS THE HEADER ON THE TAPE.
                   *
FB0B 86 16         WRTHDR  LDA A   #SYN      PUTS SYNC CHARS ONTO TAPE.
FB0D 8D 53                 BSR     CASOUT    .
FB0F 8D 51                 BSR     CASOUT    .
FB11 8D 4F                 BSR     CASOUT    .
FB13 86 1B                 LDA A   #ESC      .
FB15 8D 4B                 BSR     CASOUT    .
FB17 96 3E                 LDA A   BFRSZE    FOLLOWING OUTPUTS LENGTH.
FB19 D6 3F                 LDA B   BFRSZE+1  LOADS LO BYTE OF END PTR.
FB1B D0 3D                 SUB B   BFRPTR+1  SUBS LO BYTE OF BEGIN PTR.
FB1D 92 3C                 SBC A   BFRPTR    SUBS HI BYTE OF START PTR.
FB1F 8D 41                 BSR     CASOUT    OUTPUTS HI LENGTH BYTE.
FB21 17                    TBA               LOADS LO BYTE OF LENGTH.
FB22 8D 3E                 BSR     CASOUT    OUTPUTS LOW LENGTH BYTE.
FB24 96 33                 LDA A   BLKNAM    PUTS OUT NAME OF BLOCK.
FB26 8D 3A                 BSR     CASOUT    .
FB28 96 34                 LDA A   BLKNAM+1  .
FB2A 8D 36                 BSR     CASOUT    PUTS OUT LAST OF NAME.
FB2C 39                    RTS               RETURNS BACK TO WRTBLK.

                   *       WRTBLK WRITES OUT A BLOCK OF DATA TO THE CASSETTE.
                   *
FB2D 8D 48         WRTBLK  BSR     TURNON    TURNS ON THE CASSETTE.
                   *        FOLLOWING WAITES FOR CASSETTE TO GET UP TO SPEED.
FB2F C6 09         WRTMOD  LDA B   #TIMCNT   LOADS TIME LOOP COUNTER.
FB31 CE 5161       TIME1   LDX     #TIME     MASTER TIME LOOP (1/4 SEC).
FB34 09            TIME2   DEX               COUNTS CYCLES OF LOOP.
FB35 26 FD                 BNE     TIME2     TESTS FOR FIRST TIME OUT.
FB37 5A                    DEC B   -         COUNTS TIMES IN LOOP.
FB38 26 F7                 BNE     TIME1     SKIPS BACK UNTIL DONE.
                   *        THE TIME LOOP IS NOW FINISHED.
FB3A 8D CF                 BSR     WRTHDR    WRITES HEADER ON THE TAPE.
FB3C 8D 10                 BSR     WRTBFR    WRITES OUT BUFFER DATA.
                   *        FOLLOWING WRITES THE TRAILER OUT ONTO THE TAPE.
FB3E 86 17         WRTTLR  LDA A   #ETB      OUTPUTS END-OF-BLOCK CHAR.
FB40 8D 20                 BSR     CASOUT    ETB IS DISPLAYED AS A "W".
FB42 17                    TBA               A GETS CHECKSUM FROM B.
FB43 8D 1D                 BSR     CASOUT    OUTPUTS THE CHECKSUM.
FB45 8D 1B                 BSR     CASOUT    OUTPUTS TRAILER FILLER
FB47 8D 19                 BSR     CASOUT    BYTES.
FB49 8D 17                 BSR     CASOUT    .
                   *        END OF TRAILER WRITING ROUTINE.
FB4B 8D 63                 BSR     TRNOFF    HALTS CASSETTE DRIVE.
FB4D 39                    RTS               -
```

```
                    *       WRTBFR WRITES OUT THE CONTENTS  OF THE
                    *       BUFFER ONTO THE CASSETTE TAPE.
                    *
                    *
FB4E 5F             WRTBFR CLR B   -          INIT CHECKSUM COUNT.
FB4F DE 3C                 LDX     BFRPTR     ACIA CONTROL MASK.
FB51 A6 00          WBFR1  LDA A   0,X        LOADS CHAR. FROM BUFFER.
FB53 DF 00                 STX     TMP        SAVES BUFFER PTR.
FB55 8D 0B                 BSR     CASOUT     PUTS CHAR ONTO CASSETTE.
FB57 1B                    ABA                A GETS A+B.
FB58 16                    TAB                B GETS A+B.
FB59 DE 00                 LDX     TMP        RESTORES PTR. INTO BUFFR.
FB5B 9C 3E                 CPX     BFRSZE     TESTS IF BUFFER EMPTY.
FB5D 27 17                 BEQ     CSOEXT     EXITS WHEN EMPTY.
FB5F 08                    INX                INC POINTER.
FB60 26 EF                 BNE     WBFR1      SKIPS BACK IF CHARS. LEFT.



                    *       CASOUT TAKES THE CHAR IN A AND PUTS IT OUT
                    *       ONTO THE CASSETTE TAPE.
                    *
                    *
FB62 36             CASOUT PSH A   .          SAVES CHAR TO READ OUT.
FB63 DE 38                 LDX     ACIANO     X GETS PHYSICAL ACIA ADDR.
FB65 86 02                 LDA A   #2         LOADS CONTROL TEST BITS.
FB67 A5 00          CAS01  BIT A   0,X        TESTS IF ACIA BUFFER EMPTY.
FB69 27 FC                 BEQ     CAS01      LOOPS BACK UNTIL READY.
FB6B 32                    PUL A   .          GETS ORIG CHAR.
FB6C A7 01                 STA A   1,X        STORS CHAR INTO ACIA BUFFER.
FB6E 7D 003A               TST     NOPRNT     TESTS IF PRINTOUT ALLOWED.
FB71 27 03                 BEQ     CSOEXT     SKIPS PRINTING IF A 0.
FB73 B7 E01E               STA A   $E01E      DISPLAYS CHAR ON SCREEN.
FB76 39             CSOEXT RTS



                    *        TURNON TURNS ON THE CASSETTE DRIVE.
                    *
                    *
FB77 DE 38          TURNON LDX     ACIANO     LOADS CASSETTE ACIA ADDRESS.
FB79 86 B1                 LDA A   #ON        .
FB7B A7 00                 STA A   0,X        TURNS ACIA ON.
FB7D 39                    RTS
```

```
                    *       CASIN READS IN A CHARACTER FROM THE CASSETTE TAPE
                    *       INTO THE A ACCUMULATOR.
                    *
                    *
FB7E DE 38          CASIN   LDX     ACIANO      X GETS THE ACIA ADDRESS.
FB80 86 01                  LDA A   #1          LOADS TEST BITS.
FB82 A5 00          CASIN1  BIT A   0,X         TESTS IF ACCIA BUFFR FULL.
FB84 27 FC                  BEQ     CASIN1      SKIPS BACK IF NOT IN YET.
FB86 A6 01                  LDA A   1,X         LOADS IN CHAR FROM TAPE.
FB88 7D 003A                TST     NOPRNT      TESTS IF PRINT IS OFF.
FB8B 27 03                  BEQ     CINEXT      SKIPS DISPLAYING IF 0.
FB8D B7 E01F                STA A   $E01F       DISPLAYS CHR ON TV.
FB90 39             CINEXT  RTS



                    *       RDBLK READS IN A BLOCK FROM THE CASSETTE
                    *       TAPE INTO BUFFER MEMORY.
                    *
                    *
FB91 8D E4          RDBLK   BSR     TURNON      TURNS ON TAPE DRIVE.
FB93 8D 20          RDMOD   BSR     RDHDR       READS IN THE HEADER.
FB95 8D 55                  BSR     RDBFR       READS IN DATA INTO BUFFER.
                    *        FOLLOWING READS IN TRAILER AND CHECKS CHECKSUM.
FB97 4F             RDTLR   CLR A   -           LOADS A 0 FOR A GOOD READ.
FB98 97 09                  STA A   CSTATS      SETS STATUS BYTE TO NO ERR.
FB9A 8D E2                  BSR     CASIN       INPUTS END-OF-BLOCK CHAR.
FB9C 81 17                  CMP A   #ETB        ETB DISPLAYS AS A "W".
FB9E 27 04                  BEQ     RTLR1       SKIPS IF NO ETB ERROR.
FBA0 86 54                  LDA A   #ERR4       LOADS TRAILER ERROR CODE.
FBA2 20 07                  BRA     RTLR2       SKIPS TO STORE ERROR CODE.
FBA4 8D D8          RTLR1   BSR     CASIN       READS IN CHECKSUM.
FBA6 11                     CBA                 TESTS CHECKSUM.
FBA7 27 07                  BEQ     TRNOFF      SKIPS IF OK.
FBA9 86 43                  LDA A   #ERR5       SETS CHECKSUM ERROR CODE.
FBAB 97 09          RTLR2   STA A   CSTATS      SETS ERROR STATUS BYTE.
FBAD B7 E05F                STA A   $E05F       DISP.  ERR CODE ON SCREEN.
                    *        END OF TRAILER READ IN.
                    *        FOLLOWING TURNS OFF THE CASSETTE DRIVE.
FBB0 86 51          TRNOFF  LDA A   #OFF        LOADS COMMAND TO TURN
FBB2 A7 00                  STA A   0,X         OFF ACIA CASSETTE DRIVE.
FBB4 39                     RTS
```

```
                     *      RDHDR FINDS THE START OF THE BLOCK ON THE TAPE.
                     *      FIGURES THE BUFFER END AND CHECKS THE NAME.
                     *
                     *
FBB5 8D C7   RDHDR   BSR    CASIN     FOLLOWING FINDS THE
FBB7 81 16   RDHDR1  CMP A  #SYN      START OF THE HEADER.
FBB9 26 FA           BNE    RDHDR     .
FBBB 8D C1           BSR    CASIN     TESTS FOR HEADR CHAR.
FBBD 81 1B           CMP A  #ESC      START-OF-HEADER.
FBBF 26 F6           BNE    RDHDR1    GOES BACK IF NOT GOOD HDR.
FBC1 8D BB           BSR    CASIN     READS IN HI BYTE OF LEN.
FBC3 16              TAB              SAVES HI LENGTH BYTE.
FBC4 8D B8           BSR    CASIN     INPUTS LO SIZE (LEN) BYTE.
FBC6 9B 3D           ADD A  BFRPTR+1  FORMS POINTER TO THE
FBC8 D9 3C           ADC B  BFRPTR    TOP BYTE OF THE BUFFER.
FBCA 97 3F           STA A  BFRSZE+1  SAVES THE HI BUFF. PTR.
FBCC D7 3E           STA B  BFRSZE    TO THE CASSETTE BUFFER.
FBCE 8D AE           BSR    CASIN     READS IN BLOCK NAME.
FBD0 36              PSH A  -         SAVES FIRST CHAR OF NAME.
FBD1 8D AB           BSR    CASIN     READ IN SECOND CHAR INTO A.
FBD3 D6 3A           LDA B  NOPRNT    TESTS IF PRINT IS OFF.
FBD5 33              PUL B  -         RESTORES FIRST NAME CHAR.
FBD6 27 06           BEQ    RHDR1     SKIPS IF PRINT FLAG IS 0.
FBD8 F7 E03E         STA B  $E03E     DISPLAYS BLOCK NAME ON
FBDB B7 E03F         STA A  $E03F     THE CRT SCREEN.
FBDE 7D 0033 RHDR1   TST    BLKNAM    TESTS IF NAME IS CHECKED.
FBE1 27 08           BEQ    RHDR2     SKIPS IF NO NAME CHECK.
FBE3 D1 33           CMP B  BLKNAM    TESTS FIRST CHAR OF NAME.
FBE5 26 CE           BNE    RDHDR     SKIPS BACK IF BAD NAME
FBE7 91 34           CMP A  BLKNAM+1  TESTS SECOND NAME CHAR.
FBE9 26 CA           BNE    RDHDR     SKIPS BACK IF BAD NAME.
FBEB 39      RHDR2   RTS              .


                     *      RDBFR READS DATA INTO THE MEMORY BUFFER FROM
                     *      THE CASSETTE.
                     *
                     *

FBEC 5F      RDBFR   CLR B  -         INIT B FOR CHECKSUM.
FBED DE 3C           LDX    BFRPTR    LOADS START OF BUFFER.
FBEF DF 00   RBFR1   STX    TMP       .
FBF1 8D 8B           BSR    CASIN     A GETS CHAR READ IN.
FBF3 DE 00           LDX    TMP       X GETS BUFFER PTR.
FBF5 A7 00           STA A  0,X       STORS CHAR INTO BUFFER.
FBF7 1B              ABA              A GETS A+B.
FBF8 16              TAB              B GETS A.
FBF9 9C 3E           CPX    BFRSZE    TESTS IF BUFFER FULL
FBFB 27 EE           BEQ    RHDR2     SKIPS TO EXIT IF ALL IS IN.
FBFD 08              INX              INC TO NEXT CHAR POSITIN.
FBFE 26 EF           BNE    RBFR1     GOES BACK IF ANY LEFT.
                     END
```

```
        NAM     PDS-SYS2N
        OPT     O,NOG
*
*       PDS SYSTEM 2N  CASSETTE DRIVERS  (SYS2NF)
*
*       PROGRAMMED BY ERIC JAMESON
*
*
*
*       COPYRIGHT 1976 SPHERE CORPORATION
*       940 N.  400 EA.,  NORTH SALT LAKE, UTAH  84054 .
*
*       P. O. BOX 213,  BOUNTIFUL, UTAH  84010
*
*
*
*
*       SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
*       DISTRIBUTION AND USE OF THE PDS SOFTWARE.
*       NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
*       WRITTEN PERMISSION OF SPHERE CORP.
*
*
*
*
*
* THE PROGRAM DEVELOPMENT SYSTEM (PDS SYS2N) IS A SET OF
* PROGRAMS RESIDING ON ERASABLE PROGRAMMABLE READ ONLY
* MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS
* SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE
* DEVELOPMENT OF COMPUTER PROGRAMS.
*     TOWARD THIS END, THE 5 PDS EPROMS CONTAIN A CURSOR
* BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING
* AID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16
* BIT MULTIPLY AND DIVIDE, ASCII-TO-BINARY, AND
* BINARY-TO-ASCII ROUTINES, AND ROUTINES TO DO
* INPUT AND OUTPUT TO THE AUDIO CASSETTE.
*
*
*
*     THE SYS2N SOFTWARE IS AN UPGRADE OF THE PDS V3A
* (V3N & V3D) SOFTWARE DESIGNED TO RUN WITH THE CASSETTE
* SYSTEM.  THERE ARE TWO VERSIONS OF THE SYS2 SOFTWARE:
* SYS2N WHICH RUNS WITH THE NEW KEYBOARD AND THE SYS2A
* WHICH RUNS WITH THE ORIGIONAL (KBD/1A) KEYBOARD.  THE
* MAIN DIFFERENCE BETWEEN THE V3A AND SYS2 VERSIONS ARE
* THAT A FIFTH EPROM HAS BEEN ADDED AND THAT THE MINI-
* ASSEMBLER HAS BEEN DELEATED AND REPLACED WITH A SET OF
* COMMANDS TO DO LOADING AND DUMPING OF CASSETTE TAPES.
```

The SYS2NF prom is a version of the SYS2N cassette prom with a software bug fixed. This bug would cause the next block of a multiblock read to be skipped if the checksum on the preceeding block was a 16 and the tape had not been previously used. The change is on page 10 where the ESC test and branch now branches to RDHDR1 instead of RDHDR.

```
                *             MEMORY MAP
                *
        0000    TMP     EQU   $00
        0002    TMP1    EQU   $02
        0004    ARB     EQU   $04      16 BIT ACC. PSEUDO REG B.
        0004    AR3     EQU   $04      _HI BYTE OF ARB.
        0005    AR2     EQU   $05      _LO BYTE OF ARB.
        0006    ARA     EQU   $06      16 BIT ARITH PSEUDO REG A.
        0006    AR1     EQU   $06      _HI BYTE OF ARA.
        0007    AR0     EQU   $07      _LO BYTE OF ARA.
        0008    DIGIT   EQU   $08      BYTE USED BY ASCBIN FOR TMP.
        0009    CSTATS  EQU   $09      CASSETTE I-O STATUS BYTE.
        000A    OUTEND  EQU   $0A      END OF OUTPUT BUFFER TEXT.
        000C    BUFADR  EQU   $0C      START OF I/O BUFFER (PTR) ← POINTER
        000E    BUFEND  EQU   $0E      PTR. TO END OF I/O BUFFR.
        0011    OUTBUF  EQU   $11      START OF OUTPUT BUFFER.
        0013    CASNUM  EQU   $13      PHYSICAL CASSETTE NUMBR.
        0014    SRCADR  EQU   $14      SOURCE FOR TEXT MOVES.
        0016    DSTADR  EQU   $16      DEST. ADDR. FOR TEXT MOVE.
        001A    ENDMEM  EQU   $1A      LAST ADDRES OF REAL MEMORY.
        001C    CSRPTR  EQU   $1C      PTR TO CURSER ON SCREEN.
        001E    BUFPTR  EQU   $1E      TEMP PTR USED BY OUTSTR.
        0020    BUFFLO  EQU   $20      PTR TO END OF LOW EDIT TXT.
        0022    BUFFHI  EQU   $22      PTR TO START OF HI TEXT.
        0024    SCNPTR  EQU   $24      PTR. TO BUFFRD TXT START.
        0026    SRCASM  EQU   $26      PTR TO ASSMBLR SOURCE CODE.
        002A    ONDVAL  EQU   $2A      HAS ASSMBLR OPERND VALUE.
        002C    SYMVAL  EQU   $2C      VALUE PUT IN ASSM. SYMTBL.
        002E    BRKSAV  EQU   $2E      TEMP SAVE FOR BRKPT DATA.
        0030    BRKADR  EQU   $30      ADDDRESS OF BREAKPOINT.
        0032    EDIT    EQU   $32      0 IF EDITOR IS NOT RUNNING.
        0033    BLKNAM  EQU   $33      CASSETTE BLOCK NAME.
        0035    IOBUFF  EQU   $35      I-O BUFFER FOR DEBUGGER.
        0038    ACIANO  EQU   $38      SYS2N CASSETTE ACIA ADDR.
        003A    NOPRNT  EQU   $3A      CASSETTE NAME PRINT FLAG.
        003B    BLKTYP  EQU   $3B      CASSETTE BLOCK TYPE CODE.
        003C    BFRPTR  EQU   $3C      ADDR. OF I-O BUFF. FOR CASS.
        003E    BFRSZE  EQU   $3E      LENGTH OF CASS. BUFFER.
        0040    PCVAL   EQU   $40      PROGRAM COUNTER FOR ASSM.


