```
00001                    NAM    PDS-V3N
00002                    OPT    O,NOG
00003                *
00004                *
00005                *
00006                *       PROGRAMMED BY ERIC JAMESON
00007                *
00008                *
00009                *
00010                *       COPYRIGHT 1976 SPHERE CORPORATION
00011                *       940 N.  4TH EA. ; NORTH SALT LAKE, UTAH  84054
00012                *
00013                *
00014                *
00015                *       SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,
00016                *       DISTRIBUTION AND USE OF THE PDS SOFTWARE.
00017                *       NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE
00018                *       WRITTEN PERMISSION OF SPHERE CORP.
00019                *
00020                *
00021                *
00022                *
00023                * THE PROGRAM DEVELOPMENT SYSTEM (PDS V3N) IS A SET OF
00024                * PROGRAMS RDSIDING ON ERASABLE PROGRAMMABLE READ ONLY
00025                * MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS
00026                * SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE
00027                * DEVELOPMENT OF COMPUTER PROGRAMS.
00028                *     TOWARD THIS END, THE 4 PDS EPROMS CONTAIN A CURSOR
00029                * BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING
00030                * AID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16
00031                * BIT MULTIPLY AND DIVIDE, ASCII-TO-BINARY, AND
00032                * BINARY-TO-ASCII ROUTINES.
00033                *
00034                *
00035                *
00036                *     THE PDS-V3N PROM SET WAS WRITTEN IN ORDER TO RUN THE
00037                * NEW KEYBOARD.  CHANGES WERE MADE IN THE EDITOR AND THE
00038                * DEBUGGER.  AS THE NEW PROMS ARE A GREAT IMPROVEMENT OVER
00039                * THE V3A PROMM SET, A VERSION KNOWN AS PDS-V3D WAS MADE
00040                * WOULD RUN ON THE OLD KEYBOARDS.  THE ONLY DIFFERENCE
00041                * BETWEEN THE V3D AND THE V3N PPROMS ARE THAT THE PIA
00042                * ADDRESS IS CHANGED FROM F040 ON V3N TO F000 ON V3D.
00043                * CHANGES WERE ALSO MADE IN THE DEBUGGER AND THE
00044                * EDITOR.  THE EDITOR CHANGES WERE THAT INSERT AND DELETE
00045                * ARE NOW AT THE TOP OF THE PAGE AND THAT THERE IS A REEDIT
00046                * COMMAND IN THE EXEC (CTRL R) TO ALLOW RE-EDITING.  THE
00047                * ENTRY TO THE DEBUGGER FROM THE BREAKPOINT INSTRUCTION
00048                * HAS BEEN CHANGED SO THAT THE RETURN ADDRESS FOR THE
00049                * BREAKPOINT IS NOW CALCULATED WHEN THE BREAKPOINT IS
00050                * ENCOUNTERED AND GOES TO THE DEBUGGER.  IN ADDITION
00051                * THERE ARE 2 NEW INSTRUCTIONS:  ↑J FOR DOING A JSR TO
00052                * A ROUTINE AND ↑X TO EXIT BACK TO THE EXEC.
00053                *
```

```
00055                    *                  MEMORY MAP
00056                    *
00057   0000             TMP     EQU 2      $00
00058   0002             TMP1    EQU 2      $02
00059   0004             ARB     EQU 2      $04        16 BIT ACC. PSEUDO REG B.
00060   0004             AR3     EQU 1      $04        _HI BYTE OF ARB.
00061   0005             AR2     EQU 1      $05        _LO BYTE OF ARB.
00062   0006             ARA     EQU 2      $06        16 BIY ARITH PSEUDN REG A.
00063   0006             AR1     EQU 1      $06        _HI BYTE OF ARA.
00064   0007             AR0     EQU 1      $07        _LO BYTE OF ARA.
00065   0008             DIGIT   EQU 1      $08        BYTE USED BY ASCBIN FOR TMP.
00066   000A             OUTEND  EQU        $0A        END OF OUTPUT BUFFER TEXT.
