```
                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 PFROMS 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.
                         *
                         *
      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 ←F001  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.
                    *──── F022 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 E1                  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.
                                         POINTS TO NEXT CHARACTER.
FD85 08       SCRD2   INX                CLEARS BYTE ON LINE 1.
FD86 E7 00            STA B   0,X        TESTS IF LINE 1 CLEARED.
FD88 8C E01F          CPX     #FRSTLN    SKIPS BACK IF NOT CLEARED.
FD8B 26 F8            BNE     SCRD2      RETURNS.
FD8D 39               RTS
```

```
                      *      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;
*
*
```

PDS-V3N

E 014   PDS-V3N

```
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.
```

```
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.
*
```

```
---
PAGE  004   PDS-V3N

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  #$03      TESTS FOR A "↑C" COMMAND.
FE7A 2D 25              BLT    BRKSET    SKIPS IF A "↑B" COMMAND.
FE7C 2E 45              BGT    EXIT      SKIPS FOR NEXT COMND TEST.
FE7E DE [illegible] CLFBRK LDX BRKADR    GETS ADDRESS OF BREAKPOINT.
FE80 A6 [illegible]     LDA A  BRKSAV    LOADS ORIG BYTE CONTENTS.
FE82 A7 00              STA A  0,X       RESTORES BYTE DATA.
FE84 20 66              BRA    DSPADR    GOES TO OPEN THE LOCATION.
                 *
                 *
FE86 81 20       CHANGE CMP A  #$[illegible] 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.
```

```
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 COMND 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.
```

```
                    *
                    * 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          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).
*
```

```
0360 FF22 A6 00   ASCBIN LDA A  0,X       GETS CHR TO FORM BASE.
0361 FF24 81 2E          CMP A  #$'.      TESTS FOR DECML STRNG.
0362 FF26 2D 06          BLT    OCT       SKIPS IF BASE 8 (*).
0363 FF28 2E 09          BGT    HEX       SKIPS IF BASE 16.
0364 FF2A 86 0A          LDA A  #10       LOADS BASE 10 FOR CONVERSN.
0365 FF2C 20 02          BRA    ASC1      SKIPS TO INC. TEXT POINTR.
0366 FF2E 86 08   OCT    LDA A  #8        LOADS BASE 8 FOR CONVERSION.
0367 FF30 08      ASC1   INX              INCREMENT PTR TO NEXT CHAR.
0368 FF31 20 02          BRA    ASC2      SKIPS TO SAVE BASE.
0369 FF33 86 10   HEX    LDA A  #16       LOADS BASE 16 FOR CONVERN.
0370 FF35 97 07   ASC2   STA A  AR0       SAVES BASE IN BASE#.
00371             *
00372             *
00373 FF37 5F     ENTR2  CLR B  NUMBER    GETS 0.
00374 FF38 37            PSH B  -         (LOW NUMBER ON STACK).
00375 FF39 D7 06         STA B  AR1       CLEARS HI OF BASE.
00376 FF3B A6 00  NXTCHR LDA A  0,X       GETS CHAR TO CONVERT.
00377 FF3D 08            INX              INC TO NEXT CHARACTER.
0(  8 FF3E 81 30         CMP A  #$'0      TESTS FOR END-OF-STRING.
00379 FF40 2D 20         BLT    AEXIT     EXITS IF END.
00380 FF42 80 30         SUB A  #$'0      FORMS B. C. D. NUMBER.
00381 FF44 81 0A         CMP A  #10       TESTS IF DECIMAL DIGIT.
00382 FF46 2D 0A         BLT    ASC3      SKIPS IF DECIMAL.
00383 FF48 81 10         CMP A  #16       TESTS FOR END OF STRING.
00384 FF4A 2F 16         BLE    AEXIT     EXITS IF NOT A HEX DIGIT.
00385 FF4C 80 07         SUB A  #7        FORMS A HEX B. C. D. DIGIT.
00386 FF4E 81 10         CMP A  #16       TESTS FOR END-OF-STRING.
00387 FF50 2C 10         BGE    AEXIT     EXITS IF CHAR > "F".
00388 FF52 97 08  ASC3   STA A  DIGIT     SAVES DIGIT FOR ADD.
00389             *
00390 FF54 DF 00  CNVASC STX    TMP       SAVES INDEX REG FOR MULT.
00391 FF56 32            PUL A  -         RESTORES LO OF "NUMBER".
00392 FF57 8D 3A         BSR    MULT      NUMBER GETS NUMBER * BASE.
00393 FF59 9B 08         ADD A  DIGIT     NUMBER GETS NUMBER + DIGIT.
00394 FF5B C9 00         ADC B  #0
00395 FF5D 36            PSH A  -         SAVES LO OF NUMBER
00396 FF5E DE 00         LDX    TMP       RESTORES STRING POINTER
00397 FF60 20 D9         BRA    NXTCHR    GOES TO CONVRT NEXT CHAR.
00398 FF62 32     AEXIT  PUL A  -         RESTORES "NUMBER" IN BA.
00399 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).
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.
```

```
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 DIVISOR 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 FE4A    SWI    FDB     BKENTR   SOFTWARE INT.  VECTOR ADDR.
FFFC 0108    NMI    FDB     $0108    NON-MASKABLE-INT.  VECT.
FFFE FC00    RST    FDB     $FC00    RESTART VECTOR ADDRESS.
             *
             *
             *
             *  END OF PDS SOURCE LISTING.
             *
             *
                    END
TOTAL ERRORS 00000
```