                *       FOLLOWING ARE VARIABLE VALUES.
                *
        26F0    TIMEA   EQU   $26F0    TIMER COUNTER.
        00B1    ON      EQU   $B1      ACIA VALUE TO TURN ON CASS.
        0051    OFF     EQU   $51      ACIA VALUE TO STOP CASS.
        0003    ETX     EQU   $03      END-OF-TEXT.
        0016    SYN     EQU   $16      SYNCHRONISE.
        0017    ETB     EQU   $17      END-OF-TRANSMISSION-BLOCK.
        001B    ESC     EQU   $1B      EXCAPE TO NONSTANDARD HDR.
        0054    ERR4    EQU   $'T      SET FOR TRAILER ERROR
        0043    ERR5    EQU   $'C      SET FOR CHECKSUM ERR.
        5161    TIME    EQU   20833    TIME CNTR FOR 1/4 SEC.
        0009    TIMCNT  EQU   9        TIMES FOR 2 & 1/4 SECONDS.
0
```

PDS SYSTEM 2N CASSETTE DRIVERS

```
                        *
      FB00                      ORG     #FB00
                        *
                        *
                        *            CASSETTE I-O DRIVERS
                        *
                        *      THE CASSETTE DRIVERS LOAD AND DUMP A BLOCK OF DATA
                        * TO AND FROM THE CASSETTE.  THEY HANDLE BOTH THE HEADER
                        * AND TRAILER FORMATTING.
                        *      THE DRIVERS ARE SET UP AS A SET OF SUBROUTINES
                        * CALLED BY THE EXECUTIVE OR THE USER'S PROGRAMS.
                        *
                        *
                        *
                        *      LOW MEMORY ADDRESSES USED BY THE DRIVERS ARE:
                        *
                        *      CSTATS  AT  09  . USED TO STORE INPUT ERR CODE.
                        *      BLKNAM  AT  33    2 CHAR. NAME OF BLOCK.
                        *      ACIANO  AT  38    PTR. TO THE ACIA CURRENTLY USED.
                        *      NOPRNT  AT  3A    PRINT FLAG FOR BLOCK NAME.
                        *      BFRPTR  AT  3C    START OF CASSETTE I-O BUFFER.
                        *      BFRSZE  AT  3E    END OF CASSETTE DATA BUFFER.
                        *
                        *
                        *      THE DRIVERS CAN BE RUN FROM THE PDS-SYS2N SOFTWARE
                        * SYSTEM, WHICH CONTAINS THE CASSETTE LOAD AND DUMP
                        * COMMANDS IN THE EXEC, OR FROM THE PDS-V3A, WHERE THE
                        * DRIVERS ARE CALLED THROUGH THE DEBUGGER.
                        *
                        *      THE DRIVERS DISPLAY PERTINENT DATA ON THE SCREEN
                        * WHEN WRITING OR READING FROM THE CASSETTES.  ON A WRITE,
                        * THE CHARACTERS BEING WRITTEN ONTO THE CASSETTE ARE
                        * DISPLAYED IN THE SECOND CHARACTER POSITION FROM THE
                        * UPPER RIGHT HAND CORNER OF THE SCREEN.  ON A READ, THE
                        * CHARACTERS BEING READ IN ARE DISPLAYED IN THE UPPER
                        * RIGHT HAND CORNER OF THE SCREEN.  THE NAME OF THE BLOCK
                        * CURRENTLY BEING READ OR SEARCHED OVER IS DISPLAYED ON THE
                        * RIGHT HAND SIDE OF THE SECOND LINE.  THE ERROR CODE FOR
                        * A READ IS DISPLAYED ON THE RIGHT SIDE OF THE THIRD LINE.
                        * NO CHANGE IN CHARACTER MEANS THAT THE READ WAS O.K.
                        * A "C" MEANS THERE WAS A CHECKSUM ERROR ON THE BLOCK.
                        * A "T" MEANS THAT THE WRONG NUMBER OF BYTES WERE READ
                        *  INTO THE BUFFER (TRAILER ERROR).  THIS WOULD OF COURSE
                        *  IMPLY A CHECKSUM ERROR ALONG WITH THE TRAILER ERROR.
                        *
                        *      THE CSTATS (CASSETTE STATUS) BYTE WILL CONTAIN A Ø
                        * IF THE BLOCK READ IN WAS O.K.  IF IT WAS A BAD READ  IT
                        * WILL CONTAIN A 54 FOR TRAILER ERRROR OR A 43 FOR A
                        * CHECKSUM ERROR CODE UPON EXIT FROM THE ROUTINE.
                        *
                        *      SETTING THE NOPRNT (NO PRINTING) FLAG TO A 0 WILL
                        * STOP THE DISPLAY OF CHARACTERS ON THE SCREEN DURING
                        * CASSETTE READ AND WRITE, EXCEPT FOR THE T & C ERROR
                        * CODES, WHICH ARE ALWAYS DISPLAYED WHEN THEY OCCUR.
```

```
*      IF THE FIRST BYTE OF BLKNAM (BLOCK NAME) IS A 0
* WHEN THE READ BLOCK ROUTINE IS ENTERED, THE NEXT BLOCK
* WILL BE READ FROM TAPE NO MATTER WHAT THE NAME OF THE
* TAPE BLOCK IS.  ON THE SYS2N EXEC, A CONTROL SPACE
* CHARACTER CAN BE TYPED IN AS THE FIRST CHARACTER OF THE
* NAME IN THE LOAD BLOCK COMMAND.  THUS, A (CNTL L)(CNTL
* SPACE)(X) WOULD READ IN THE NEXT BLOCK ON THE TAPE.
*
*
*
*      THE TAPE FORMAT IS:
*        SYN
*        SYN
*        SYN
*        ESC
*        HI BYTE OF 16 BIT BLOCK LENGTH
*        LOW BYTE OF BLOCK LENGTH
*        FIRST CHAR OF BLOCK NAME
*        SECOND CHARACTER OF NAME
*        DATA
*        .
*        .
*        .
*        DATA
*        ETB
*        CHECKSUM
*        CHECKSUM
*        CHECKSUM
*        CHECKSUM
*
*
*
*      THE CHECKSUM IS CALCULATED FROM THE DATA, WHICH
* IS READ IN FROM THE CASSETTE BUFFER RESIDING IN MEMORY.
*
*
*      THE FORMAT FOR THE CASSETTE BUFFER IS:
*
*        BFRPTR                          BFRSZE
*
*        |
*        |C|A|S|S|E|T|T|E| | | |B|U|F|F|E|R|
*
*
*      WHEN DATA IS READ IN, THE READ ROUTINE SETS 'BFRSZE'
* TO POINT TO THE LAST CHARACTER READ INTO THE BUFFER.
* NOTE THAT THERE IS NO OVERFLOW CHECK WHEN DATA IS READ
* INTO MEMORY.  ON OUTPUT TO THE CASSETTE, THE BLOCK
* LENGTH IS CALCULATED FROM THE BFRPTR AND BFRSZE POINTERS.
*
*
*      THE READ BLOCK AND WRITE BLOCK ROUTINES AUTOMATICALLY
* TURN ON AND OFF THE CASSETTE.
```

```
*
*       THE ADDRESSES OF THE ROUTINES FOR CONTROLLING
*       THE CASSETTE ARE AS FOLLOWS:
*
*       INTLZ  - FB00 : INITIALIZES THE ACIA FOR USE.
*       WRTBLK - FB2D : WRITES A  FORMATTED BLOCK TO CASSETTE
*       RDBLK  - FB91 : READS IN A FORMATTED BLOCK OF TAPE
*       WRTMOD - FB2F : WRITES A BLOCK TO A RUNNING CASSETTE
*       RDMOD  - FB93 : READS A BLOCK FROM A RUNNING TAPE.
*       CASOUT - FB62 : WRITES OUT ONTO TAPE THE BYTE IN A.
*       CASIN  - FB7E : READS IN A BYTE INTO ACCUM. A
*       TURNON - FB77 : TURNS ON THE CASSETTE DRIVE.
*       TRNOFF - FBB0 : TURNS OFF THE CASSETTE DRIVE WHEN THE
*                       POINTR TO ACIA IS PASSED IN THE X REG.
*
*      NOTE THAT ALL THE ABOVE ROUTINES ARE SUBROUTINES THAT
*       ARE ENTERED BY A JSR OR BSR CALL.
*
*
*
*
*                    USING THE CASSETTE DRIVERS.
*
*       IF THE EPROMS ON THE CPU BOARD ARE THE SYS2N OR THE
* SYS2D PROMS, THE DRIVERS ARE USED BY THE I (INITIALIZE),
* L (LOAD FROM CASSETTE) AND S (STORE ONTO CASSETTE)
* COMMANDS THAT ARE A PART OF THE SYS-2 EXECUTIVE.
*
*       IF THE CPU PROMS BEING USED ARE THE PDS-V3A OR THE
* PDS-V3N OR V3D PROM SETS, THEN THE USER MUST SET UP THE
* POINTERS AND CALL THE ROUTINES HIMSELF, USING THE
* DEBUGGER ON THE CPU PROMS.  (NOTE THAT THE V3A AND V3N
* DEBUGGERS VARY, FOR INSTANCE THE V3A GO COMMAND IS A 'G'
* WHILE THE V3N GO COMMAND IS A 'CNTL G'.  REFER TO THE
* USERS MANUAL FOR DETAILES.)
*       THE FIRST THING TO DO IS OPEN THE LOW MEMORY LOCATIONS
* USED BY THE DRIVERS AS FLAGS AND POINTERS AND INITIALIZE
* THE LOCATIONS.  THE ACIANO, BLKNAM, BFRPTR AND BFRSZE
* SHOULD NOW BE GIVEN VALUES.
*       ACIANO WOULD POINT TO F050 FOR THE FIRST CASSETTE AND
* TO F060 FOR THE SECOND CASSETTE DRIVE.
*       A SIMPLE PROGRAM TO CALL THE ROUTINES SHOULD NOW BE
* WRITTEN.  OPEN LOCATION 900.  TYPE IN THROUGH THE DEBUGGER
* THE INSTRUCTIONS JSR BDXX, JMP FE4F.  XX IS THE SECOND BYTE
* OF THE DESIRED DRIVER ROUTINE, I.E. 00 FOR INITIALLIZATION
* OR 91 FOR READING A BLOCK OR 2D FOR WRITING A BLOCK.  THUS
* TO INITIALIZE AN ACIA, ACIANO WOULD BE LOADED WITH
* THE ADDRESS OF THE ACIA AND THE USER WOULD THEN JUMP TO THE
* ROUTINE AT 900 BY OPENING 900 AND JUMPING TO IT WITH
* THE 'G' COMMAND.  THE ROUTINE WOULD BE AS FOLLOWS:
*
*CALL 900   BD      JSR
*     901   FB      HI BYTE OF ROUTINE AADDRESS
*     902   00      LOW BYTE OF DRIVER ADDRESS.
*     903   7E      JMP
*     904   FE      ADDRESS OF THE DEBUGGER ON
*     905   4F      THE PDS V3A PROM SET
0
```

```
*      THE ACIA WOULD NOW BE INITIALIZED AND CONTROL WOULD
* HAVE RETURNED TO THE DEBUGGER.  TO READ IN A BLOCK,
* THE NAME WOULD BE PUT IN BLKNAM AND LOCATION 902 WOULD
* BE CHANGED TO 91.  THE USER WOULD THEN  JUMP TO 900.
*
*      TO TEST THE CASSETTE, SET BFRPTR TO E060 AND BFRSZE
* TO E0DF.  THIS WILL ALLOW THE USER TO WRITE OUT DATA
* FROM THE FOURTH, FIFTH, SIXTH AND SEVENTH LINE OF THE
* CRT DISPLAY AND THEN READ IT BACK ONTO THE DISPLAY.  DATA
* CAN BE TYPED ONTO THE SCREEN BY OPENING A LOCATION WITH
* THE DEBUGGER AND THEN MOVING THE CURSOR AROUND THE SCREEN
* TO CHANGE THE CHARACTERS.  THIS IS POSSIBLE BECAUSE THE
* ROUTINE TO INPUT AN ADDRESS CALLS THE EDITOR FOR INPUT.
*
*      THE DRIVER ROUTINES CAN ALSO BE USED TO PERFORM
* I-O WITH A MODEM OR TELETYPE.  THE MAIN HARDWARE
* DIFFERENCE BETWEEN THE CASSETTE AND MODEM/TTY IS THAT THE
* CASSETTE HAS A DIVIDE BY 16 CLOCK AND IS UNIDIRECTIONAL
* WHILE THE MODEM/TTY HAVE A DIVIDE BY 64 CLOCK AND CAN BE
* BIDIRECTIONAL.  BECAUSE OF THE CLOCK CHANGE THE ACIA MUST
* BE TURNED ON WITH A DIFFERENT VALUE BEFORE THE READ BLOCK
* OR WRITE BLOCK ROUTINES ARE ENTERED.  TO TURN ON THE ACIA
* STORE THE VALUE 'B2' INTO LOCATION F050 OR F060.  ONCE
* IT IS TURNED ON, EITHER CASIN OR CASOUT MAY BE CALLED
* REPEATEDLY OR WRITE MODEM BLOCK (WRTMOD) OR READ MODEM
* (RDMOD) CAN BE CALLED ONCE.  WRTMOD & RDMOD ARE THE SAME
* AS WRTBLK & RDBLK EXCEPT THEY DO NOT TURN ON THE ACIA.  A
* PROGRAM TO READ IN A BLOCK OF DATA FROM A MODEM OR TTY
* WOULD THUS BE (PLACED BEFORE THE 'CALL' ROUTINE):
*      8FA  DE 38  LDX     ACIAND    LOADS ACIA POINTTER.
*      8FC  86 B2  LDA A   #$B2      LOADS STARTUP VALUE.
*      8FE  A7 00  STA A   0,X       PUTS START CODE INTO ACIA.
* LOCATION 902 WOULD NOW BE 2F FOR WRITING AND 93 TO READ.
*      TO READ IN A CHARACTER FROM THE TELETYPE TURN ON THE
* ACIA AND GO TO THE FOLLOWING ROUTINE:
*      TTYIN  JSR    CASIN    READS IN A CHAR FROM KEYBOARD.
*             JMP    CASOUT   TYPES OUT CHAR ON PRINTER.
* THE RS232 SHOULD HAVE BEEN STRAPPED TO HALF DUPLEX.
*
*
*      IT IS TO BE STRESSED THAT  THE RELIABILITY OF THE
* CASSETTE CONTROLLER DEPENDS ON THE ADJUSTMENT OF THE
* TRIMMER ON THE SIM BOARD.  IF THE TRIMMER IS OUT OF
* ADJUSTMENT THE DATA WILL NOT READ IN PROPERLY.  BESIDES
* USING THE OSCILLOSCOPE TO ADJUST THE TRIMMER, IT CAN ALSO
* BE ADJUSTED BY READING IN A STRING OF SINGLE CHARACTERS
* FROM  THE CASSETTE AND ADJUSTING IT UNTIL THE CHARACTERS
* SYNC IN PROPERLY.  THE BEST CHARACTER TO USE IS A STRING
* OF 'U'S.  TO READ IN THE STRING FOR TESTING USE, WRITE A
* LOOP TO GET A CHARACTER FROM THE CASSETTE AND THEN
* DISPLAY THAT CHARACTER.  A SAMPLE ROUTINE TO DO THIS IS:
*      A  JSR  $FE7E    LOADS A WITH CASSETTE CHAR.
*         JSR  $FCAD    PDS-V3A PUTCHR ROUTINE.
*         TST  $F001    TESTS KEYBOARD FOR A KEY.
*         BPL  A        SKIPS BACK IF NO INPUT.
* THE ABOVE ROUTINE WOULD INPUT CHARACCTERS UNTIL A KEY
* ON THE KEYBOARD HAS DEPRESSED.
```

B2 will produce a signel of eight data bits and 2 stop bits. If the TTY operates on a different code, look up the proper initialization value in the ACIA section of the chip description appendex.

0

```
                  *       INTLZ INITIALIZES THE ACIA CONTROLLER FOR
                  *       A SPECIFIC TAPE UNIT AT MOUNT TIME.
                  *
FB00 DE 38        INTLZ   LDX     ACIANO    X GETS ACIA ADDRESS.
FB02 86 13                LDA A   #$13      RESETS THE ACIA.
FB04 A7 00                STA A   0,X       .
FB06 86 51                LDA A   #0FF      SETS ACIA TO
FB08 A7 00                STA A   0,X       /16 2 STOPS BIT FORMAT.
FB0A 39                   RTS

                  *       WRTHDR FORMATS THE HEADER ON THE TAPE.
                  *
FB0B 86 16        WRTHDR  LDA A   #SYN      PUTS SYNC CHARS ONTO TAPE.
FB0D 8D 53                BSR     CASOUT    .
FB0F 8D 51                BSR     CASOUT    .
FB11 8D 4F                BSR     CASOUT    .
FB13 86 1B                LDA A   #ESC      .
FB15 8D 4B                BSR     CASOUT    .
FB17 96 3E                LDA A   BFRSZE    FOLLOWING OUTPUTS LENGTH.
FB19 D6 3F                LDA B   BFRSZE+1  LOADS LO BYTE OF END PTR.
FB1B D0 3D                SUB B   BFRPTR+1  SUBS LO BYTE OF BEGIN PTR.
FB1D 92 3C                SBC A   BFRPTR    SUBS HI BYTE OF START PTR.
FB1F 8D 41                BSR     CASOUT    OUTPUTS HI LENGTH BYTE.
FB21 17                   TBA               LOADS LO BYTE OF LENGTH.
FB22 8D 3E                BSR     CASOUT    OUTPUTS LOW LENGTH BYTE.
FB24 96 33                LDA A   BLKNAM    PUTS OUT NAME OF BLOCK.
FB26 8D 3A                BSR     CASOUT    .
FB28 96 34                LDA A   BLKNAM+1  .
FB2A 8D 36                BSR     CASOUT    PUTS OUT LAST OF NAME.
FB2C 39                   RTS               RETURNS BACK TO WRTBLK.

                  *       WRTBLK WRITES OUT A BLOCK OF DATA TO THE CASSETTE.
                  *
FB2D 8D 48        WRTBLK  BSR     TURNON    TURNS ON THE CASSETTE.
                  *        FOLLOWING WAITES FOR CASSETTE TO GET UP TO SPEED.
FB2F C6 09        WRTMOD  LDA B   #TIMCNT   LOADS TIME LOOP COUNTER.
FB31 CE 5161      TIME1   LDX     #TIME     MASTER TIME LOOP (1/4 SEC).
FB34 09           TIME2   DEX               COUNTS CYCLES OF LOOP.
FB35 26 FD                BNE     TIME2     TESTS FOR FIRST TIME OUT.
FB37 5A                   DEC B   -         COUNTS TIMES IN LOOP.
FB38 26 F7                BNE     TIME1     SKIPS BACK UNTIL DONE.
                  *        THE TIME LOOP IS NOW FINISHED.
FB3A 8D CF                BSR     WRTHDR    WRITES HEADER ON THE TAPE.
FB3C 8D 10                BSR     WRTBFR    WRITES OUT BUFFER DATA.
                  *        FOLLOWING WRITES THE TRAILER OUT ONTO THE TAPE.
FB3E 86 17        WRTTLR  LDA A   #ETB      OUTPUTS END-OF-BLOCK CHAR.
FB40 8D 20                BSR     CASOUT    ETB IS DISPLAYED AS A "W".
FB42 17                   TBA               A GETS CHECKSUM FROM B.
FB43 8D 1D                BSR     CASOUT    OUTPUTS THE CHECKSUM.
FB45 8D 1B                BSR     CASOUT    OUTPUTS TRAILER FILLER
FB47 8D 19                BSR     CASOUT    BYTES.
FB49 8D 17                BSR     CASOUT    .
                  *        END OF TRAILER WRITING ROUTINE.
FB4B 8D 63                BSR     TRNOFF    HALTS CASSETTE DRIVE.
FB4D 39                   RTS               -
```

```
                  *      WRTBFR WRITES OUT THE CONTENTS  OF THE
                  *      BUFFER ONTO THE CASSETTE TAPE.
                  *
                  *
FB4E 5F           WRTBFR CLR B    -        INIT CHECKSUM COUNT.
FB4F DE 3C               LDX      BFRPTR   ACIA CONTROL MASK.
FB51 A6 00        WBFR1  LDA A    0,X      LOADS CHAR.  FROM BUFFER.
FB53 DF 00               STX      TMP      SAVES BUFFER PTR.
FB55 8D 0B               BSR      CASOUT   PUTS CHAR ONTO CASSETTE.
FB57 1B                  ABA               A GETS A+B.
FB58 16                  TAB               B GETS A+B.
FB59 DE 00               LDX      TMP      RESTORES PTR.  INTO BUFFR.
FB5B 9C 3E               CPX      BFRSZE   TESTS IF BUFFER EMPTY.
FB5D 27 17               BEQ      CSOEXT   EXITS WHEN EMPTY.
FB5F 08                  INX               INC POINTER.
FB60 26 EF               BNE      WBFR1    SKIPS BACK IF CHARS.  LEFT.