00067   000C             BUFADR  EQU 2      $0C        START OF I/O BUFFER (PTR)
00068   000E             BUFEND  EQU 2      $0E        PTR. TO END OF I/O BUFFR.
00069   0011             OUTBUF  EQU        $11        START OF OUTPUT BUFFER.
00070   0014             SRCADR  EQU        $14        SOURCE FOR TEXT MOVES.
00071   0016             DSTADR  EQU 2      $16        DEST. ADDR. FOR TEXT MOVE.
00072   001A             ENDMEM  EQU 2      $1A        LAST ADDRES OF REAL MEMORY.
00073   001C             CSRPTR  EQU 2      $1C        PTR TO CURSER ON SCREEN.
00074   001E             BUFPTR  EQU        $1E        TEMP PTR USED BY OUTSTR.
00075   0020             BUFFLO  EQU 2      $20        PTR TO END OF LOW EDIT TXT.
00076   0022             BUFFHI  EQU 2      $22        PTR TO START OF HI TEXT.
00077   0024             SCNPTR  EQU 2      $24        PTR. TO BUFFRD TXT START.
00078   0026             SRCASM  EQU 2      $26        PTR TO ASSMBLR SOURCE CODE.
00079   002A             OPNDVAL EQU        $2A        HAS ASSMBLR OPERND VALUE.
00080   002C             SYMVAL  EQU        $2C        VALUE PUT IN ASSM. SYMTBL.
00081   002E             BRKSAV  EQU 1      $2E        TEMP SAVE FOR BRKPT DATA.
00082   0030             BRKADR  EQU 2      $30        ADDDRESS OF BREAKPOINT.
00083   0032             EDIT    EQU 1      $32        0 IF EDITOR IS NOT RUNNING.
00084   0035             IOBUFF  EQU 3      $35        I-O BUFFER FOR DEBUGGER.
00085   0040             PCVAL   EQU 3      $40        PROGRAM COUNTER FOR ASSM.
00086                    *
00087   E01F             FRSTLN  EQU        $E01F      RIGHTMOST CHAR OF LINE ONE.
00088   E1E0             LASTLN  EQU        $E1E0      LEFT SIDE OF BOTTOM OF CRT.
00089   E1FF             LASTCH  EQU        $E1FF      LAST CHAR ON CRT DISPLAY.
00090   F040             KBDPIA  EQU        $F040      ADDRESS OF PIA FOR KBD/2.
00091                    *KEYBOARD PIA ADDRESS FOR OLD KEYBOARD (KBD/1A) IS F000.
                         HOMECH
        E000             ~~HOMPOS~~ EQU     $E000
        0060             ENDCHR  EQU        $60


00093   FE71             INPCHR  EQU        $FE71      INPUTS A CHARACTER.
00094   FE64             DEBUG   EQU        $FE64      DEBUGGER ROUTINE.
00095   FF22             ASCBIN  EQU        $FF22      ASCII TO BINARY ROUTINE.
```

```
00097                    *                INITIALIZATION
00098                    *
00099                    *
00100                    *
00101                    *   THE INITIALIZATION ROUTINES SET UP THE INITIAL VALUES
00102                    * UPON SYSTEM RESET.
00103                    *
00104                    *
00105 FC00                       ORG     $FC00
00106 FC00 8E 01FF RESTRT LDS    #$1FF     SETS STACK POINTER
00107 FC03 30              TSX             MOVES STK PTR TO INDEX REG
00108 FC04 DF 26           STX   SRCASM    SETS ASSEMBLR OUTPUT PTR
00109 FC06 DF 0C           STX   BUFADR    INIT INPUT BUFFR ADDR.
00110 FC08 86 1F           LDA A #$1F      INTLZ. KEYBOARD PIA.
00111 FC0A B7 F041         STA A KBDPIA+1  PIA CONTROL REG. ADDRESS.
00112 FC0D CE 0FFF         LDX   #$FFF     LAST LOCATN OF MEMORY.
00113 FC10 DF 0E           STX   BUFEND    INIT END-OF-EDIT BUFFR.
00114 FC12 DF 1A           STX   ENDMEM    INIT END-OF-MEM ADDR.
00115                    *
00116                    *
00117                    *
00118                    *
00119                    *
00120                    *                COMMAND LANGUAGE
00121                    *
00122                    *    THIS EXECUTIVE ACCEPTS COMMANDS FROM THE KEYBOARD TO
00123                    * DETERMINE WHAT UTILITY IS TO BE RUN.  INVALID COMMANDS
00124                    * WILL SPACE THE CURSOR DOWN ONE LINE. DO NOT SPACE OFF THE
00125                    * BOTTOM OF THE SCREEN.