                  *      CASOUT TAKES THE CHAR IN A AND PUTS IT OUT
                  *      ONTO THE CASSETTE TAPE.
                  *
                  *
FB62 36           CASOUT PSH A    .        SAVES CHAR TO READ OUT.
FB63 DE 38               LDX      ACIANO   X GETS PHYSICAL ACIA ADDR.
FB65 86 02               LDA A    #2       LOADS CONTROL TEST BITS.
FB67 A5 00        CAS01  BIT A    0,X      TESTS IF ACIA BUFFER EMPTY.
FB69 27 FC               BEQ      CAS01    LOOPS BACK UNTIL READY.
FB6B 32                  PUL A    .        GETS ORIG CHAR.
FB6C A7 01               STA A    1,X      STORS CHAR INTO ACIA BUFFER.
FB6E 7D 003A             TST      NOPRNT   TESTS IF PRINTOUT ALLOWED.
FB71 27 03               BEQ      CSOEXT   SKIPS PRINTING IF A 0.
FB73 B7 E01E             STA A    $E01E    DISPLAYS CHAR ON SCREEN.
FB76 39           CSOEXT RTS


                  *      TURNON TURNS ON THE CASSETTE DRIVE.
                  *
                  *
FB77 DE 38        TURNON LDX      ACIANO   LOADS CASSETTE ACIA ADDRESS.
FB79 86 81               LDA A    #ON
FB7B A7 00               STA A    0,X      TURNS ACIA ON.
FB7D 39                  RTS
```

```
                *       CASIN READS IN A CHARACTER FROM THE CASSETTE TAPE
                *       INTO THE A ACCUMULATOR.
                *
                *
FB7E DE 38      CASIN   LDX     ACIANO    X GETS THE ACIA ADDRESS.
FB80 86 01              LDA A   #1        LOADS TEST BITS.
FB82 A5 00      CASIN1  BIT A   0,X       TESTS IF ACCIA BUFFR FULL.
FB84 27 FC              BEQ     CASIN1    SKIPS BACK IF NOT IN YET.
FB86 A6 01              LDA A   1,X       LOADS IN CHAR FROM TAPE.
FB88 7D 003A            TST     NOPRNT    TESTS IF PRINT IS OFF.
FB8B 27 03              BEQ     CINEXT    SKIPS DISPLAYING IF 0.
FB8D B7 E01F            STA A   $E01F     DISPLAYS CHR ON TV.
FB90 39         CINEXT  RTS               .


                *       RDBLK READS IN A BLOCK FROM THE CASSETTE
                *       TAPE INTO BUFFER MEMORY.
                *
                *
FB91 8D E4      RDBLK   BSR     TURNON    TURNS ON TAPE DRIVE.
FB93 8D 20      RDMOD   BSR     RDHDR     READS IN THE HEADER.
FB95 8D 55              BSR     RDBFR     READS IN DATA INTO BUFFER.
                *        FOLLOWING READS IN TRAILER AND CHECKS CHECKSUM.
FB97 4F         RDTLR   CLR A   -         LOADS A 0 FOR A GOOD READ.
FB98 97 09              STA A   CSTATS    SETS STATUS BYTE TO NO ERR.
FB9A 8D E2              BSR     CASIN     INPUTS END-OF-BLOCK CHAR.
FB9C 81 17              CMP A   #ETB      ETB DISPLAYS AS A "W".
FB9E 27 04              BEQ     RTLR1     SKIPS IF NO ETB ERROR.
FBA0 86 54              LDA A   #ERR4     LOADS TRAILER ERROR CODE.
FBA2 20 07              BRA     RTLR2     SKIPS TO STORE ERROR CODE.
FBA4 8D D8      RTLR1   BSR     CASIN     READS IN CHECKSUM.
FBA6 11                 CBA               TESTS CHECKSUM.
FBA7 27 07              BEQ     TRNOFF    SKIPS IF OK.
FBA9 86 43              LDA A   #ERR5     SETS CHECKSUM ERROR CODE.
FBAB 97 09      RTLR2   STA A   CSTATS    SETS ERROR STATUS BYTE.
FBAD B7 E05F            STA A   $E05F     DISP.  ERR CODE ON SCREEN.
                *        END OF TRAILER READ IN.
                *        FOLLOWING TURNS OFF THE CASSETTE DRIVE.
FBB0 86 51      TRNOFF  LDA A   #OFF      LOADS COMMAND TO TURN
FBB2 A7 00              STA A   0,X       OFF ACIA CASSETTE DRIVE.
FBB4 39                 RTS               .
```

```
                      *       RDHDR FINDS THE START OF THE BLOCK ON THE TAPE,
                      *       FIGURES THE BUFFER END AND CHECKS THE NAME.
                      *
                      *
FBB5 8D C7      RDHDR  BSR     CASIN     FOLLOWING FINDS THE
FBB7 81 16      RDHDR1 CMP A   #SYN      START OF THE HEADER.
FBB9 26 FA             BNE     RDHDR     .
FBBB 8D C1             BSR     CASIN     TESTS FOR HEADR CHAR.
FBBD 81 1B             CMP A   #ESC      START-OF-HEADER.
FBBF 26 F6             BNE     RDHDR1    GOES BACK IF NOT GOOD HDR.
FBC1 8D BB             BSR     CASIN     READS IN HI BYTE OF LEN.
FBC3 16                TAB               SAVES HI LENGTH BYTE.
FBC4 8D B8             BSR     CASIN     INPUTS LO SIZE (LEN) BYTE.
FBC6 9B 3D             ADD A   BFRPTR+1  FORMS POINTER TO THE
FBC8 D9 3C             ADC B   BFRPTR    TOP BYTE OF THE BUFFER.
FBCA 97 3F             STA A   BFRSZE+1  SAVES THE HI BUFF. PTR.
FBCC D7 3E             STA B   BFRSZE    TO THE CASSETTE BUFFER.
FBCE 8D AE             BSR     CASIN     READS IN BLOCK NAME.
FBD0 36                PSH A   -         SAVES FIRST CHAR OF NAME.
FBD1 8D AB             BSR     CASIN     READ IN SECOND CHAR INTO A.
FBD3 D6 3A             LDA B   NOPRNT    TESTS IF PRINT IS OFF.
FBD5 33                PUL B   -         RESTORES FIRST NAME CHAR.
FBD6 27 06             BEQ     RHDR1     SKIPS IF PRINT FLAG IS 0.
FBD8 F7 E03E           STA B   $E03E     DISPLAYS BLOCK NAME ON
FBDB B7 E03F           STA A   $E03F     THE CRT SCREEN.
FBDE 7D 0033    RHDR1  TST     BLKNAM    TESTS IF NAME IS CHECKED.
FBE1 27 08             BEQ     RHDR2     SKIPS IF NO NAME CHECK.
FBE3 D1 33             CMP B   BLKNAM    TESTS FIRST CHAR OF NAME.
FBE5 26 CE             BNE     RDHDR     SKIPS BACK IF BAD NAME
FBE7 91 34             CMP A   BLKNAM+1  TESTS SECOND NAME CHAR.
FBE9 26 CA             BNE     RDHDR     SKIPS BACK IF BAD NAME.
FBEB 39         RHDR2  RTS               .




                      *       RDBFR READS DATA INTO THE MEMORY BUFFER FROM
                      *       THE CASSETTE.
                      *
                      *


FBEC 5F         RDBFR  CLR B   -         INIT B FOR CHECKSUM.
FBED DE 3C             LDX     BFRPTR    LOADS START OF BUFFER.
FBEF DF 00      RBFR1  STX     TMP       .
FBF1 8D 8B             BSR     CASIN     A GETS CHAR READ IN.
FBF3 DE 00             LDX     TMP       X GETS BUFFER PTR.
FBF5 A7 00             STA A   0,X       STORS CHAR INTO BUFFER.
FBF7 1B                ABA               A GETS A+B.
FBF8 16                TAB               B GETS A.
FBF9 9C 3E             CPX     BFRSZE    TESTS IF BUFFER FULL.
FBFB 27 EE             BEQ     RHDR2     SKIPS TO EXIT IF ALL IS IN.
FBFD 08                INX               INC TO NEXT CHAR POSITIN.
FBFE 26 EF             BNE     RBFR1     GOES BACK IF ANY LEFT.
                       END
```

2B


```
                NAM     PDS-SYS2N
                OPT     O,NOG
*
*               PDS SYSTEM 2N  CASSETTE DRIVERS  (SYS2NF)
*
*               PROGRAMMED BY ERIC JAMESON
*
*
*
*               COPYRIGHT 1976 SPHERE CORPORATION
*               940 N.   400 EA.;  NORTH SALT LAKE, UTAH  84054 .
*
*               P. O. BOX 213;  BOUNTIFUL, UTAH  84010
*
*
*
*
*               SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
*               DISTRIBUTION AND USE OF THE PDS SOFTWARE.
*               NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
*               WRITTEN PERMISSION OF SPHERE CORP.
*
*
*
*
*
* THE PROGRAM DEVELOPMENT SYSTEM (PDS SYS2N) IS A SET OF
* PROGRAMS RESIDING ON ERASABLE PROGRAMMABLE READ ONLY
* MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS
* SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE
* DEVELOPMENT OF COMPUTER PROGRAMS.
*      TOWARD THIS END, THE 5 PDS EPROMS CONTAIN A CURSOR
* BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING
* AID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16
* BIT MULTIPLY AND DIVIDE, ASCII-TO-BINARY, AND
* BINARY-TO-ASCII ROUTINES, AND ROUTINES TO DO
* INPUT AND OUTPUT TO THE AUDIO CASSETTE.
*
*
*
*      THE SYS2N SOFTWARE IS AN UPGRADE OF THE PDS V3A
* (V3N & V3D) SOFTWARE DESIGNED TO RUN WITH THE CASSETTE
* SYSTEM.  THERE ARE TWO VERSIONS OF THE SYS2 SOFTWARE:
* SYS2N WHICH RUNS WITH THE NEW KEYBOARD AND THE SYS2A
* WHICH RUNS WITH THE ORIGIONAL (KBD/1A) KEYBOARD.  THE
* MAIN DIFFERENCE BETWEEN THE V3A AND SYS2 VERSIONS ARE
* THAT A FIFTH EPROM HAS BEEN ADDED AND THAT THE MINI-
* ASSEMBLER HAS BEEN DELEATED AND REPLACED WITH A SET OF
* COMMANDS TO DO LOADING AND DUMPING OF CASSETTE TAPES.
```

The SYS2NF prom is a version of the SYS2N cassette prom with a software bug fixed. This bug would cause the next block of a multiblock read to be skipped if the checksum on the preceeding block was a 16 and the tape had not been previously used. The change is on page 1Ø where the ESC test and branch now branches to RDHDR1 instead of RDHDR.

```
                *              MEMORY MAP
                *
       0000     TMP     EQU    $00
       0002     TMP1    EQU    $02
       0004     ARB     EQU    $04        16 BIT ACC. PSEUDO REG B.
       0004     AR3     EQU    $04        _HI BYTE OF ARB.
       0005     AR2     EQU    $05        _LO BYTE OF ARB.
       0006     ARA     EQU    $06        16 BIT ARITH PSEUDO REG A.
       0006     AR1     EQU    $06        _HI BYTE OF ARA.
       0007     AR0     EQU    $07        _LO BYTE OF ARA.
       0008     DIGIT   EQU    $08        BYTE USED BY ASCBIN FOR TMP.
       0009     CSTATS  EQU    $09        CASSETTE I-O STATUS BYTE.
       000A     OUTEND  EQU    $0A        END OF OUTPUT BUFFER TEXT.
       000C     BUFADR  EQU    $0C        START OF I/O BUFFER (PTR)
       000E     BUFEND  EQU    $0E        PTR. TO END OF I/O BUFFR.
       0011     OUTBUF  EQU    $11        START OF OUTPUT BUFFER.
       0013     CASNUM  EQU    $13        PHYSICAL CASSETTE NUMBR.
       0014     SRCADR  EQU    $14        SOURCE FOR TEXT MOVES.
       0016     DSTADR  EQU    $16        DEST. ADDR. FOR TEXT MOVE.
       001A     ENDMEM  EQU    $1A        LAST ADDRES OF REAL MEMORY.
       001C     CSRPTR  EQU    $1C        PTR TO CURSER ON SCREEN.
       001E     BUFPTR  EQU    $1E        TEMP PTR USED BY OUTSTR.
       0020     BUFFLO  EQU    $20        PTR TO END OF LOW EDIT TXT.
       0022     BUFFHI  EQU    $22        PTR TO START OF HI TEXT.
       0024     SCNPTR  EQU    $24        PTR. TO BUFFRD TXT START.
       0026     SPCASM  EQU    $26        PTR TO ASSMBLR SOURCE CODE.
       002A     ONDVAL  EQU    $2A        HAS ASSMBLR OPERND VALUE.
       002C     SYMVAL  EQU    $2C        VALUE PUT IN ASSM. SYMTBL.
       002E     BRKSAV  EQU    $2E        TEMP SAVE FOR BRKPT DATA.
       0030     BRKADR  EQU    $30        ADDDRESS OF BREAKPOINT.
       0032     EDIT    EQU    $32        0 IF EDITOR IS NOT RUNNING.
       0033     BLKNAM  EQU    $33        CASSETTE BLOCK NAME.
       0035     IOBUFF  EQU    $35        I-O BUFFER FOR DEBUGGER.
       0038     ACIANO  EQU    $38        SYS2N CASSETTE ACIA ADDR.
       003A     NOPRNT  EQU    $3A        CASSETTE NAME PRINT FLAG.
       003B     BLKTYP  EQU    $3B        CASSETTE BLOCK TYPE CODE.
       003C     BFRPTR  EQU    $3C        ADDR. OF I-O BUFF. FOR CASS.
       003E     BFRSZE  EQU    $3E        LENGTH OF CASS. BUFFER.
       0040     PCVAL   EQU    $40        PROGRAM COUNTER FOR ASSM.


                *       FOLLOWING ARE VARIABLE VALUES.
                *
       26F0     TIMEA   EQU    $26F0      TIMER COUNTER.
       0081     ON      EQU    $81        ACIA VALUE TO TURN ON CASS.
       0051     OFF     EQU    $51        ACIA VALUE TO STOP CASS.
       0003     ETX     EQU    $03        END-OF-TEXT.
       0016     SYN     EQU    $16        SYNCHRONISE.
       0017     ETB     EQU    $17        END-OF-TRANSMISSION-BLOCK.
       001B     ESC     EQU    $1B        ESCAPE TO NONSTANDARD ADR.
       0054     ERR4    EQU    $'T        SET FOR TRAILER ERROR.
       0043     ERR5    EQU    $'C        SET FOR CHECKSUM ERR.
       5161     TIME    EQU    20833      TIME CNTR FOR 1/4 SEC.
       0009     TIMCNT  EQU    9          TIMES FOR 2 & 1/4 SECONDS.
0
```

```
                *
FB00                ORG     $FB00
                *
                *
                *           CASSETTE I-O DRIVERS
                *
                *       THE CASSETTE DRIVERS LOAD AND DUMP A BLOCK OF DATA
                * TO AND FROM THE CASSETTE.   THEY HANDLE BOTH THE HEADER
                * AND TRAILER FORMATTING.
                *       THE DRIVERS ARE SET UP AS A SET OF SUBROUTINES
                * CALLED BY THE EXECUTIVE OR THE USER'S PROGRAMS.
                *
                *
                *
                *         LOW MEMORY ADDRESSES USED BY THE DRIVERS ARE:
                *
                *         CSTATS  AT  09  . USED TO STORE INPUT ERR CODE.
                *         BLKNAM  AT  33    2 CHAR. NAME OF BLOCK.
                *         ACIANO  AT  38    PTR. TO THE ACIA CURRENTLY USED.
                *         NOPRNT  AT  3A    PRINT FLAG FOR BLOCK NAME.
                *         BFRPTR  AT  3C    START OF CASSETTE I-O BUFFER.
                *         BFRSZE  AT  3E    END OF CASSETTE DATA BUFFER.
                *
                *
                *       THE DRIVERS CAN BE RUN FROM THE PDS-SYS2N SOFTWARE
                * SYSTEM, WHICH CONTAINS THE CASSETTE LOAD AND DUMP
                * COMMANDS IN THE EXEC, OR FROM THE PDS-V3A, WHERE THE
                * DRIVERS ARE CALLED THROUGH THE DEBUGGER.
                *
                *       THE DRIVERS DISPLAY PERTINENT DATA ON THE SCREEN
                * WHEN WRITING OR READING FROM THE CASSETTES.   ON A WRITE,
                * THE CHARACTERS BEING WRITTEN ONTO THE CASSETTE ARE
                * DISPLAYED IN THE SECOND CHARACTER POSITION FROM THE
                * UPPER RIGHT HAND CORNER OF THE SCREEN.   ON A READ, THE
                * CHARACTERS BEING READ IN ARE DISPLAYED IN THE UPPER
                * RIGHT HAND CORNER OF THE SCREEN.   THE NAME OF THE BLOCK
                * CURRENTLY BEING READ OR SEARCHED OVER IS DISPLAYED ON THE
                * RIGHT HAND SIDE OF THE SECOND LINE.   THE ERROR CODE FOR
                * A READ IS DISPLAYED ON THE RIGHT SIDE OF THE THIRD LINE.
                * NO CHANGE IN CHARACTER MEANS THAT THE READ WAS O.K.
                * A "C" MEANS THERE WAS A CHECKSUM ERROR ON THE BLOCK.
                * A "T" MEANS THAT THE WRONG NUMBER OF BYTES WERE READ
                *  INTO THE BUFFER (TRAILER ERROR).   THIS WOULD OF COURSE
                *  IMPLY A CHECKSUM ERROR ALONG WITH THE TRAILER ERROR.
                *
                *       THE CSTATS (CASSETTE STATUS) BYTE WILL CONTAIN A 0
                * IF THE BLOCK READ IN WAS O.K.   IF IT WAS A BAD READ, IT
                * WILL CONTAIN A 54 FOR TRAILER ERRROR OR A 43 FOR A
                * CHECKSUM ERROR CODE UPON EXIT FROM THE ROUTINE.
                *
                *       SETTING THE NOPRNT (NO PRINTING) FLAG TO A 0 WILL
                * STOP THE DISPLAY OF CHARACTERS ON THE SCREEN DURING
                * CASSETTE READ AND WRITE, EXCEPT FOR THE T & C ERROR
                * CODES, WHICH ARE ALWAYS DISPLAYED WHEN THEY OCCUR.
```