00126                    *
00127                    *
00128 FC14 8D 21    EXEC   BSR    HOME      CSRPTR IS HOMED.
00129 FC16 8D 25           BSR    CLEAR     CLEARS SCREEN
00130 FC18 8D 73    EXEC1  BSR    CR1       CRLF FOR NEW LINE.
00131 FC1A BD FE71         JSR    INPCHR    GETS & DISPLAYS CHR.
00132 FC1D 81 01           CMP A  #$01      TESTS IF ASSEMBLY COMMD.
00133 FC1F 26 03           BNE    EXEC2     SKIPS IF OTHER COMMAND.
00134 FC21 BD FDA1         JSR    ASMBLR    JUMPS TO ASSEMBL PRROGRM.
00135 FC24 81 05    EXEC2  CMP A  #$05      TESTS IF CONTRL 'E'.
00136 FC26 26 02           BNE    EXEC3     SKIPS IF NOT EDIT CMD.
00137 FC28 8D 3D           BSR    EDITOR    JUMPS TO EDIT TEXT.
00138 FC2A 81 12    EXEC3  CMP A  #$12      TESTS FOR A ↑R COMMAND.
00139 FC2C 26 02           BNE    EXEC4     SKIPS IF NOT A REEDIT COMND.
00140 FC2E 8D 3F           BSR    REEDIT    GOES TO REEDIT TEXT.
00141 FC30 81 04    EXEC4  CMP A  #$04      TESTS FOR CONTRL 'D'.
00142 FC32 26 E4           BNE    EXEC1     SKIPS BACK FOR A NEW COMND.
00143 FC34 7E FE64         JMP    DEBUG     JUMPS TO DEBUGGER.
```

```
00145                *                    THE EDITOR
00146                *
00147                *
00148                *     THE EDITOR ALLOWS INPUT FROM THE KEYBOARD INTO A
00149                * BUFFER MEMORY.  INPUT IS DISPLAYED ON THE SCREEN.  WHEN
00150                * IT IS TYPED IN, THE SCREEN TEXT CAN THEN BE EDITED BY USE
00151                * OF THE CURSOR.  WHEN THE SCREEN IS FULL OR EDITING IS
00152                * FINISHED, THE DATA IS SCROLLED OFF THE SCREEN INTO THE
00153                * EDIT BUFFER.  WHEN TEXT IS SCROLLED OFF THE TOP OF THE
00154                * SCREEN, IT IS STORED FROM THE BUFFER ADDRESS POINTER
00155                * (BUFADR) TO THE LOW BUFFER POINTER (BUFFLO).  BUFFLO
00156                * POINTS TO THE END OF TEXT + ONE (I.E.  IT POINTS TO THE
00157                * FIRST UNUSED BYTE).  WHEN IT IS SCROLLED OFF THE BOTTOM
00158                * OF THE SCREEN, IT IS STORED IN THE TOP OF THE EDIT BUFFER.
00159                * THE TEXT GOES FROM THE HIGH BUFFER POINTER (BUFFHI) TO
00160                * THE END OF BUFFER POINTER (BUFEND).  BUFFHI POINTS TO
00161                * THE END OF TEXT - ONE (I.E.  IT POINTS TO THE LAST UNUSED
00162                * BYTE IN THE BUFFER).  WHEN THE TEXT IS SCROLLED UP
00163                * OFF THE TOP OF THE SCREEN, TEXT IS TAKEN FROM THE HIGH
00164                * AREA OF THE EDIT BUFFER AND DISPLAYED ON THE LAST LINE OF
00165                * THE SCREEN.  WHEN TEXT IS SCROLLED DOWN OFF THE BOTTOM,
00166                * TEXT, IF ANY EXISTS, IS MOVED FROM THE LOW EDIT BUFFER
00167                * AREA TO THE TOP LINE OF THE SCREEN.