```
*        IF THE FIRST BYTE OF BLKNAM (BLOCK NAME) IS A 0
* WHEN THE READ BLOCK ROUTINE IS ENTERED, THE NEXT BLOCK
* WILL BE READ FROM TAPE NO MATTER WHAT THE NAME OF THE
* TAPE BLOCK IS.  ON THE SYS2N EXEC, A CONTROL SPACE
* CHARACTER CAN BE TYPED IN AS THE FIRST CHARACTER OF THE
* NAME IN THE LOAD BLOCK COMMAND.  THUS, A (CNTL L)(CNTL
* SPACE)(X) WOULD READ IN THE NEXT BLOCK ON THE TAPE.
*
*
*
*        THE TAPE FORMAT IS:
*          SYN
*          SYN
*          SYN
*          ESC
*          HI BYTE OF 16 BIT BLOCK LENGTH
*          LOW BYTE OF BLOCK LENGTH
*          FIRST CHAR OF BLOCK NAME
*          SECOND CHARACTER OF NAME
*          DATA
*          .
*          .
*          .
*          DATA
*          ETB
*          CHECKSUM
*          CHECKSUM
*          CHECKSUM
*          CHECKSUM
*
*
*
*        THE CHECKSUM IS CALCULATED FROM THE DATA, WHICH
* IS READ IN FROM THE CASSETTE BUFFER RESIDING IN MEMORY.
*
*
*        THE FORMAT FOR THE CASSETTE BUFFER IS:
*
*          BFRPTR                        BFRSZE
*
*
*         |C|A|S|S|E|T|T|E| | | |B|U|F|F|E|R|
*
*
*        WHEN DATA IS READ IN, THE READ ROUTINE SETS 'BFRSZE'
* TO POINT TO THE LAST CHARACTER READ INTO THE BUFFER.
* NOTE THAT THERE IS NO OVERFLOW CHECK WHEN DATA IS READ
* INTO MEMORY.  ON OUTPUT TO THE CASSETTE, THE BLOCK
* LENGTH IS CALCULATED FROM THE BFRPTR AND BFRSZE POINTERS.
*
*
*        THE READ BLOCK AND WRITE BLOCK ROUTINES AUTOMATICALLY
* TURN ON AND OFF THE CASSETTE.
```

```
*
*       THE ADDRESSES OF THE ROUTINES FOR CONTROLLING
*       THE CASSETTE ARE AS FOLLOWS:
*
*       INTLZ  - FB00 : INITIALIZES THE ACIA FOR USE.
*       WRTBLK - FB2D : WRITES A  FORMATTED BLOCK TO CASSETTE
*       RDBLK  - FB91 : READS IN A FORMATTED BLOCK OF TAPE
*       WRTMOD - FB2F : WRITES A BLOCK TO A RUNNING CASSETTE
*       RDMOD  - FB93 : READS A BLOCK FROM A RUNNING TAPE.
*       CASOUT - FB62 : WRITES OUT ONTO TAPE THE BYTE IN A.
*       CASIN  - FB7E : READS IN A BYTE INTO ACCUM. A
*       TURNON - FB77 : TURNS ON THE CASSETTE DRIVE.
*       TRNOFF - FBB0 : TURNS OFF THE CASSETTE DRIVE WHEN THE
*                       POINTR TO ACIA IS PASSED IN THE X REG.
*
*      NOTE THAT ALL THE ABOVE ROUTINES ARE SUBROUTINES THAT
*       ARE ENTERED BY A JSR OR BSR CALL.
*
*
*
*
*                  USING THE CASSETTE DRIVERS.
*
*       IF THE EPROMS ON THE CPU BOARD ARE THE SYS2N OR THE
* SYS2D PROMS, THE DRIVERS ARE USED BY THE I (INITIALIZE),
* L (LOAD FROM CASSETTE) AND S (STORE ONTO CASSETTE)
* COMMANDS THAT ARE A PART OF THE SYS-2 EXECUTIVE.
*
*       IF THE CPU PROMS BEING USED ARE THE PDS-V3A OR THE
* PDS-V3N OR V3D PROM SETS, THEN THE USER MUST SET UP THE
* POINTERS AND CALL THE ROUTINES HIMSELF, USING THE
* DEBUGGER ON THE CPU PROMS.  (NOTE THAT THE V3A AND V3N
* DEBUGGERS VARY, FOR INSTANCE THE V3A GO COMMAND IS A 'G'
* WHILE THE V3N GO COMMAND IS A 'CNTL G'.  REFER TO THE
* USERS MANUAL FOR DETAILES.)
*       THE FIRST THING TO DO IS OPEN THE LOW MEMORY LOCATIONS
* USED BY THE DRIVERS AS FLAGS AND POINTERS AND INITIALIZE
* THE LOCATIONS.  THE ACIANO, BLKNAM, BFRPTR AND BFRSZE
* SHOULD NOW BE GIVEN VALUES.
*       ACIANO WOULD POINT TO F050 FOR THE FIRST CASSETTE AND
* TO F060 FOR THE SECOND CASSETTE DRIVE.
*       A SIMPLE PROGRAM TO CALL THE ROUTINES SHOULD NOW BE
* WRITTEN.  OPEN LOCATION 900.  TYPE IN THROUGH THE DEBUGGER
* THE INSTRUCTIONS JSR BDXX, JMP FE4F.  XX IS THE SECOND BYTE
* OF THE DESIRED DRIVER ROUTINE, I.E. 00 FOR INITIALLIZATION
* OR 91 FOR READING A BLOCK OR 2D FOR WRITING A BLOCK.  THUS,
* TO INITIALIZE AN ACIA, ACIANO WOULD BE LOADED WITH
* THE ADDRESS OF THE ACIA AND THE USER WOULD THEN JUMP TO THE
* ROUTINE AT 900 BY OPENING 900 AND JUMPING TO IT WITH
* THE 'G' COMMAND.  THE ROUTINE WOULD BE AS FOLLOWS:
*
*CALL 900  BD      JSR
*     901  FB      HI BYTE OF ROUTINE AADDRESS
*     902  00      LOW BYTE OF DRIVER ADDRESS.
*     903  7E      JMP
*     904  FE      ADDRESS OF THE DEBUGGER ON
*     905  4F      THE PDS V3A PROM SET
```

0

```
*       THE ACIA WOULD NOW BE INITIALIZED AND CONTROL WOULD
* HAVE RETURNED TO THE DEBUGGER.   TO READ IN A BLOCK,
* THE NAME WOULD BE PUT IN BLKNAM AND LOCATION 902 WOULD
* BE CHANGED TO 91.  THE USER WOULD THEN  JUMP TO 900.
*
*       TO TEST THE CASSETTE, SET BFRPTR TO E060 AND BFRSZE
* TO E0DF.   THIS WILL ALLOW THE USER TO WRITE OUT DATA
* FROM THE FOURTH, FIFTH, SIXTH AND SEVENTH LINE OF THE
* CRT DISPLAY AND THEN READ IT BACK ONTO THE DISPLAY.   DATA
* CAN BE TYPED ONTO THE SCREEN BY OPENING A LOCATION WITH
* THE DEBUGGER AND THEN MOVING THE CURSOR AROUND THE SCREEN
* TO CHANGE THE CHARACTERS.   THIS IS POSSIBLE BECAUSE THE
* ROUTINE TO INPUT AN ADDRESS CALLS THE EDITOR FOR INPUT.
*
*       THE DRIVER ROUTINES CAN ALSO BE USED TO PERFORM
* I-O WITH A MODEM OR TELETYPE.   THE MAIN HARDWARE
* DIFFERENCE BETWEEN THE CASSETTE AND MODEM/TTY IS THAT THE
* CASSETTE HAS A DIVIDE BY 16 CLOCK AND IS UNIDIRECTIONAL
* WHILE THE MODEM/TTY HAVE A DIVIDE BY 64 CLOCK AND CAN BE
* BIDIRECTIONAL.   BECAUSE OF THE CLOCK CHANGE THE ACIA MUST
* BE TURNED ON WITH A DIFFERENT VALUE BEFORE THE READ BLOCK
* OR WRITE BLOCK ROUTINES ARE ENTERED.   TO TURN ON THE ACIA
* STORE THE VALUE 'B2' INTO LOCATION F050 OR F060.   ONCE
* IT IS TURNED ON, EITHER CASIN OR CASOUT MAY BE CALLED
* REPEATEDLY OR WRITE MODEM BLOCK (WRTMOD) OR READ MODEM
* (RDMOD) CAN BE CALLED ONCE.   WRTMOD & RDMOD ARE THE SAME
* AS WRTBLK & RDBLK EXCEPT THEY DO NOT TURN ON THE ACIA.   A
* PROGRAM TO READ IN A BLOCK OF DATA FROM A MODEM OR TTY
* WOULD THUS BE (PLACED BEFORE THE 'CALL' ROUTINE):
*     8FA  DE 38  LDX     ACIANO    LOADS ACIA POINTTER.
*     8FC  86 B2  LDA A   #$B2      LOADS STARTUP VALUE.
*     8FE  A7 00  STA A   0,X       PUTS START CODE INTO ACIA.
* LOCATION 902 WOULD NOW BE 2F FOR WRITING AND 93 TO READ.
*       TO READ IN A CHARACTER FROM THE TELETYPE TURN ON THE
* ACIA AND GO TO THE FOLLOWING ROUTINE:
*     TTYIN  JSR     CASIN    READS IN A CHAR FROM KEYBOARD.
*            JMP     CASOUT   TYPES OUT CHAR ON PRINTER.
* THE RS232 SHOULD HAVE BEEN STRAPPED TO HALF DUPLEX.
*
*
*       IT IS TO BE STRESSED THAT  THE RELIABILITY OF THE
* CASSETTE CONTROLLER DEPENDS ON THE ADJUSTMENT OF THE
* TRIMMER ON THE SIM BOARD.   IF THE TRIMMER IS OUT OF
* ADJUSTMENT THE DATA WILL NOT READ IN PROPERLY.   BESIDES
* USING THE OSCILLOSCOPE TO ADJUST THE TRIMMER, IT CAN ALSO
* BE ADJUSTED BY READING IN A STRING OF SINGLE CHARACTERS
* FROM  THE CASSETTE AND ADJUSTING IT UNTIL THE CHARACTERS
* SYNC IN PROPERLY.   THE BEST CHARACTER TO USE IS A STRING
* OF 'U'S.   TO READ IN THE STRING FOR TESTING USE, WRITE A
* LOOP TO GET A CHARACTER FROM THE CASSETTE AND THEN
* DISPLAY THAT CHARACTER.   A SAMPLE ROUTINE TO DO THIS IS:
*     A.  JSR   $FB7E    LOADS A WITH CASSETTE CHAR.
*         JSR   $FCAD    PDS-V3A PUTCHR ROUTINE.
*         TST   $F001    TESTS KEYBOARD FOR A KEY.
*         BPL   A        SKIPS BACK IF NO INPUT.
* THE ABOVE ROUTINE WOULD INPUT CHARACCTERS UNTIL A KEY
* ON THE KEYBOARD WAS DEPRESSED.
0
```

B2 will produce a signel of eight data bits and 2 stop bits. If the TTY operates on a different code, look up the proper initialization value in the ACIA section of the chip description appendex.

```
                        *       INTLZ INITIALIZES THE ACIA CONTROLLER FOR
                        *       A SPECIFIC TAPE UNIT AT MOUNT TIME.
                        *
FB00 DE 38      INTLZ   LDX     ACIANO     X GETS ACIA ADDRESS.
FB02 86 13              LDA A   #$13       RESETS THE ACIA.
FB04 A7 00              STA A   0,X        .
FB06 86 51              LDA A   #OFF       SETS ACIA TO
FB08 A7 00              STA A   0,X        /16 2 STOPS BIT FORMAT.
FB0A 39                 RTS

                        *       WRTHDR FORMATS THE HEADER ON THE TAPE.
                        *
FB0B 86 16      WRTHDR  LDA A   #SYN       PUTS SYNC CHARS ONTO TAPE.
FB0D 8D 53              BSR     CASOUT     .
FB0F 8D 51              BSR     CASOUT     .
FB11 8D 4F              BSR     CASOUT     .
FB13 86 1B              LDA A   #ESC       .
FB15 8D 4B              BSR     CASOUT     .
FB17 96 3E              LDA A   BFRSZE     FOLLOWING OUTPUTS LENGTH.
FB19 D6 3F              LDA B   BFRSZE+1   LOADS LO BYTE OF END PTR.
FB1B D0 3D              SUB B   BFRPTR+1   SUBS LO BYTE OF BEGIN PTR.
FB1D 92 3C              SBC A   BFRPTR     SUBS HI BYTE OF START PTR.
FB1F 8D 41              BSR     CASOUT     OUTPUTS HI LENGTH BYTE.
FB21 17                 TBA                LOADS LO BYTE OF LENGTH.
FB22 8D 3E              BSR     CASOUT     OUTPUTS LOW LENGTH BYTE.
FB24 96 33              LDA A   BLKNAM     PUTS OUT NAME OF BLOCK.
FB26 8D 3A              BSR     CASOUT     .
FB28 96 34              LDA A   BLKNAM+1   .
FB2A 8D 36              BSR     CASOUT     PUTS OUT LAST OF NAME.
FB2C 39                 RTS                RETURNS BACK TO WRTBLK.

                        *       WRTBLK WRITES OUT A BLOCK OF DATA TO THE CASSETTE.
                        *
FB2D 8D 48      WRTBLK  BSR     TURNON     TURNS ON THE CASSETTE.
                        *        FOLLOWING WAITES FOR CASSETTE TO GET UP TO SPEED.
FB2F C6 09      WRTMOD  LDA B   #TIMCNT    LOADS TIME LOOP COUNTER.
FB31 CE 5161    TIME1   LDX     #TIME      MASTER TIME LOOP (1/4 SEC).
FB34 09         TIME2   DEX                COUNTS CYCLES OF LOOP.
FB35 26 FD              BNE     TIME2      TESTS FOR FIRST TIME OUT.
FB37 5A                 DEC B   -          COUNTS TIMES IN LOOP.
FB38 26 F7              BNE     TIME1      SKIPS BACK UNTIL DONE.
                        *        THE TIME LOOP IS NOW FINISHED.
FB3A 8D CF              BSR     WRTHDR     WRITES HEADER ON THE TAPE.
FB3C 8D 10              BSR     WRTBFR     WRITES OUT BUFFER DATA.
                        *        FOLLOWING WRITES THE TRAILER OUT ONTO THE TAPE.
FB3E 86 17      WRTTLR  LDA A   #ETB       OUTPUTS END-OF-BLOCK CHAR.
FB40 8D 20              BSR     CASOUT     ETB IS DISPLAYED AS A "W".
FB42 17                 TBA                A GETS CHECKSUM FROM B.
FB43 8D 1D              BSR     CASOUT     OUTPUTS THE CHECKSUM.
FB45 8D 1B              BSR     CASOUT     OUTPUTS TRAILER FILLER
FB47 8D 19              BSR     CASOUT     BYTES.
FB49 8D 17              BSR     CASOUT     .
                        *        END OF TRAILER WRITING ROUTINE.
FB4B 8D 63              BSR     TRNOFF     HALTS CASSETTE DRIVE.
FB4D 39                 RTS                -
```

```
                    *       WRTBFR WRITES OUT THE CONTENTS  OF THE
                    *       BUFFER ONTO THE CASSETTE TAPE.
                    *
                    *
FB4E 5F             WRTBFR CLR B    -         INIT CHECKSUM COUNT.
FB4F DE 3C                 LDX      BFRPTR    ACIA CONTROL MASK.
FB51 A6 00          WBFR1  LDA A    0,X       LOADS CHAR. FROM BUFFER.
FB53 DF 00                 STX      TMP       SAVES BUFFER PTR.
FB55 8D 0B                 BSR      CASOUT    PUTS CHAR ONTO CASSETTE.
FB57 1B                    ABA                A GETS A+B.
FB58 16                    TAB                B GETS A+B.
FB59 DE 00                 LDX      TMP       RESTORES PTR. INTO BUFFR.
FB5B 9C 3E                 CPX      BFRSZE    TESTS IF BUFFER EMPTY.
FB5D 27 17                 BEQ      CSOEXT    EXITS WHEN EMPTY.
FB5F 08                    INX                INC POINTER.
FB60 26 EF                 BNE      WBFR1     SKIPS BACK IF CHARS. LEFT.


                    *       CASOUT TAKES THE CHAR IN A AND PUTS IT OUT
                    *       ONTO THE CASSETTE TAPE.
                    *
                    *
FB62 36             CASOUT PSH A    .         SAVES CHAR TO READ OUT.
FB63 DE 38                 LDX      ACIANO    X GETS PHYSICAL ACIA ADDR.
FB65 86 02                 LDA A    #2        LOADS CONTROL TEST BITS.
FB67 A5 00          CAS01  BIT A    0,X       TESTS IF ACIA BUFFER EMPTY.
FB69 27 FC                 BEQ      CAS01     LOOPS BACK UNTIL READY.
FB6B 32                    PUL A    .         GETS ORIG CHAR.
FB6C A7 01                 STA A    1,X       STORS CHAR INTO ACIA BUFFER.
FB6E 7D 003A               TST      NOPRNT    TESTS IF PRINTOUT ALLOWED.
FB71 27 03                 BEQ      CSOEXT    SKIPS PRINTING IF A 0.
FB73 B7 E01E               STA A    $E01E     DISPLAYS CHAR ON SCREEN.
FB76 39             CSOEXT RTS


                    *        TURNON TURNS ON THE CASSETTE DRIVE.
                    *
                    *
FB77 DE 38          TURNON LDX      ACIANO    LOADS CASSETTE ACIA ADDRESS.
FB79 86 B1                 LDA A    #ON
FB7B A7 00                 STA A    0,X       TURNS ACIA ON.
FB7D 39                    RTS
```

```
                  *        CASIN READS IN A CHARACTER FROM THE CASSETTE TAPE
                  *        INTO THE A ACCUMULATOR.
                  *
                  *
FB7E DE 38        CASIN   LDX      ACIANO     X GETS THE ACIA ADDRESS.
FB80 86 01                LDA A    #1         LOADS TEST BITS.
FB82 A5 00        CASIN1  BIT A    0,X        TESTS IF ACCIA BUFFR FULL.
FB84 27 FC                BEQ      CASIN1     SKIPS BACK IF NOT IN YET.
FB86 A6 01                LDA A    1,X        LOADS IN CHAR FROM TAPE.
FB88 7D 003A              TST      NOPRNT     TESTS IF PRINT IS OFF.
FB8B 27 03                BEQ      CINEXT     SKIPS DISPLAYING IF 0.
FB8D B7 E01F              STA A    #E01F      DISPLAYS CHR ON TV.
FB90 39           CINEXT  RTS