00168                *
00169                *
00170                *
00171                *
00172                *
00173                *
00174                *
00175                *
00176                *
00177                *
00178                *                    POINTERS USED
00179                *
00180                * CSRPTR   POSITION OF CHARACTERS INSERTED ON THE SCREEN.
00181                * SCNPTR   POSITION OF START OF EDITED TEXT ON SCREEN.
00182                * BUFADR   START OF TEXT BUFFER IN MAIN MEMORY.
00183                * BUFEND   END OF TEXT BUFFER IN MAIN MEMORY.
00184                * BUFFLO   END OF TEXT SCROLLED OFF TOP OF SCREEN.
00185                * BUFFHI   START OF TEXT SCROLED OFF BOTTOM OF SCREEN.
00186                * BUFLEN   NOT CURRENTLY USED.
00187                *
00188                *
00189                *
00190                *
00191                *
00192                *                    EDITOR COMMANDS
00193                *
00194          ^Q    * "UP ARROW"  MOVES CURSOR UP ONE LINE; CSRPTR GETS
00195                *  CSRPTR-32; CALL NDRFLO.
00196                *
00197          ^S    * "DOWN ARROW"  MOVES CURSOR DOWN ONE LINE; CSRPTR GETS
00198                *  CSRPTR+32; CALL OVRFLO.
00199                *
00200          ^R    * "RIGHT ARROW"  MOVES CURSOR ONE POSITION RIGHT; CSRPTR
00201                *  GETS CSRPTR+1; CALL OVRFLO.
```

```
00203          ^T  * "LEFT ARROW"  MOVES CURSOR ONE POSITION LEFT; CSRPTR
00204              *  GETS CSRPTR-1; CALL NDRFLO.
00205              *
00206          ^   * "CONTROL & LEFT ARROW (ON KEYBOARD)"  LEFT JUSTIFY CURSOR;
00207          ---* CSRPTR GETS CSRPTR TRUNCATED; CALL NDRFLO FOR SCNLOC CHK.
00208              *
00209              * "PUTCHR"  OUTPUTS CHARACTER; CSRPTR GETS
00210              *  CSRPTR+1; GOES TO OVRFLO.
00211              *
00212          ESC * "ENDCHR"  TERMINATION CHAR; CLEAR EDIT FLAG;
00213              *  EXIT THE EDITOR.
00214              *
00215      ^@ - ^L * "HOME"  HOMES CURSOR POINTER; CSRPTR GETS E000; NDRFLO.
00216              *
00217      ^U - ^_ * "CLEAR"  CSRPTR TO END OF THE SCREEN GETS SPACES.
00218              *
00219              * "CTRL I"  INSERT A LINE AT THE ~~LAST~~ FIRST LINE ON THE SCREEN;
00220              *  CALL OVR1 (SCROLLS UP ONE LINE); CSRPTR GETS ~~E1E0~~.
00221              *
00222              * "CTRL D"  DELETE ~~LAST~~ FIRST LINE; SCROLL ~~DOWN~~ UP (UNDR2);
00223              *  CSRPTR GETS ~~E1E0~~.
00224              *                    E000   1st line - scroll up
00225              *
00226              *
00227              *
00228              * OVERFLOW CHECKS IF SCROLL UP IS NEEDED; IF IT IS, IT
00229              *  SCROLLS UP AND MOVES DATA TO & FROM THE BUFFEERS.
00230              *
00231              * OVRFLO:  IF CSRPTR < E200 THEN RETURN; IF EDIT IS ON THEN
00232              * OVR1:    BUFFLO+ GETS SCNPTR TO 'C.R.';
00233              * DSTADR GETS CSRPTR GETS E1E0 (LAST LINE ON SCREEN);
00234              * IF EDIT IS ON AND BUFFHI < BUFEND THEN MOVE THE TEXT
00235              * (THE STRING FROM BUFFHI TO 'C.R.') TO THE LAST LINE.