                  *        RDBLK READS IN A BLOCK FROM THE CASSETTE
                  *        TAPE INTO BUFFER MEMORY.
                  *
                  *
FB91 8D E4        RDBLK   BSR      TURNON     TURNS ON TAPE DRIVE.
FB93 8D 20        RDMOD   BSR      RDHDR      READS IN THE HEADER.
FB95 8D 55                BSR      RDBFR      READS IN DATA INTO BUFFER.
                  *         FOLLOWING READS IN TRAILER AND CHECKS CHECKSUM.
FB97 4F           RDTLR   CLR A    -          LOADS A 0 FOR A GOOD READ.
FB98 97 09                STA A    CSTATS     SETS STATUS BYTE TO NO ERR.
FB9A 8D E2                BSR      CASIN      INPUTS END-OF-BLOCK CHAR.
FB9C 81 17                CMP A    #ETB       ETB DISPLAYS AS A "W".
FB9E 27 04                BEQ      RTLR1      SKIPS IF NO ETB ERROR.
FBA0 86 54                LDA A    #ERR4      LOADS TRAILER ERROR CODE.
FBA2 20 07                BRA      RTLR2      SKIPS TO STORE ERROR CODE.
FBA4 8D D8        RTLR1   BSR      CASIN      READS IN CHECKSUM.
FBA6 11                   CBA                 TESTS CHECKSUM.
FBA7 27 07                BEQ      TRNOFF     SKIPS IF OK.
FBA9 86 43                LDA A    #ERR5      SETS CHECKSUM ERROR CODE.
FBAB 97 09        RTLR2   STA A    CSTATS     SETS ERROR STATUS BYTE.
FBAD B7 E05F              STA A    #E05F      DISP.  ERR CODE ON SCREEN.
                  *         END OF TRAILER READ IN.
                  *         FOLLOWING TURNS OFF THE CASSETTE DRIVE.
FBB0 86 51        TRNOFF  LDA A    #OFF       LOADS COMMAND TO TURN
FBB2 A7 00                STA A    0,X        OFF ACIA CASSETTE DRIVE.
FBB4 39                   RTS
```

```
                  *         RDHDR FINDS THE START OF THE BLOCK ON THE TAPE.
                  *         FIGURES THE BUFFER END AND CHECKS THE NAME.
                  *
                  *
FBB5 8D C7        RDHDR     BSR     CASIN      FOLLOWING FINDS THE
FBB7 81 16        RDHDR1    CMP A   #SYN       START OF THE HEADER.
FBB9 26 FA                  BNE     RDHDR      .
FBBB 8D C1                  BSR     CASIN      TESTS FOR HEADR CHAR.
FBBD 81 1B                  CMP A   #ESC       START-OF-HEADER.
FBBF 26 F6                  BNE     RDHDR1     GOES BACK IF NOT GOOD HDR.
FBC1 8D BB                  BSR     CASIN      READS IN HI BYTE OF LEN.
FBC3 16                     TAB                SAVES HI LENGTH BYTE.
FBC4 8D B8                  BSR     CASIN      INPUTS LO SIZE (LEN) BYTE.
FBC6 9B 3D                  ADD A   BFRPTR+1   FORMS POINTER TO THE
FBC8 D9 3C                  ADC B   BFRPTR     TOP BYTE OF THE BUFFER.
FBCA 97 3F                  STA A   BFRSZE+1   SAVES THE HI BUFF. PTR.
FBCC D7 3E                  STA B   BFRSZE     TO THE CASSETTE BUFFER.
FBCE 8D AE                  BSR     CASIN      READS IN BLOCK NAME.
FBD0 36                     PSH A   -          SAVES FIRST CHAR OF NAME.
FBD1 8D AB                  BSR     CASIN      READ IN SECOND CHAR INTO A.
FBD3 D6 3A                  LDA B   NOPRNT     TESTS IF PRINT IS OFF.
FBD5 33                     PUL B   -          RESTORES FIRST NAME CHAR.
FBD6 27 06                  BEQ     RHDR1      SKIPS IF PRINT FLAG IS 0.
FBD8 F7 E03E                STA B   $E03E      DISPLAYS BLOCK NAME ON
FBDB B7 E03F                STA A   $E03F      THE CRT SCREEN.
FBDE 7D 0033      RHDR1     TST     BLKNAM     TESTS IF NAME IS CHECKED.
FBE1 27 08                  BEQ     RHDR2      SKIPS IF NO NAME CHECK.
FBE3 D1 33                  CMP B   BLKNAM     TESTS FIRST CHAR OF NAME.
FBE5 26 CE                  BNE     RDHDR      SKIPS BACK IF BAD NAME
FBE7 91 34                  CMP A   BLKNAM+1   TESTS SECOND NAME CHAR.
FBE9 26 CA                  BNE     RDHDR      SKIPS BACK IF BAD NAME.
FBEB 39           RHDR2     RTS                .




                  *         RDBFR READS DATA INTO THE MEMORY BUFFER FROM
                  *         THE CASSETTE.
                  *
                  *


FBEC 5F           RDBFR     CLR B   -          INIT B FOR CHECKSUM.
FBED DE 3C                  LDX     BFRPTR     LOADS START OF BUFFER.
FBEF DF 00        RBFR1     STX     TMP        .
FBF1 8D 8B                  BSR     CASIN      A GETS CHAR READ IN.
FBF3 DE 00                  LDX     TMP        X GETS BUFFER PTR.
FBF5 A7 00                  STA A   0,X        STORS CHAR INTO BUFFER.
FBF7 1B                     ABA                A GETS A+B.
FBF8 16                     TAB                B GETS A.
FBF9 9C 3E                  CPX     BFRSZE     TESTS IF BUFFER FULL.
FBFB 27 EE                  BEQ     RHDR2      SKIPS TO EXIT IF ALL IS IN.
FBFD 08                     INX                INC TO NEXT CHAR POSITIN.
FBFE 26 EF                  BNE     RBFR1      GOES BACK IF ANY LEFT.
                            END
```

SYS 2N FIXED

V3 N CASSETTE

FB

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	DE	38	86	13	A7	00	86	51	A7	00	39	86	16	8D	53	8D
1	51	8D	4F	86	1B	8D	4B	96	3E	D6	3F	D0	3D	92	3C	8D
2	41	17	8D	3E	96	33	8D	3A	96	34	8D	36	39	8D	48	C6
3	09	CE	51	61	09	26	FD	5A	26	F7	8D	CF	8D	10	86	17
4	8D	20	17	8D	1D	8D	1B	8D	19	8D	17	8D	63	39	5F	DE
5	3C	A6	00	DF	00	8D	0B	1B	16	DE	00	9C	3E	27	17	08
6	26	EF	36	DE	38	86	02	A5	00	27	FC	32	A7	01	7D	00
7	3A	27	03	B7	E0	1E	39	DE	38	86	B1	A7	00	39	DE	38
8	86	01	A5	00	27	FC	A6	01	7D	00	3A	27	03	B7	E0	1F
9	39	8D	E4	8D	20	8D	55	4F	97	09	8D	E2	81	17	27	04
A	86	54	20	07	8D	D8	11	27	07	86	43	97	09	B7	E0	5F
B	86	51	A7	00	39	8D	C7	81	16	26	FA	8D	C1	81	1B	26
C	F6	8D	BB	16	8D	B8	9B	3D	D9	3C	97	3F	D7	3E	8D	AE
D	36	8D	AB	D6	3A	33	27	06	F7	E0	3E	B7	E0	3F	7D	00
E	33	27	08	D1	33	26	CE	91	34	26	CA	39	5F	DE	3C	DF
F	00	8D	8B	DE	00	A7	00	1B	16	9C	3E	27	EE	08	26	EF

VERIFY AGAINST
CONTENT OF
EPROM FROM JOHN -
~~THEN MAKE COPY, MARKING IN~~
~~CONVERSION TO OCTAL~~
~~FOR~~ IF SAME,
COPY EPROM ONTO
FASTER ONE

PDS/2 (28.2)

TOTAL 5 EPROM $1\frac{1}{4}$K SET WHICH <u>INCLUDES</u> CALLS TO CASSETTE SUBROUTINES.

MINIASSEMBLER REMOVED FROM ROM. WRITTEN FOR RAM LOADING NOW.

SOME OTHER CHANGES MADE IN OPERATING SYSTEM.

PDS/2
(28.2)

NOTES ON PDS/2 (28.2)

SPHERE'S 1¼K OPERATING SYST WHICH HAD TYPE ROUTINES.
LATER THAN V3N

PDS/2, formerly known as SYS 2 Software, was written to provide keyboard control of cassettes. To provide room for these functions, the mini-assembler was removed form the PROMS. In removing the mini-assembler, fixing some potential bugs in the "SIM PROM" and cleaning/compacting other code there ASCII character input to the INPNUM routine, add the display of ASCII to the debugger, allow user specified "ESC" character, implementation of the real SWI vector and automatic check for the end of memory. It should be noted that this 5 PROM set has lost the boundary between the "SIM PROM" and the "CPU PROMS" and that all 5 PDS/2 PROMS are required to function properly.

OPERATION OF THE PDS/2

INITIALIZE CASSETTE: C/I n ESC where N is the last four bits of a number. If EVEN then FØ5x is used while if ODD then FØ6x is used. The exact ACIA address is determined by x=n added with E_X, ie n=FØ5Ø, n=1=FØ6Ø, n=2 =FØ52, n=F=FØ6E.

LOAD FROM CASSETTE: C/L bn ESC sa, C where bn=the block name of the first block entered with space specifies starting address follows---- anything else will be echoed but ignore and starting address will not be changed from what is currently at 3C/3D. Starting address ends with ESC, c=block count, specifies number of blocks to read. If Ø will substitute 1.

EXAMPLES: C/L 'SM ESC ƀ 4C00 ESC, ESC will load the basic loader.

C/L 'MA ESC ƀ 2ØØ ESC, ESC will load the Mini-Assembler.

C/L ESC , ESC will load the next block.

SAVE TO CASSETTE: C/S bn ESC where bn=the block name as specified for load above. The starting and ending addresses must already be set up at 3C/3D and 3E/3F.

GO FROM EXECUTIVE: C/G sa where sa=the execution address the same way as in the Load command: begins with the space, ends with ESC. If not specified, will execute last block loaded (3C/3D is default go address).

GO TO CANNED PROGRAM: C/A goes to the program pointed to by ØA/ØB.

ENTER ASCII TEXT TO INPNUM: Begin text with single quote (') the same way decimal (.) or OCTAL (*) are specified. ASCII text is terminated by internal carriage return (back single quote, $6Ø_X$) or by a value less than space ($2Ø_X$).

DEBUGGER: To prevent accidental exiting of the debugger only C/X when in the command made of the debugger will exit to the Executive. Also the period and comma have officially been added to OPEN PREVIOUS and OPEN NEXT respectivly (x equiv to - and , equiv to +) so shifting the keyboard is not necessary. Invalid inputs "close" the opened location.

USING THE MINI-ASSEMBLER:

C/I 1 ESC	Initialize cassette drive FØ6Ø
C/L 'MA ESC ɃØ 2ØØ⊥1 ESC	Read Mini-Assembler into memory if no errors, proceed, else repeat load.
C/G CR	Execute Mini-Assembler initialization: Change editor Low Buffer Bound from 2ØØ, set execution address for C/A, set cassette Buffer for read of edited source.
C/E ESC INITIALIZE EDITOR, MUST DO IF GOING TO READ IN SOURCE	Either read in pre-edited source such as:
C/L 'ES ESC CR ⊥ ESC	Re-edit if necessary with:
C/R RE-EDIT MANDITORY IF READ SOURCE FROM TAPE	or enter new source with editor:
C/E	when editing done and text is off top of CRT, exit editor:
ESC	and assemble:
C/A	the mini-Assembler displays the ending address and symbol table and sets up the CASBFR to the <u>LAST</u> origin statement or to the default origin of end of source + 1 and sets up CAS BFREND as end of object + 1 so assembled program can be saved:
C/S 'AB ESC	and/or executed directly:
C/G CR	

If a reset is performed with the mini-assembler already in memory, the initialization must be executed again: C/G ɃØ 2ØØ ESC to reset the editor low buffer boundry.

CHANGES IN PDSV3N MEMORY MAP

OUTEND	deleted
AEXEC	ØØØA-ØØØB --- C/A execution address

Changes in Memory cont......

ENDCHR	ØØ1Ø ---	Terminate editor character
OUTBUF	deleted	
LSTCHR	ØØ13 ---	Last keyboard character
BFRPTR	deleted	

Hardware interrupt vectored to Ø1ØØ
Software interrupt vectored to Ø1Ø4
Non-Maskable interrupt vectored to Ø1Ø8

ADD32	FD7B	INPNUM	FECE
ASCBIN	FF11	INTLZC	FBØØ
BINASC	FF64	LFTJST	FDA3
CASIN	FB77	MULT	FF92
CASOUT	FB62	OUTSTR	deleted
CLEAR	FCC7	PNTBYT	FEF3
CRLF	FDBA	PUTCHR	FD64
DEBUG	FE4B	RDBLK	FB8C
DIVIDE	FFAE	RDMOD	deleted
EDITIN	FDIA	REEDIT	FD1Ø
EDITOR	FDØ8	SUB32	FD73
EDITRD	FD18	TRNOFF	FBBØ
GETCHR	FCD4	TURNON	FB7Ø
HOME	FCBF	WRTBLK	FB2D
INPCHR	FE58	WRTMOD	deleted

Cassette output flag NOPRNT (3A): 1=Yes, Ø=No

Bit 7 = Cassette status
Bit 6 = Character I/O
Bit 5 = Blockname

```
                NAM     PDS-2
                OPT     NOPAGE
                OPT     O,NOG
                OPT     SYMBOL
*
* COPYRIGHT 1976 SPHERE CORPORATION
* 940 NORTH 4TH EAST, NORTH SALT LAKE, UTAH 84054
*
* SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
* DISTRIBUTION AND USE OF THE PDS SOFTWARE
* NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
* WRITTEN PERMISSION OF SPHERE CORP
*
* PDS-2 WAS DERRIVED FROM PDS-V3N TO ALLOW KEYBOARD
* LOADING OF CASSETTE TAPES AND IMPLEMENT OTHER FEATURES
* TO DATE PLANNED, BUT FOR WHICH THERE WASN'T ENOUGH ROOM
* THE MINI-ASSEMBLER HAS BEEN REMOVED AND PLACED ON
* CASSETTE TAPE TO ALLOW FOR THESE NEW ROUTINES
```

```
                * MEMORY MAP
      0000      TMP    EQU      $00
      0002      TMP1   EQU      $02
      0004      ARB    EQU      $04       PSEUDO REG B
      0004      AR3    EQU      $04       _HI BYTE
      0005      AR2    EQU      $05       _LO BYTE
      0006      ARA    EQU      $06       PSEUDO REG A
      0006      AR1    EQU      $06       _HI BYTE
      0007      AR0    EQU      $07       _LO BYTE
      0008      DIGIT  EQU      $08
      0009      CSTATS EQU      $09       CASS STAT
      000A      AEXEC  EQU      $0A       ASSM EXEC ADDR
      000C      BUFADR EQU      $0C       START OF I/O BFR
      000E      BUFEND EQU      $0E       END OF I/O BFR
      0010      ENDCHR EQU      $10       TERMINATE EDITOR CHAR
      0013      LSTCHR EQU      $13       LAST CHAR INPUT
      0014      SRCADR EQU      $14       MOVE SOURCE
      0016      DSTADR EQU      $16       MOVE DEST
      001A      ENDMEM EQU      $1A       END OF MAIN MEM
      001C      CSRPTR EQU      $1C       CURSER ADDR
      0020      BUFFLO EQU      $20       PTR TO END OF LOW EDIT TXT
      0022      BUFFHI EQU      $22       PTR TO START OF HI TXT
      0024      SCNPTR EQU      $24       PTR TO BFRD TXT START
      0026      SRCASM EQU      $26       PTR TO ASM SRC CODE
      0028      DSTASM EQU      $28       OBJ OUTPUT ADDR
      002A      ONDVAL EQU      $2A       ASM OPERAND VAL
      002C      SYMVAL EQU      $2C       SYMTBL VAL
      002E      BRKSAV EQU      $2E       SAVE FOR BKPT
      0030      BRKADR EQU      $30       ADDR OF BKPT
      0032      EDIT   EQU      $32       0 = EDIT OFF, NOT 0 = ON
      0033      BLKNAM EQU      $33       CASS BLK NAM
      0035      IOBUFF EQU      $35       NUMERIC SCRATCH
      0038      ACIANO EQU      $38       ACIA ADDR
      003A      NOPRNT EQU      $3A       NON-ZERO = DISP ON CRT
      003C      BFRPTR EQU      $3C       START CASS BFR
      003E      BFRSZE EQU      $3E       END CASS BFR
      0040      PCVAL  EQU      $40       P C
      E01F      FRSTLN EQU      $E01F   , RIGHT CHAR LINE 1
      E1E0      LASTLN EQU      $E1E0     LEFT SIDE BOTTOM OF CRT
      E1FF      LASTCH EQU      $E1FF     LAST CHAR ON CRT
      F040      KBDPIA EQU      $F040     ADDR OF PIA FOR KBD/2
                *KBDPIA EQU     $F000     ADDR OF PIA FOR KBD/1
      00B1      ON     EQU      $B1       ACIA ON VALUE
      0051      OFF    EQU      $51       ACIA OFF VALUE
                *
                * MINI-ASSEMBLER
                *
                * ON ENTRY:
                * SRCASM = ADDR OF SRC TXT TO BE ASSEMBLED
                * BUFFLO = DEFAULT ORIGIN
                *
                * ON EXIT:
                * BFRPTR = BUFFLO OR LAST ORIGIN
                * PCVAL, BFRSZE = LST LOC OF ASSMBLD OBJ
                *
0200                   ORG      $200
                *
0200 CE 0297    ASMINT LDX      #ASMXEQ   INIT EXEC ADDR
0203 DF 0A             STX      AEXEC
0205 CE 0356           LDX      #ASMEND   SET BFR ADDR'ES
0208 DF 0C             STX      BUFADR    EDIT
020A DF 3C             STX      BFRPTR    CASS
020C 7E FC9D           JMP      EXEC1     RTN TO EXEC
```

```
                     *
020F BD FCC5 ASMEXT JSR    HC
0212 CE 0040        LDX    #$40      SET 1ST CHAR
0215 86 20          LDA A  #'
0217 36      AL     PSH A  .         1ST COL
0218 8D 48          BSR    AS
021A E6 00          LDA B  0,X
021C A6 01          LDA A  1,X
021E 8D 4A          BSR    AA
0220 8D 67          BSR    A1
0222 32             PUL A
0223 36             PSH A  .         2ND COL
0224 8B 0D          ADD A  #13
0226 8D 3A          BSR    AS
0228 E6 1A          LDA B  26,X
022A A6 1B          LDA A  27,X
022C 8D 3C          BSR    AA
022E 8D 59          BSR    A1
0230 32             PUL A
0231 36             PSH A  .         3RD COL
0232 8B 1A          ADD A  #26
0234 8D 2C          BSR    AS
0236 E6 34          LDA B  52,X
0238 A6 35          LDA A  53,X
023A 8D 2E          BSR    AA
023C 8D 4B          BSR    A1
023E 32             PUL A
023F 36             PSH A  .         4TH COL
0240 8B 27          ADD A  #39
0242 8D 1E          BSR    AS
0244 E6 4E          LDA B  78,X
0246 A6 4F          LDA A  79,X
0248 8D 20          BSR    AA
024A 8D 3D          BSR    A1
024C 32             PUL A
024D 36             PSH A  .         5TH COL
024E 8B 34          ADD A  #52
0250 8D 10          BSR    AS
0252 E6 68          LDA B  104,X
0254 A6 69          LDA A  105,X
0256 8D 12          BSR    AA
0258 08             INX              BUMP INDEX TO SYMTAB
0259 08             INX
025A 86 60          LDA A  #$60      END LINE
025C 8D 2D          BSR    AP
025E 32             PUL A  .         BUMP CHAR
025F 4C             INC A
0260 20 B5          BRA    AL
                     *
0262 81 60   AS     CMP A  #$60      DONE?
0264 2D 25          BLT    AP        NO
0266 31      AE     INS
0267 31             INS
0268 31             INS
0269 39             RTS
                     *
026A C1 F0   AA     CMP B  #$F0
026C 26 04          BNE    AM
026E 81 F0          CMP A  #$F0
0270 27 13          BEQ    A4
0272 36      AM     PSH A
0273 17             TBA
0274 8D 01          BSR    AH
0276 32             PUL A
0277 FF 0295 AH     STX    AX
027A BD FFE3        JSR    PNTBYT
```