00236              *
00237              *
00238              *
00239              *
00240              * UNDERFLOW CHECKS IF SCROLL DOWN IS NEEDED AND MOVES
00241              *  DATA TO AND FROM THE BUFFERS.  CURSOR HAD BEEN
00242              *  MOVED OFF THE TOP OF THE SCREEN AND IS NOW PUT AT THE
00243              *  HOME POSITION ON THE SCREEN.
00244              *
00245              * NDRFLO:  IF CSRPTR > DFFF THEN RETURN (GO TO OVRFLO);
00246              *  IF EDIT FLAG IS ON THEN MOVE LAST LINE TO BUFFHI
00247              *   ON DOWN; SCRLDN; CSRPTR GETS E000; MOVE LINE FROM
00248              *   BUFFLO TO FIRST LINE ON THE CRT.
00249              *
00250              *
00251              *
00252              *
00253              *  NOTE:  DON'T SCROLL OFF SCREEN IN EXEC UNTIL AFTER
00254              *    THE EDITOR HAS BEEN RUN.
00255              *
00256              *  NOTE:  EVERY LINE MUST HAVE A 'C.R.  ON IT.
```

```
00258 FC37 CE E000 HOME   LDX    #$E000   LOADS HOME POSITION.
00259 FC3A DF 1C          STX    CSRPTR   STORES HOME IN CURSOR PTR.
00260 FC3C 39             RTS             RETURNS TO CALLER.
00261                *
00262                *
00263 FC3D C6 60    CLEAR LDA B  #$60      LOADS BLANK (C.R.).
00264 FC3F CE E200        LDX    #LASTCH+1  LOADS END-OF-SCREEN PTR.
00265 FC42 09       CLEAR1 DEX             DECREMENTS BLANKING PTR.
00266 FC43 E7 00          STA B  0,X       BLANKS LOCATION.
00267 FC45 9C 1C          CPX    CSRPTR    TESTS IF DONE.
00268 FC47 26 F9          BNE    CLEAR1    BRANCHES BACK IF NOT DONE.
00269 FC49 39             RTS             RETURNS.
00270                *
00271                *
00272                *
00273                *       GETCHR INPUTS A CHARACTER INTO ACC A WITHOUT
00274                *       MOVING THE CURSOR, AND BLINKS THE CURSOR.
00275                *
00276 FC4A DE 1C    GETCHR LDX    CSRPTR   LOADS CRT CURSOR POSITION.
00277 FC4C 63 00          COM    0,X      COMPLIMENT (FLASH POSITION).
00278 FC4E CE 26F0        LDX    #9968    LOADS BLINK COUNT VALUE.
00279 FC51 09       GET1  DEX             COUNT GETS COUNT-1.
00280 FC52 27 F6          BEQ    GETCHR   RESETS CTR WHEN TIMED OUT.
00281 FC54 86 40          LDA A  #$40     LOADS MASK FOR CA2 FLAG.
00282 FC56 B5 F041        BIT A  KBDPIA+1 TESTS IF A CHAR. TYPED IN.
00283 FC59 27 F6          BEQ    GET1     BRANCH IF CHAR NOT ENTERED.
00284 FC5B DE 1C          LDX    CSRPTR   LOADS CURSOR POSITION.
00285 FC5D A6 00          LDA A  0,X      TESTS IF BLINKMD (SOLID).
00286 FC5F 2A 02          BPL    GET2     SKIPS IF NOT BLINKED.
00287 FC61 63 00          COM    0,X      CLEARS THE CHARACTER.
00288 FC63 B6 F040  GET2  LDA A  KBDPIA   LOADS A WITH KEYBRD CHAR.
00289 FC66 39             RTS             RETURNS TO CALLER.
00290                *
00291                *
00292                *
00293                *
00294                *       EDITOR IS THE MAIN ENTRY POINT FOR EDITING.