```
0027D FE 0295          LDX     AX
0280 39                RTS
                *
0281 20 8C      ADONE  BRA     ASMEXT
                *
0283 8D 00      A8     BSR     A4
0285 8D 00      A4     BSR     A2
0287 8D 00      A2     BSR     A1
0289 86 20      A1     LDA A   #'
028B FF 0295    AP     STX     AX
028E BD FD64           JSR     PUTCHR
0291 FE 0295           LDX     AX
0294 39                RTS
                *
0295 0002       AX     RMB     2
                *
0297 CE 0042    ASMXEQ LDX     #$42      CLEAR SYMBOL TABLE
029A 86 F0             LDA A   #$F0
029C A7 00      AI1    STA A   0,X
029E 08                INX
029F 8C 00C0           CPX     #$C0
02A2 26 F8             BNE     AI1
02A4 7F 0004           CLR     AR3       INIT CTR TO PASS 1
02A7 DE 20      ASM1   LDX     BUFFLO    SET
02A9 DF 3C             STX     BFRPTR    PTR FOR CASS.
02AB DF 40             STX     PCVAL     P C
02AD DE 26             LDX     SRCASM    LOAD ADR OF LINE 1
02AF DF 02      ASM1A  STX     TMP1      SAVE ADR OF CURRENT LINE
02B1 A6 08             LDA A   8,X       LOAD SYMBOL
02B3 E6 07             LDA B   7,X       LOAD OPERAND TYPE
02B5 C1 40             CMP B   #$'@      IF @, LOAD SYM VAL
02B7 27 72             BEQ     INDADR
02B9 08                INX
02BA 08                INX
02BB 08                INX
02BC 08                INX
02BD 08                INX
02BE 08                INX
02BF 08                INX
02C0 BD FF11           JSR     ASCBIN    CNVT OPERAND TO BINARY
02C3 D7 2A             STA B   ONDVAL
02C5 97 2B             STA A   ONDVAL+1
02C7 DE 02      SYMBL  LDX     TMP1      LOAD LINE PTR
02C9 A6 00             LDA A   0,X       LOAD SYMBOL
02CB E6 01             LDA B   1,X
02CD DE 40             LDX     PCVAL     SYMBOL = PC
02CF DF 2C             STX     SYMVAL
02D1 DF 3E             STX     BFRSZE    SET CASS END
02D3 C1 3D             CMP B   #$'=      LABEL EQUATED?
02D5 26 0C             BNE     ASM2      SKIP IF NO
02D7 DE 2A             LDX     ONDVAL    SYMBOL VALUE GETS
02D9 DF 2C             STX     SYMVAL    OPERAND VALUE
02DB 81 20             CMP A   #$'       TEST FOR ORIGIN
02DD 26 11             BNE     ASM3      SKIP IF NO
02DF DF 3C             STX     BFRPTR    SET CASS PTR
02E1 20 0D             BRA     ASM3
02E3 C1 20      ASM2   CMP B   #$'       TEST FOR END
02E5 27 09             BEQ     ASM3
02E7 7D 0004           TST     AR3
02EA 26 95             BNE     ADONE     EXIT IF PASS 2
02EC D7 04             STA B   AR3       SET CTR TO PASS 2
02EE 20 B7             BRA     ASM1
02F0 8D 41      ASM3   BSR     SYMPTR    GET SYM TBL ADR
02F2 96 2C             LDA A   SYMVAL    STORE SYMBOL ADDR
02F4 A7 00             STA A   0,X
        96 2D LDA A                                SYMVAL+1
```

```
0280 39            RTS
                *
0281 20 8C  ADONE  BRA     ASMEXT
                *
0283 8D 00  A8     BSR     A4
0285 8D 00  A4     BSR     A2
0287 8D 00  A2     BSR     A1
0289 86 20  A1     LDA A   #'
028B FF 0295 AP    STX     AX
028E BD FD64       JSR     PUTCHR
0291 FE 0295       LDX     AX
0294 39            RTS
                *
0295 0002   AX     RMB     2
                *
0297 CE 0042 ASMXEQ LDX    #$42      CLEAR SYMBOL TABLE
029A 86 F0         LDA A   #$F0
029C A7 00  AI1    STA A   0,X
029E 08            INX
029F 8C 00C0       CPX     #$C0
02A2 26 F8         BNE     AI1
02A4 7F 0004       CLR     AR3       *INIT CTR TO PASS 1
02A7 DE 20  ASM1   LDX     BUFFLO    SET
02A9 DF 3C         STX     BFRPTR    PTR FOR CASS.
02AB DF 40         STX     PCVAL     P C
02AD DE 26         LDX     SRCASM    LOAD ADR OF LINE 1
02AF DF 02  ASM1A  STX     TMP1      SAVE ADR OF CURRENT LINE
02B1 A6 08         LDA A   8,X       LOAD SYMBOL
02B3 E6 07         LDA B   7,X       LOAD OPERAND TYPE
02B5 C1 40         CMP B   #$'@      IF @, LOAD SYM VAL
02B7 27 72         BEQ     INDADR
02B9 08            INX
02BA 08            INX
02BB 08            INX
02BC 08            INX
02BD 08            INX
02BE 08            INX
02BF 08            INX
02C0 BD FF11       JSR     ASCBIN    CNVT OPERAND TO BINARY
02C3 D7 2A         STA B   ONDVAL
02C5 97 2B         STA A   ONDVAL+1
02C7 DE 02  SYMBL  LDX     TMP1      LOAD LINE PTR
02C9 A6 00         LDA A   0,X       LOAD SYMBOL
02CB E6 01         LDA B   1,X
02CD DE 40         LDX     PCVAL     SYMBOL = PC
02CF DF 2C         STX     SYMVAL
02D1 DF 3E         STX     BFRSZE    SET CASS END
02D3 C1 3D         CMP B   #$'=      LABEL EQUATED?
02D5 26 0C         BNE     ASM2      SKIP IF NO
02D7 DE 2A         LDX     ONDVAL    SYMBOL VALUE GETS
02D9 DF 2C         STX     SYMVAL    OPERAND VALUE
02DB 81 20         CMP A   #$'       TEST FOR ORIGIN
02DD 26 11         BNE     ASM3      SKIP IF NO
02DF DF 3C         STX     BFRPTR    SET CASS PTR
02E1 20 0D         BRA     ASM3
02E3 C1 20  ASM2   CMP B   #$'       TEST FOR END
02E5 27 09         BEQ     ASM3
02E7 7D 0004       TST     AR3
02EA 26 95         BNE     ADONE     EXIT IF PASS 2
02EC D7 04         STA B   AR3       SET CTR TO PASS 2
02EE 20 B7         BRA     ASM1
02F0 8D 41  ASM3   BSR     SYMPTR    GET SYM TBL ADR
02F2 96 2C         LDA A   SYMVAL    STORE SYMBOL ADDR
02F4 A7 00         STA A   0,X
```

```
02F8 A7 01           STA A  1,X
02FA DE 02           LDX    TMP1       LINE PTR
02FC 08              INX               POINT TO
02FD 08              INX               OP CODE
02FE 08              INX
02FF A6 00           LDA A  0,X        GET CHAR INTO A
0301 81 20           CMP A  #$C'       ,
0303 27 0A           BEQ    OPRND      SKIP IF NO OP CODE
0305 BD FF11         JSR    ASCBIN     CNVT OP CODE TO BINARY
0308 DE 40           LDX    PCVAL      LOAD PC
030A A7 00           STA A  0,X        PUT OP INTO OBJECT
030C 08              INX
030D DF 40           STX    PCVAL
030F DE 02    OPRND  LDX    TMP1       LOAD LINE PTR
0311 A6 06           LDA A  6,X        LOAD OPERAND SIZE CHAR
0313 DE 40           LDX    PCVAL      LOAD PC
0315 81 45           CMP A  #$'E       TEST LENGTH TYPE
0317 2E 31           BGT    RELTIV     SKIP IF "R"
0319 27 21           BEQ    EXTEND     SKIP IF "E"
031B 81 44           CMP A  #$'D       SKIP IF "D"
031D 27 22           BEQ    DIRECT     FALL THRU IF NO OPERAND
031F DE 02    ASM4   LDX    TMP1       LOAD LINE PTR
0321 08       ASM4A  INX
0322 A6 00           LDA A  0,X
0324 81 60           CMP A  #$60
0326 26 F9           BNE    ASM4A      SRCH UNTIL CR FOUND
0328 08              INX
0329 20 84           BRA    ASM1A      ASSM. NEXT LINE
              *
032B 8D 06    INDADR BSR    SYMPTR     GET ADDR OF SYMBOL
032D EE 00           LDX    0,X
032F DF 2A           STX    ONDVAL
0331 20 94           BRA    SYMBL
              *
0333 48       SYMPTR ASL A  .          MULT LABEL BY 2 TO FORM
0334 5F              CLR B  .          PTR INTO SYMBOL TABLE
0335 97 01           STA A  TMP+1
0337 D7 00           STA B  TMP
0339 DE 00           LDX    TMP
033B 39              RTS
033C D6 2A    EXTEND LDA B  ONDVAL     STORE HI BYTE OF OPRND
033E E7 00           STA B  0,X        INTO OBJ
0340 08              INX
0341 96 2B    DIRECT LDA A  ONDVAL+1 STORE LO BYTE OF OPRND
0343 A7 00           STA A  0,X        INTO OBJ
0345 08              INX
0346 DF 40           STX    PCVAL      ADVANCE PC
0348 20 D5           BRA    ASM4
              *
034A 08       RELTIV INX               INCR PC
034B DF 40           STX    PCVAL
034D 96 2B           LDA A  ONDVAL+1 LOAD LO BYTE OF OPRND
034F 90 41           SUB A  PCVAL+1  FORM RELATIVE OFFSET
0351 09              DEX
0352 A7 00           STA A  0,X        PUT INTO OBJ
0354 20 C9           BRA    ASM4
0356 00       ASMEND FCB    0
              *
FB00                 ORG    $FB00
              *
FB00 DE 38    INTLZ  LDX    ACIANO     GET ACIA ADDRESS
FB02 86 13           LDA A  #$13       RESET ACIA
FB04 A7 00           STA A  0,X
FB06 86 51           LDA A  #OFF       SET ACIA TO
FB08 A7 00           STA A  0,X        DIV BY 16,2 STOP BITS
FB0A 39              RTS
```

```
02F8 A7 01          STA A  1,X
02FA DE 02          LDX    TMP1      LINE PTR
02FC 08             INX              POINT TO
02FD 08             INX              OP CODE
02FE 08             INX
02FF A6 00          LDA A  0,X       GET CHAR INTO A
0301 81 20          CMP A  #$C'      ,
0303 27 0A          BEQ    OPRND     SKIP IF NO OP CODE
0305 BD FF11        JSR    ASCBIN    CNVT OP CODE TO BINARY
0308 DE 40          LDX    PCVAL     LOAD PC
030A A7 00          STA A  0,X       PUT OP INTO OBJECT
030C 08             INX
030D DF 40          STX    PCVAL
030F DE 02   OPRND  LDX    TMP1      LOAD LINE PTR
0311 A6 06          LDA A  6,X       LOAD OPERAND SIZE CHAR
0313 DE 40          LDX    PCVAL     LOAD PC
0315 81 45          CMP A  #$'E      TEST LENGTH TYPE
0317 2E 31          BGT    RELTIV    SKIP IF "R"
0319 27 21          BEQ    EXTEND    SKIP IF "E"
031B 81 44          CMP A  #$'D      SKIP IF "D"
031D 27 22          BEQ    DIRECT    FALL THRU IF NO OPERAND
031F DE 02   ASM4   LDX    TMP1      LOAD LINE PTR
0321 08      ASM4A  INX
0322 A6 00          LDA A  0,X
0324 81 60          CMP A  #$60
0326 26 F9          BNE    ASM4A     SRCH UNTIL CR FOUND
0328 08             INX
0329 20 84          BRA    ASM1A     ASSM. NEXT LINE
             *
032B 8D 06   INDADR BSR    SYMPTR    GET ADDR OF SYMBOL
032D EE 00          LDX    0,X
032F DF 2A          STX    ONDVAL
0331 20 94          BRA    SYMBL
             *
0333 48      SYMPTR ASL A  .         MULT LABEL BY 2 TO FORM
0334 5F             CLR B  .         PTR INTO SYMBOL TABLE
0335 97 01          STA A  TMP+1
0337 D7 00          STA B  TMP
0339 DE 00          LDX    TMP
033B 39             RTS
033C D6 2A   EXTEND LDA B  ONDVAL    STORE HI BYTE OF OPRND
033E E7 00          STA B  0,X       INTO OBJ
0340 08             INX
0341 96 2B   DIRECT LDA A  ONDVAL+1  STORE LO BYTE OF OPRND
0343 A7 00          STA A  0,X       INTO OBJ
0345 08             INX
0346 DF 40          STX    PCVAL     ADVANCE PC
0348 20 D5          BRA    ASM4
             *
034A 08      RELTIV INX              INCR PC
034B DF 40          STX    PCVAL
034D 96 2B          LDA A  ONDVAL+1  LOAD LO BYTE OF OPRND
034F 90 41          SUB A  PCVAL+1   FORM RELATIVE OFFSET
0351 09             DEX
0352 A7 00          STA A  0,X       PUT INTO OBJ
0354 20 C9          BRA    ASM4
0356 00      ASMEND FCB    0
             *
FB00                ORG    $FB00
             *
FB00 DE 38   INTLZ  LDX    ACIANO    GET ACIA ADDRESS
FB02 86 13          LDA A  #$13      RESET ACIA
FB04 A7 00          STA A  0,X
FB06 86 51          LDA A  #OFF      SET ACIA TO
FB08 A7 00          STA A  0,X       DIV BY 16,2 STOP BITS
FB0A 39             RTS
```

```
                     *
FB0B 86 16    WRTHDR LDA A  #$16      FMT TAPE HDR
FB0D 8D 53           BSR    CASOUT
FB0F 8D 51           BSR    CASOUT
FB11 8D 4F           BSR    CASOUT
FB13 86 1B           LDA A  #$1B
FB15 8D 4B           BSR    CASOUT
FB17 96 3E           LDA A  BFRSZE    OUTPUT BLK LEN
FB19 D6 3F           LDA B  BFRSZE+1
FB1B D0 3D           SUB B  BFRPTR+1
FB1D 92 3C           SBC A  BFRPTR
FB1F 8D 41           BSR    CASOUT
FB21 17              TBA
FB22 8D 3E           BSR    CASOUT
FB24 96 33           LDA A  BLKNAM    OUTPUT BLK NAME
FB26 8D 3A           BSR    CASOUT
FB28 96 34           LDA A  BLKNAM+1
FB2A 8D 36           BSR    CASOUT
FB2C 39              RTS
                     *
FB2D 8D 41    WRTBLK BSR    TURNON    TURNON CASS
FB2F C6 09           LDA B  #$9       TIME LOOP 1/4 SEC FOR
FB31 CE 5161  TIME1  LDX    #$5161    DRIVE TO GET TO SPEED
FB34 09       TIME2  DEX
FB35 26 FD           BNE    TIME2
FB37 5A              DEC B
FB38 26 F7           BNE    TIME1
FB3A 8D CF           BSR    WRTHDR    WRITE HDR
FB3C 8D 10           BSR    WRTBFR    WRITE DATA
FB3E 86 17           LDA A  #$17      OUTPUT TRAILER
FB40 8D 20           BSR    CASOUT
FB42 17              TBA
FB43 8D 1D           BSR    CASOUT
FB45 8D 1B           BSR    CASOUT
FB47 8D 19           BSR    CASOUT
FB49 8D 17           BSR    CASOUT
FB4B 8D 63           BSR    TRNOFF    STOP CASS
FB4D 39              RTS
                     *
FB4E 5F       WRTBFR CLR B            WRITE BFR TO TAPE, CLEAR
FB4F DE 3C           LDX    BFRPTR    CHECKSUM
FB51 A6 00    WBFR1  LDA A  0,X       LOAD CHAR
FB53 DF 00           STX    TMP       SAVE BFR PTR
FB55 8D 0B           BSR    CASOUT    OUTPUT CHAR
FB57 1B              ABA              COMPUTE CHECKSUM
FB58 16              TAB
FB59 DE 00           LDX    TMP
FB5B 9C 3E           CPX    BFRSZE    DONE?
FB5D 27 2C           BEQ    CINEXT    YES
FB5F 08              INX
FB60 20 EF           BRA    WBFR1
                     *
FB62 36       CASOUT PSH A            OUTPUT ACCUM A TO TAPE
FB63 DE 38           LDX    ACIANO
FB65 86 02           LDA A  #2
FB67 A5 00    CAS01  BIT A  0,X
FB69 27 FC           BEQ    CAS01     LOOP TIL OUTPUT BFR EMPTY
FB6B 32              PUL A
FB6C A7 01           STA A  1,X       OUTPUT DATA
FB6E 20 11           BRA    CSPRT
                     *
FB70 DE 38    TURNON LDX    ACIANO    TURN DRIVE ON
FB72 86 B1           LDA A  #ON
FB74 A7 00    TRNO   STA A  0,X
FB76 39              RTS
                     *
```

```
FB77 DE 38    CASIN  LDX    ACIANO    READ CHAR INTO ACCUM A
FB79 86 01           LDA A  #1
FB7B A5 00    CASIN1 BIT A  0,X
FB7D 27 FC           BEQ    CASIN1
FB7F A6 01           LDA A  1,X       PICK UP CHAR
FB81 36       CSPRT  PSH A
FB82 96 3A           LDA A  NOPRNT    OUTPUT IT TO CRT?
FB84 48              ASL A
FB85 32              PUL A
FB86 2A 03           BPL    CINEXT    IF BIT 6 = 1 YES
FB88 B7 E01F         STA A  $E01F
FB8B 39       CINEXT RTS
              *
FB8C 8D E2    RDBLK  BSR    TURNON    READ BLOCK, TURN ON DRIVE
FB8E 8D 24           BSR    RDHDR     READ HDR
FB90 8D 57           BSR    RDBFR     READ DATA
FB92 8D E3           BSR    CASIN     READ TRAILER
FB94 81 17           CMP A  #$17
FB96 27 04           BEQ    RTLR1
FB98 86 54           LDA A  #$'T      OUTPUT TRAILER ERROR
FB9A 20 07           BRA    RTLR2
FB9C 8D D9    RTLR1  BSR    CASIN     READ CHECKSUM
FB9E 10              SBA
FB9F 27 02           BEQ    RTLR2     TURNOFF DRIVE
FBA1 86 43           LDA A  #$'C      OUTPUT CHECKSUM ERROR
FBA3 97 09    RTLR2  STA A  CSTATS
FBA5 26 02           BNE    OCASER    IF ZERO OUTPUT SPACE
FBA7 86 20           LDA A  #$20
FBA9 D6 3A    OCASER LDA B  NOPRNT    DISPLAY ERROR?
FBAB 2A 03           BPL    TRNOFF    IF BIT 7 = 1 YES
FBAD B7 E05F         STA A  $E05F     DISPLAY ERROR
FBB0 86 51    TRNOFF LDA A  #OFF      TURN DRIVE OFF
FBB2 20 C0           BRA    TRNO
              *
FBB4 8D C1    RDHDR  BSR    CASIN     READ HDR, FIND THE
FBB6 81 16    RDHDR1 CMP A  #$16      START OF IT
FBB8 26 FA           BNE    RDHDR
FBBA 8D BB           BSR    CASIN     TEST FOR HDR CHAR
FBBC 81 1B           CMP A  #$1B
FBBE 26 F6           BNE    RDHDR1
FBC0 8D B5           BSR    CASIN     READ LENGTH
FBC2 16              TAB
FBC3 8D B2           BSR    CASIN
FBC5 9B 3D           ADD A  BFRPTR+1 COMPUTE END ADDR
FBC7 D9 3C           ADC B  BFRPTR
FBC9 97 3F           STA A  BFRSZE+1
FBCB D7 3E           STA B  BFRSZE
FBCD 8D A8           BSR    CASIN     READ BLK NAME
FBCF 97 00           STA A  TMP
FBD1 8D A4           BSR    CASIN
FBD3 97 01           STA A  TMP+1
FBD5 DE 00           LDX    TMP
FBD7 D6 3A           LDA B  NOPRNT    DISPLAY BLKNAM?
FBD9 58              ASL B
FBDA 58              ASL B
FBDB 2A 03           BPL    RHDR1     IF BIT 5 = 1 YES
FBDD FF E03E         STX    $E03E
FBE0 DE 33    RHDR1  LDX    BLKNAM    DOES NAME MATTER?
FBE2 27 A7           BEQ    CINEXT    NO
FBE4 9C 00           CPX    TMP       DOES NAME MATCH?
FBE6 26 CC           BNE    RDHDR     NO
FBE8 39              RTS              DONE (FINALLY)
              *
FBE9 5F       RDBFR  CLR B            READ BLK
FBEA DE 3C           LDX    BFRPTR
```

```
FBEE 8D 87              BSR    CASIN
FBF0 DE 00              LDX    TMP
FBF2 A7 00              STA A  0,X
FBF4 1B                 ABA              COMPUTE CHECKSUM
FBF5 16                 TAB
FBF6 9C 3E              CPX    BFRSZE    DONE?
FBF8 27 91              BEQ    CINEXT    YES
FBFA 08                 INX
FBFB 20 EF              BRA    RBFR1     GO FOR MORE
                  *
FBFD BD FECE INUM       JSR    INPNUM
FC00 39                 RTS
                  *
FC01 81 09      INIT    CMP A  #$09      CTRL/I
FC03 26 16              BNE    E2
FC05 8D F6              BSR    INUM      READ A NUMBER
FC07 84 0F              AND A  #$0F      MASK TO 1 DIGIT
FC09 C6 50              LDA B  #$50
FC0B 44                 LSR A
FC0C 24 02              BCC    IN1       IF INPUT WAS ODD,
FC0E C6 60              LDA B  #$60      ACIA = F060 + INPUT/2
FC10 48         IN1     ASL A            DROP ODD/EVEN BIT
FC11 1B                 ABA
FC12 97 39              STA A  ACIANO+1 MAKE UP ACIA ADDR
FC14 86 F0              LDA A  #$F0
FC16 97 38              STA A  ACIANO
FC18 BD FB00            JSR    INTLZ     INITIALIZE ACIA
FC1B 20 4C      E2      BRA    E1
FC1D 8D 39      CASBFR  BSR    CHRIN1    GET CHAR
FC1F 81 20              CMP A  #$'       SPACE?
FC21 26 06              BNE    CASEXT    RETURN IF NOT
FC23 8D D8              BSR    INUM      INPUT NUMBER
FC25 D7 3C              STA B  BFRPTR    STORE INTO
FC27 97 3D              STA A  BFRPTR+1 CASS I/O PTR
FC29 39         CASEXT  RTS
                  *
FC2A 81 0C      LOAD    CMP A  #$0C      CTRL/L
FC2C 26 D3              BNE    INIT
FC2E 8D CD              BSR    INUM
FC30 D7 33              STA B  BLKNAM    STORE CASS BLK NAME
FC32 97 34              STA A  BLKNAM+1
FC34 8D E7              BSR    CASBFR    INPUT STARTING ADDR
FC36 86 2C              LDA A  #$',
FC38 BD FD64            JSR    PUTCHR    OUTPUT SEPERATING COMMA
FC3B 8D C0              BSR    INUM      INPUT BLOCK COUNT
FC3D 16                 TAB
FC3E 26 01              BNE    LD3
FC40 5C                 INC B
FC41 37         LD3     PSH B
FC42 BD FB8C            JSR    RDBLK     INPUT A BLK
FC45 DE 3E              LDX    BFRSZE
FC47 DF 20              STX    BUFFLO    SET UP EDITOR END OF BLOCK
FC49 33                 PUL B
FC4A 5A                 DEC B
FC4B 27 50              BEQ    EXEC1     IF ALL BLKS READ, DONE
FC4D DE 33              LDX    BLKNAM    ADVANCE BLK NAME
FC4F 08                 INX
FC50 DF 33              STX    BLKNAM
FC52 DE 3E              LDX    BFRSZE    NXT BLK FOLLOWS LAST BLK
FC54 DF 3C              STX    BFRPTR
FC56 20 E9              BRA    LD3
                  *
FC58 BD FE58 CHRIN1     JSR    INPCHR    GET & DISPLAY A CHAR
FC5B 39                 RTS
                  *
FC5C 81 13      SAVE    CMP A  #$13      CTRL/S
```

```
FC5E 26 CA              BNE     LOAD
FC60 8D 9B              BSR     INUM      GET BLKNAM
FC62 D7 33              STA B   BLKNAM
FC64 97 34              STA A   BLKNAM+1
FC66 BD FB2D            JSR     WRTBLK    WRITE BLK
FC69 20 32      E1      BRA     EXEC1
                *
FC6B 81 07      GO      CMP A   #$07      CNTL/G
FC6D 26 ED              BNE     SAVE
FC6F 8D AC              BSR     CASBFR    GET ADDR TO GO TO
FC71 DE 3C              LDX     BFRPTR
FC73 6E 00              JMP     0,X       GO
                *
FC75 86 1B      RESTRT  LDA A   #$1B      SETS TERMINATION
FC77 97 10              STA A   ENDCHR    CHAR TO "ESC" CHAR
FC79 86 7E              LDA A   #$7E      INIT SWI
FC7B B7 0104            STA A   $104
FC7E CE FE33            LDX     #BKENTR
FC81 FF 0105            STX     $105
FC84 8E 01FF            LDS     #$1FF     SET STK PTR
FC87 30                 TSX
FC88 DF 0C              STX     BUFADR    SET UP INPUT ADDR
FC8A CE E000            LDX     #$E000    FIND THE END OF MEMORY
FC8D 09         R1      DEX
FC8E A6 00              LDA A   0,X       PICK UP DATA
FC90 63 00              COM     0,X       COMPLEMENT IT,
FC92 A8 00              EOR A   0,X       DID IT WORK?
FC94 4C                 INC A
FC95 26 F6              BNE     R1        NO, DO NEXT ONE
FC97 DF 0E              STX     BUFEND    SAVE END OF BUFFER ADDR
FC99 DF 1A              STX     ENDMEM    SAVE END OF MEMORY ADDR
FC9B 8D 28              BSR     HC        HOME AND CLEAR
FC9D DE 1C      EXEC1   LDX     CSRPTR
FC9F BD FDBA            JSR     CRLF
FCA2 8D B4              BSR     CHRIN1    GET
FCA4 81 01              CMP A   #$01      CTRL/A
FCA6 26 04              BNE     EXEC2
FCA8 DE 0A              LDX     AEXEC
FCAA AD 00              JSR     0,X
FCAC 81 05      EXEC2   CMP A   #$05      CTRL/E
FCAE 26 02              BNE     EXEC3
FCB0 8D 56              BSR     EDITOR
FCB2 81 12      EXEC3   CMP A   #$12      CTRL/R
FCB4 26 02              BNE     EXEC4
FCB6 8D 58              BSR     REEDIT
FCB8 81 04      EXEC4   CMP A   #$04      CTRL/D
FCBA 26 AF              BNE     GO
FCBC 7E FE4B            JMP     DEBUG
                *
FCBF CE E000    HOME    LDX     #$E000    SET CSR TO TOP LFT OF CRT
FCC2 DF 1C              STX     CSRPTR
FCC4 39                 RTS
                *
FCC5 8D F8      HC      BSR     HOME
FCC7 C6 60      CLEAR   LDA B   #$60      LOAD BLANK (C.R.)
FCC9 CE E200            LDX     #LASTCH+1
FCCC 09         CLEAR1  DEX
FCCD E7 00              STA B   0,X
FCCF 9C 1C              CPX     CSRPTR
FCD1 26 F9              BNE     CLEAR1
FCD3 39                 RTS
                *
FCD4 86 1F      GETCHR  LDA A   #$1F      INIT KBD
FCD6 B7 F041            STA A   KBDPIA+1
FCD9 37                 PSH B
```

```
FCDC E6 00            LDA B  0,X      SAVE CHAR ON CRT
FCDE DE 1C    GET0    LDX    CSRPTR
FCE0 63 00            COM    0,X      COMPLIMENT
FCE2 CE 26F0          LDX    #9968
FCE5 09       GET1    DEX
FCE6 27 F6            BEQ    GET0
FCE8 8C 15B3          CPX    #5555    CHECK FOR REPEAT
FCEB 26 09            BNE    GET2
FCED B6 F040          LDA A  KBDPIA   TIMING DONE, READ DATA
FCF0 2A 04            BPL    GET2
FCF2 96 13            LDA A  LSTCHR   BIT 7 ON, PICKUP LAST INPUT
FCF4 20 0A            BRA    GET3
FCF6 86 40    GET2    LDA A  #$40     KBD STROBE = CA2
FCF8 B5 F041          BIT A  KBDPIA+1
FCFB 27 E8            BEQ    GET1     BRANCH IF CHAR NOT ENTERED
FCFD B6 F040          LDA A  KBDPIA   LOAD CHAR
FD00 DE 1C    GET3    LDX    CSRPTR
FD02 E7 00            STA B  0,X      RESTORE CHAR TO CRT
FD04 33               PUL B
FD05 97 13            STA A  LSTCHR   SAVE CHAR INPUT
FD07 39               RTS

FD08 DE 0C    EDITOR  LDX    BUFADR   SET UP BFR PTRS
FD0A DF 20            STX    BUFFLO
FD0C DE 0E            LDX    BUFEND
FD0E DF 22            STX    BUFFHI
FD10 8D B3    REEDIT  BSR    HC
FD12 DE 0C            LDX    BUFADR
FD14 DF 26            STX    SRCASM   SET ASSM SRC PTR
FD16 DF 3C            STX    BFRPTR   SET CASS I/O PTR
FD18 97 32    EDITRD  STA A  EDIT     TURN ON EDIT MODE
FD1A DE 1C    EDITIN  LDX    CSRPTR   SET SCAN PTR
FD1C DF 24            STX    SCNPTR
              *
FD1E 8D B4    EDREAD  BSR    GETCHR   GET CHAR
FD20 91 10            CMP A  ENDCHR
FD22 26 04            BNE    ED1      SKIP IF NOT END CHAR
FD24 7F 0032          CLR    EDIT     TURN OFF EDIT FLAG
FD27 39               RTS
FD28 8D 0A    ED1     BSR    INSERT   EDIT CHAR'S
FD2A 20 F2            BRA    EDREAD
              *
FD2C 81 0D    CR      CMP A  #$0D     TEST FOR CR
FD2E 2D 8F            BLT    HOME     SKIP IF HOME CMD
FD30 2E 12            BGT    RTCSR
FD32 86 60            LDA A  #$60     LOAD INTERNAL CR
              *
FD34 81 09    INSERT  CMP A  #$09     CTRL/I
FD36 2D 28            BLT    DELETE   SKIP TO DELETE
FD38 2E F2            BGT    CR
FD3A D6 32    INSRT1  LDA B  EDIT
FD3C 27 03            BEQ    INSRT2   EXIT IF EDITOR OFF
FD3E BD FDEC          JSR    MOVE2    MOVE LAST LINE TO BUFFHI
FD41 7E FE19  INSRT2  JMP    SCRLDN   MOVE ALL LINES DOWN 1
              *
FD44 81 12    RTCSR   CMP A  #$12     TEST FOR RIGHT ARROW
FD46 2D 2B            BLT    SUB32    SKIP IF "UP ARROW"
FD48 2E 0D            BGT    LFTCSR
FD4A DE 1C    RTARRO  LDX    CSRPTR
FD4C 20 22            BRA    PUTCH1   INCR CSR
              *
FD4E 81 1F    CLER    CMP A  #$1F     TEST FOR CTRL/DELETE
FD50 27 51            BEQ    LFTJST
FD52 2E 10            BGT    PUTCHR   IF NOT DELETE, ECHO IT
```

```
                           *
FD57 81 14      LFTCSR CMP A   #$14       TEST FOR "LEFT ARROW"
FD59 2D 20             BLT     ADD32      SKIP IF "DOWN ARROW"
FD5B 2E F1             BGT     CLER
FD5D 09                DEX                DECR CSR
FD5E 20 21             BRA     ADD2
                *
FD60 8D 56      DELETE BSR     OVR1A      SCROLL UP
FD62 20 2F             BRA     OVR3
                *
FD64 DE 1C      PUTCHR LDX     CSRPTR
FD66 81 0D             CMP A   #$0D       TEST FOR EXTERNAL CR
FD68 27 50             BEQ     CRLF
FD6A A7 00             STA A   0,X        DISPLAY CHAR
FD6C 81 60             CMP A   #$60       TEST FOR INTERNAL CR
FD6E 27 4A             BEQ     CRLF
FD70 08         PUTCH1 INX                INCR CSRPTR
FD71 20 0E             BRA     ADD2       TEST FOR OVRFLO
                *
FD73 DE 1C      SUB32  LDX     CSRPTR
FD75 C6 40             LDA B   #64
FD77 09         SUB32A DEX                CSRPTR = CSRPTR - 64
FD78 5A                DEC B
FD79 26 FC             BNE     SUB32A
                *
FD7B C6 20      ADD32  LDA B   #32
FD7D 08         ADD32A INX                CSRPTR = CSRPTR + 32
FD7E 5A                DEC B
FD7F 26 FC             BNE     ADD32A
FD81 DF 1C      ADD2   STX     CSRPTR
                *
                * NDRFLO (UNDERFLOW) CHECK CURSOR OFF TOP OF CRT
FD83 8C E000           CPX     #$E000
FD86 2C 04             BGE     OVRFLO     SKIPS IF CSRPTR >= E000
FD88 8D B0             BSR     INSRT1     SCROLL DOWN 1 LINE
FD8A 8D 32             BSR     MOVE3
                *
                * OVERFLOW CHECK FOR CURSOR OFF BOTTOM OF CRT
FD8C 8C E200    OVRFLO CPX     #$E200
FD8F 2B 18             BMI     OVREXT     EXIT IF CSRPTR < E200
FD91 8D 17             BSR     OVR1       SCROLL UP
FD93 D6 32      OVR3   LDA B   EDIT
FD95 27 12             BEQ     OVREXT     EXIT IF EDIT IS OFF
FD97 DE 22             LDX     BUFFHI
FD99 9C 0E             CPX     BUFEND     IF CHARS IN BFR
FD9B 27 0C             BEQ     OVREXT
FD9D 8D 3C             BSR     MOVE1A     MOVE TO LAST LINE
FD9F DE 14             LDX     SRCADR     SET NEW BUFFHI LOCATION
FDA1 DF 22             STX     BUFFHI
                *
FDA3 D6 1D      LFTJST LDA B   CSRPTR+1 LOAD LOW BYTE
FDA5 C4 E0             AND B   #$E0       TRUNCATE TO LEFT OF LINE
FDA7 D7 1D             STA B   CSRPTR+1
FDA9 39         OVREXT RTS
                *
FDAA D6 32      OVR1   LDA B   EDIT
FDAC 27 0A             BEQ     OVR1A      SKIP IF EDIT OFF
FDAE DE 20             LDX     BUFFLO     SETUP
FDB0 DF 16             STX     DSTADR     DEST OF MOVE
FDB2 DE 24             LDX     SCNPTR     SRC FOR MOVE
FDB4 8D 26             BSR     MOVE1      MOVE 1 LINE TO BUFFLO
FDB6 DF 20             STX     BUFFLO     SAVE NEW BUFFLO
FDB8 20 4B      OVR1A  BRA     SCRLUP
                *
FDBA 8D BF      CRLF   BSR     ADD32      LF
```

```
                 *
                 * MOVE3 CALCULATES THE SOURCE ADDRESS OF THE DATA IN
                 * BUFFLO (IF IT EXISTS) FOR MOVING TO THE FIRST LINE ON
                 * THE CRT. MOVE 1 IS THEN ENTERED TO DO THE MOVING
                 *
FDBE DE 24       MOVE3  LDX    SCNPTR
FDC0 DF 16              STX    DSTADR   SET MOVE ADDR
FDC2 DE 20              LDX    BUFFLO   LOAD LO BFR ADDR
FDC4 9C 0C              CPX    BUFADR
FDC6 27 23              BEQ    MOVEXT   EXIT IF BFR EMPTY
FDC8 09                 DEX             MOVE BACK FROM BLANK
FDC9 9C 0C       MV31   CPX    BUFADR   TEST IF SRCADR = BUFFADR
FDCB 27 08              BEQ    MV32     MOVE IF START OF LINE
FDCD 09                 DEX             NEXT LOWER CHAR
FDCE E6 00              LDA B  0,X      GET SOURCE CHAR FOR TEST
FDD0 C1 60              CMP B  #$60     TEST FOR CR
FDD2 26 F5              BNE    MV31     SKIP BACK UNTIL CR
FDD4 08                 INX             POINT TO FIRST CHAR
FDD5 DF 20       MV32   STX    BUFFLO   SAVE LO ADDRESS
FDD7 20 03              BRA    MOVE1
                 *
                 * SRC ADDR -1 = SRCADR, DEST ADDR = DSTADR,
                 * MOVE TERMINATED BY CR
                 *
FDD9 DE 14       MOVE   LDX    SRCADR
FDDB 08          MOVE1A INX
FDDC E6 00       MOVE1  LDA B  0,X      LOAD CHAR
FDDE DF 14              STX    SRCADR
FDE0 DE 16              LDX    DSTADR
FDE2 E7 00              STA B  0,X      STORE CHAR
FDE4 08                 INX
FDE5 DF 16              STX    DSTADR
FDE7 C1 60              CMP B  #$60
FDE9 26 EE              BNE    MOVE     GO BACK IF NOT DONE
FDEB 39          MOVEXT RTS
                 *
                 * MOVE LAST LINE ON SCREEN TO HIGH AREA OF THE BFR
                 * (BUFFHI) DURING SCROLLING, MOVE TERMINATED BY CR
                 *
FDEC CE E1E0     MOVE2  LDX    #LASTLN  GET ADDR OF LAST LINE
FDEF 5F                 CLR B  -        SET TERM FOR POPPING
FDF0 37          MV21   PSH B
FDF1 E6 00              LDA B  0,X
FDF3 08                 INX
FDF4 C1 60              CMP B  #$60
FDF6 26 F8              BNE    MV21     STORE IF NOT TERM'TING CR
FDF8 DE 22              LDX    BUFFHI   INIT DEST
FDFA E7 00       MV23   STA B  0,X      STORE CHAR
FDFC 09                 DEX
FDFD DF 22              STX    BUFFHI
FDFF 33                 PUL B
FE00 C1 00              CMP B  #00
FE02 26 F6              BNE    MV23     GO BACK IF NOT DONE
FE04 39                 RTS
                 *
                 * SCROLLUP - ALL LINES UP 1, & CLEAR LAST LINE
FE05 CE E000     SCRLUP LDX    #$E000
FE08 E6 20       SCRP1  LDA B  $20,X    GET CHAR FROM NEXT LINE
FE0A E7 00              STA B  00,X     STORE CHAR ON THIS LINE
FE0C 08                 INX
FE0D 8C E1E0            CPX    #LASTLN
FE10 26 F6              BNE    SCRP1    GO BACK IF MOVE NOT DONE
FE12 DF 1C              STX    CSRPTR   SET CSRPTR
FE14 DF 16              STX    DSTADR   INIT DEST FOR NEXT MOVE
FE16 7E FCC7            JMP    CLEAR    CLEAR LAST LINE
```

```
                      *
FE19 CE E1DF SCRLDN LDX     #LASTLN-1
FE1C E6 00   SCRD1  LDA B   0,X        GET CHAR FROM THIS LINE
FE1E E7 20          STA B   $20,X      STORE CHAR ON NEXT LINE
FE20 DF 1C          STX     CSRPTR     SAVE CURSOR
FE22 09             DEX
FE23 8C DFFF        CPX     #$DFFF
FE26 26 F4          BNE     SCRD1      GO BACK IF MOVE NOT DONE
FE28 C6 60          LDA B   #$60       LOAD BLANK TO CLEAR LINE
FE2A 08      SCRD2  INX
FE2B E7 00          STA B   0,X        CLEAR LINE 1
FE2D 8C E01F        CPX     #FRSTLN
FE30 26 F8          BNE     SCRD2
FE32 39             RTS
                    *
                    * DEBUGGER
                    *
                    * ENTRY OF THE BRKPT VECTOR
FE33 30      BKENTR TSX                X GETS SP
FE34 A6 06          LDA A   6,X
FE36 80 01          SUB A   #1         SUB 1 FROM RTN ADDR
FE38 A7 06          STA A   6,X
FE3A 24 0F          BCC     DEBUG
FE3C 6A 05          DEC     5,X
FE3E 20 0B          BRA     DEBUG
                    *
FE40 81 0D   LINE   CMP A   #$0D       TEST FOR CR
FE42 2D 1F          BLT     JMPLCN     GO JSR
FE44 2E 55          BGT     OPNREG
FE46 BD FD73        JSR     SUB32      MOVE UP 1 LINE
FE49 31      POPLIN INS                CLEAN UP STACK
FE4A 31             INS
                    *
FE4B 8D 79   DEBUG  BSR     NEWLIN     CR>
FE4D 8D 09   DBUG1  BSR     INPCHR
FE4F BD FD4A        JSR     RTARRO     OUTPUT SPACE
FE52 DE 40          LDX     PCVAL      LOAD OPENED LOC ADDR
FE54 8D 0F          BSR     RUNBUG     EXECUTE COMMAND
FE56 20 F5          BRA     DBUG1
                    *
FE58 BD FCD4 INPCHR JSR     GETCHR
FE5B 7E FD64 OUTCHR JMP     PUTCHR
                    *
FE5E CE FC9D EXECTV LDX     #EXEC1
FE61 31      GOLOCN INS                CLEAN UP STACK
FE62 31             INS
FE63 6E 00   JMPLCN JMP     0,X        JUMP TO PROGRAM
                    *
FE65 81 03   RUNBUG CMP A   #$03
FE67 2D 26          BLT     BRKSET
FE69 2E 46          BGT     EXIT
FE6B DE 30          LDX     BRKADR     CTRL/C CLEAR BKPT
FE6D 96 2E          LDA A   BRKSAV
FE6F A7 00          STA A   0,X        RESTORE DATA
FE71 20 62          BRA     DSPADR
FE73 8D 59   CHANGE BSR     INPNUM     INPUT NEW CONTENTS
FE75 DE 40          LDX     PCVAL
FE77 A7 00          STA A   0,X        STORE NEW CONTENTS.
FE79 08      OPNNXT INX                INCR OPENED LOC ADDR
FE7A 20 59          BRA     DSPADR     GO DISPLAY OPEN BYTE
FE7C 81 18   OPNPRE CMP A   #$18
FE7E 27 DE          BEQ     EXECTV     RETURN TO EXEC IF CTRL/X
FE80 81 2B          CMP A   #$'+
FE82 2D EF          BLT     CHANGE     IF < '+' CHANGE CONTENTS
FE84 81 2D          CMP A   #$'-
```

```
FE88 81 2E          CMP A  #'.
FE8A 2E BD          BGT    POPLIN   IGNORE IF NOT . OR -
FE8C 09             DEX             DECR OPENED LOC IF . OR -
FE8D 20 46          BRA    DSPADR   GO DISPLAY OPEN BYTE
FE8F A6 00   BRKSET LDA A  0,X
FE91 97 2E          STA A  BRKSAV   SAVE DATA OF OPNED BYTE
FE93 DF 30          STX    BRKADR   SAVE ADDR OF BKPT
FE95 86 3F          LDA A  #$3F
FE97 A7 00          STA A  0,X      SET SWI AT OPEN BYTE
FE99 20 3A          BRA    DSPADR
FE9B 81 12   OPNREG CMP A  #$12
FE9D 2D 0A          BLT    OPNLOC
FE9F 2E 19          BGT    OPNTBL
FEA1 30             TSX             OPEN TOP-OF-STACK
FEA2 08             INX
FEA3 08             INX
FEA4 20 2F          BRA    DSPADR   DISPLAY T-O-S
FEA6 35      SETSTK TXS             STACK POINTER GETS PCVAL
FEA7 20 A2          BRA    DEBUG
FEA9 8D 23   OPNLOC BSR    INPNUM   GET NEW OPEN LOC ADDR
FEAB D7 40   OPNLC1 STA B  PCVAL
FEAD 97 41          STA A  PCVAL+1
FEAF 20 26          BRA    DSPAD1
FEB1 81 07   EXIT   CMP A  #$07
FEB3 27 AC          BEQ    GOLOCN
FEB5 2E 89          BGT    LINE
FEB7 31             INS             CLEAN UP STACK
FEB8 31             INS
FEB9 3B             RTI             RETURN FROM BREAKPOINT
FEBA 81 14   OPNTBL CMP A  #$14
FEBC 2D E8          BLT    SETSTK
FEBE 2E BC          BGT    OPNPRE
FEC0 8D 96          BSR    INPCHR   LOADS A WITH TAG
FEC2 48             ASL A           ADJUST ADDR
FEC3 5F             CLR B
FEC4 20 E5          BRA    OPNLC1   SAVE & DISPLAY
FEC6 86 0D   NEWLIN LDA A  #$0D
FEC8 8D 44          BSR    PNTBF1   PRINT CR
FECA 86 3E          LDA A  #'>
FECC 20 40          BRA    PNTBF1   PRINT PROMPT CHAR
             *
FECE BD FD1A INPNUM JSR    EDITIN   INPUT A STRING
FED1 DE 24          LDX    SCNPTR   LOAD ADDR OF 134 DIGIT
FED3 20 3C          BRA    ASCBIN   CONVERT TO BINARY IN BA
             *
FED5 DF 40   DSPADR STX    PCVAL    SAVE OPEN LOC ADDR
FED7 8D ED   DSPAD1 BSR    NEWLIN   PRINT CR >
FED9 8D 12          BSR    PNTDIG   DISPLAY PCVAL
FEDB BD FD4A        JSR    RTARRO
FEDE DE 40          LDX    PCVAL
FEE0 A6 00          LDA A  0,X      LOAD DATA FROM OPEN LOC
FEE2 36             PSH A
FEE3 8D 0E          BSR    PNTBYT   DISPLAY DATA IN HEX
FEE5 32             PUL A
FEE6 DE 1C          LDX    CSRPTR   DISPLAY ASCII
FEE8 A7 00          STA A  0,X      WITHOUT BEING MESSED UP
FEEA 7E FD70        JMP    PUTCH1   BY OUTPUT EDITING
             *
FEED 96 40   PNTDIG LDA A  PCVAL    DISPLAY LOC
FEEF 8D 02          BSR    PNTBYT
FEF1 96 41          LDA A  PCVAL+1
FEF3 CE 0010 PNTBYT LDX    #16      LOAD BASE
FEF6 DF 04          STX    ARB
FEF8 5F             CLR B  -        CLEAR HI DIGITS
FEF9 CE 0035        LDX    #IOBUFF  LOAD OUTPUT BUFF ADDR
FEFC D7 36          STA B  IOBUFF+1 CLEAR 2 $ DIGIT
```

```
FEFE 8D 64          BSR    BINASC    CNVT TO ASCII
FF00 96 35          LDA A  IOBUFF    LOAD HI DIGIT
FF02 D6 36          LDA B  IOBUFF+1
FF04 26 04          BNE    PNTBUF    SKIPS IF BOTH DIGITS CONVTD
FF06 97 36          STA A  IOBUFF+1  SETS UP LOW DIGIT
FF08 86 30          LDA A  #$'0      HIGH DIGIT GETS A "0"
                    *
FF0A 8D 02   PNTBUF BSR    PNTBF1    PRINT HI DIGIT
FF0C 96 36          LDA A  IOBUFF+1  LOAD LOW DIGIT
FF0E 7E FD64 PNTBF1 JMP    PUTCHR    DISPLAY CHAR
                    *
                    * UTILITY PROGRAMS
FF11 A6 00   ASCBIN LDA A  0,X       GET CHR TO FORM BASE
FF13 81 2E          CMP A  #$'.      TEST BASE
FF15 2D 06          BLT    OCT       SKIP IF 8 (*)
FF17 2E 1C          BGT    HEX       SKIP IF 16
FF19 C6 0A          LDA B  #10       LOAD 10
FF1B 20 02          BRA    ASC1
FF1D C6 08   OCT    LDA B  #8        LOAD 8
FF1F 08      ASC1   INX              INCR PTR TO NEXT CHAR
FF20 81 27          CMP A  #$''      CHECK FOR ASCII STRING
FF22 26 13          BNE    ASC2      NOT ASCII, SAVE BASE
FF24 5F             CLR B
FF25 37             PSH B
FF26 A6 00   ASCASC LDA A  0,X       GET CHAR
FF28 08             INX
FF29 81 20          CMP A  #$20      DONE?
FF2B 2D 35          BLT    AEXIT
FF2D 81 60          CMP A  #$60
FF2F 27 31          BEQ    AEXIT     YES
FF31 33             PUL B  .         MOVE LEFT CHAR TO B
FF32 36             PSH A  .         PUT RIGHT CHR ON STACK
FF33 20 F1          BRA    ASCASC
FF35 C6 10   HEX    LDA B  #16       LOAD 16
FF37 D7 07   ASC2   STA B  AR0       SAVE BASE
FF39 5F             CLR B
FF3A 37             PSH B  .         ZERO LOW HALF OF NUM
FF3B D7 06          STA B  AR1
FF3D A6 00   NXTCHR LDA A  0,X       GET CHAR TO CNVT
FF3F 08             INX
FF40 80 30          SUB A  #$'0
FF42 2B 1E          BMI    AEXIT     EXIT IF LESS THAN ZERO
FF44 81 0A          CMP A  #10
FF46 2D 06          BLT    ASC3
FF48 80 07          SUB A  #7
FF4A 81 0A          CMP A  #10
FF4C 2D 14          BLT    AEXIT
FF4E 91 07   ASC3   CMP A  AR0       EXIT IF > BASE
FF50 2C 10          BGE    AEXIT
FF52 97 08          STA A  DIGIT     SAVE DIGIT
FF54 DF 00          STX    TMP       SAVE INDEX REG
FF56 32             PUL A  -         RESTORE LO OF NUM
FF57 8D 39          BSR    MULT      NUM = NUM * BASE + DIGIT
FF59 9B 08          ADD A  DIGIT     NUMBER GETS NUMBER + DIGIT
FF5B C9 00          ADC B  #0
FF5D 36             PSH A  -         SAVE LO OF NUMBER
FF5E DE 00          LDX    TMP       RESTORE STRING PTR
FF60 20 DB          BRA    NXTCHR    CNVT NEXT CHAR
FF62 32      AEXIT  PUL A  -         RESTORE NUMBER IN BA
FF63 39             RTS
                    *
FF64 DF 00   BINASC STX    TMP       SAVE OUTPUT PTR
FF66 34             DES              SP = SP - 1
FF67 30             TSX
FF68 6F 00          CLR    0,X
```

```
FF6C DE 04   BIN1   LDX    ARB      RESTORE DIVISOR (BASE)
FF6E DF 06          STX    ARA
FF70 8D 3C          BSR    DIVIDE
FF72 97 02          STA A  TMP1     SAVE A
FF74 96 07          LDA A  AR0      LOAD DIGIT (REMAINDER)
FF76 36             PSH A  -        STACK DIGIT (REVERSE ORDER)
FF77 96 02          LDA A  TMP1     RESTORE A
FF79 4D             TST A  -        IF QUOTIENT = 0, DONE
FF7A 26 F0          BNE    BIN1
FF7C 5D             TST B
FF7D 26 ED          BNE    BIN1
FF7F DE 00          LDX    TMP      RESTORE OUTPUT PTR
FF81 32      BIN3   PUL A  -        UNSTACK A DIGIT
FF82 4D             TST A
FF83 2B 28          BMI    ARTS     EXIT IF NEGATIVE
FF85 81 09          CMP A  #9       TEST IF RESULT IS HEX
FF87 2F 02          BLE    BIN5     SKIP IF DIGIT NOT HEX
FF89 8B 07          ADD A  #7       FORM HEX VALUE
FF8B 8B 30   BIN5   ADD A  #$'0     FORM DECIMAL CHAR
FF8D A7 00          STA A  0,X      OUTPUT CHAR
FF8F 08             INX
FF90 20 EF          BRA    BIN3     GO FOR NEXT DIGIT
             *
FF92 36      MULT   PSH A  -        STACK BA
FF93 37             PSH B
FF94 86 10          LDA A  #16      LOAD COUNT ONTO STACK
FF96 36             PSH A
FF97 4F             CLR A  -        ZERO BA
FF98 5F             CLR B
FF99 30             TSX
FF9A 48      MUL1   ASL A  -        SHIFT LEFT BA
FF9B 59             ROL B
FF9C 68 02          ASL    2,X      SHIFTS ORIG BA OPERAND
FF9E 69 01          ROL    1,X      ONE LEFT INTO CARRY
FFA0 24 04          BCC    MUL2     SKIPS ADDING IF CARRY = 0
FFA2 9B 07          ADD A  AR0      BA GETS BA + ARA
FFA4 D9 06          ADC B  AR1
FFA6 6A 00   MUL2   DEC    0,X
FFA8 26 F0          BNE    MUL1     GO BACK IF NOT DONE
FFAA 31             INS             CLEAN UP THE STACK
FFAB 31             INS
FFAC 31             INS
FFAD 39      ARTS   RTS
             *
FFAE 36      DIVIDE PSH A  -        LOAD DIVIDEND INTO X3,4
FFAF 37             PSH B
FFB0 96 06          LDA A  AR1      LOAD DIVISOR FROM ARA
FFB2 D6 07          LDA B  AR0
FFB4 37             PSH B  -        PUT DIVISOR INTO X1,2
FFB5 36             PSH A
FFB6 34             DES             SET UP SPACE FOR COUNT
FFB7 30             TSX
FFB8 86 01          LDA A  #1       INITIALIZE COUNT
FFBA 6D 01          TST    1,X      TEST HI DIVISOR BIT
FFBC 2B 0B          BMI    DIV2     SKIP IF ON
FFBE 4C      DIV1   INC A  -        COUNT LEADING ZEROS
FFBF 68 02          ASL    2,X      LEFT JUSTIFY X1,2
FFC1 69 01          ROL    1,X
FFC3 2B 04          BMI    DIV2     SKIPS IF NO LEADING ZERO
FFC5 81 11          CMP A  #17      TESTS FOR ALL ZERO DIVISOR
FFC7 26 F5          BNE    DIV1     GO BACK IF BITS LEFT
FFC9 A7 00   DIV2   STA A  0,X      SAVE COUNTER
FFCB E6 03          LDA B  3,X      BA = ORIG DIVIDEND
FFCD A6 04          LDA A  4,X
FFCF 6F 03          CLR    3,X      CLEAR X3,4 FOR QUOTIENT
```

```
01007 FFD3 A0 02   DIV3    SUB A   2,X     START DIVIDE LOOP
01008 FFD5 E2 01           SBC B   1,X
01009 FFD7 24 07           BCC     DIV4    SKIP IF DIVIDEND < DIVISOR
01010 FFD9 AB 02           ADD A   2,X     RESTORE DIVIDEND IN BA
01011 FFDB E9 01           ADC B   1,X
01012 FFDD 0C              CLC             CLEAR THE CARRY
01013 FFDE 20 01           BRA     DIV5
01014 FFE0 0D      DIV4    SEC             SET CARRY TO 1
01015 FFE1 69 04   DIV5    ROL     4,X     SHIFT CARRY INTO QUOTIENT
01016 FFE3 69 03           ROL     3,X
01017 FFE5 64 01           LSR     1,X     SHIFT DIVISOR RIGHT 1
01018 FFE7 66 02           ROR     2,X
01019 FFE9 6A 00           DEC     0,X
01020 FFEB 26 E6           BNE     DIV3    GO BACK IF COUNTER NOT 0
01021 FFED D7 06           STA B   AR1     STORE REMAINDER IN ARA
01022 FFEF 97 07           STA A   AR0
01023 FFF1 31              INS             CLEANS UP THE STACK
01024 FFF2 31              INS
01025 FFF3 31              INS
01026 FFF4 33              PUL B   .       STORE QUOTIENT IN BA
01027 FFF5 32              PUL A
01028 FFF6 39              RTS
01029              *
01030 FFF8                 ORG     $FFF8
01031 FFF8 0100            FDB     $0100   INTERRUPT REQUEST VECTOR
01032 FFFA 0104            FDB     $0104   SOFTWARE INT.  VECTOR
01033 FFFC 0108            FDB     $0108   NON-MASKABLE-INT.  VECT
01034 FFFE FC75            FDB     RESTRT  RESTART VECTOR
01035              *   END
0
01037                      END
TP      0000
TMP1    0002
AB      0004
AR3     0004
A2      0005
ARA     0006
A1      0006
AR0     0007
DGIT    0008
STATS   0009
AXEC    000A
BUFADR  000C
BFEND   000E
ENDCHR  0010
LTCHR   0013
SRCADR  0014
DTADR   0016
ENDMEM  001A
CRPTR   001C
BUFFLO  0020
BFFHI   0022
SCNPTR  0024
SCASM   0026
>STASM  0028
ODVAL   002A
SYMVAL  002C
BKSAV   002E
BRKADR  0030
EIT     0032
BLKNAM  0033
IBUFF   0035
ACIANO  0038
NPRNT   003A
TRPTR   003C
```