00295                *
00296 FC67 DE 0C    EDITOR LDX    BUFADR   BUFFLO GETS THE
00297 FC69 DF 20          STX    BUFFLO   VALUE OF BUFADR.
00298 FC6B DE 0E          LDX    BUFEND   BUFFHI GETS THE
00299 FC6D DF 22          STX    BUFFHI   VALUE OF BUFEND.
00300 FC6F 8D C6    REEDIT BSR    HOME     ENTRY POINT FOR
00301 FC71 8D CA          BSR    CLEAR    RE-EDITING TEXT.
00302 FC73 97 32    EDITRD STA A  EDIT     TURNS ON EDIT MODE.
00303 FC75 DE 1C    EDITIN LDX    CSRPTR   SETS SCNPTR TO CSRPTR.
00304 FC77 DF 24          STX    SCNPTR
00305                *
00306                *
00307 FC79 8D CF    EDREAD BSR    GETCHR   X GETS CSRPTR & A GETS CHR.
00308 FC7B 81 1B    ENDCHR CMP A  #$1B     TESTS FOR AN "ESC" CHAR.
00309 FC7D 26 04          BNE    ED1      SKIPS IF NOT EDIT END.
00310 FC7F 7F 0032        CLR    EDIT     TURNS OFF EDIT FLAG.
00311 FC82 39             RTS             EXITS THE EDITOR.
00312 FC83 8D 0A    ED1   BSR    INSERT   EDITS CHARACTER.
00313 FC85 20 F2          BRA    EDREAD   GOES FOR NEXT CHARACTER.
```

```
00315                    *
00316                    *
00317                    *       FOLLOWING IS THE MAIN EDITOR EXECUTION LOOP.
00318                    *
00319                    *
00320                    *
00321                    *
00322                    *
00323 FC87 81 0D  CR     CMP A   #$0D      TESTS FOR CARRIAGE RETURN.
00324 FC89 2D AC         BLT     HOME      SKIPS IF HOME CURSR COMND.
00325 FC8B 2E 12         BGT     RTCSR     GOES TO NEXT COMND TEST.
00326 FC8D 86 60  CR1    LDA A   #$60      LOADS INTERNAL C. R.  VALUE.
00327                    *
00328                    *
00329                    *
00330                    *
00331 FC8F 81 09  INSERT CMP A   #$09      TESTS FOR A CONTROL 'I'.
00332 FC91 2D 16         BLT     DELETE    SKIPS TO DELETE COMMD.
00333 FC93 2E F2         BGT     CR        SKIPS FOR NEXT TEST.
00334 FC95 D6 32  INSRT1 LDA B   EDIT      TESTS IF EDITOR IS ON.
00335 FC97 27 03         BEQ     INSRT2    SKIP TO EXIT IF EDITOR OFF.
00336 FC99 BD FD46       JSR     MOVE2     MOVES LAST LINE TO BUFFHI.
00337 FC9C 7E FD74 INSRT2 JMP    SCRLDN    MOVES ALL LINES DOWN ONE.
00338                    *
00339                    *
00340                    *
00341 FC9F 81 12  RTCSR  CMP A   #$12      TESTS FOR RIGHT ARROW.
00342 FCA1 2D 28         BLT     SUB32     SKIPS IF AN "UP ARROW".
00343 FCA3 2E 08         BGT     LFTCSR    SKIPS IF A "LEFT ARROW".
00344 FCA5 DE 1C  RTARRO LDX     CSRPTR    LOADS CURSOR POINTER.
00345 FCA7 20 1F         BRA     PUTCH1    STORES & INCREMENTS CSR.
00346                    *
00347                    *
00348                    *
00349 FCA9 8D 67  DELETE BSR     OVR1A     SCROLLS UP ONE LINE.
00350 FCAB 20 40         BRA     OVR3      MOVES NEW LAST SCREEN LINE.
00351                    *
00352                    *
00353                    *
00354 FCAD 81 14  LFTCSR CMP A   #$14      TESTS IF "LEFT ARROW".
00355 FCAF 2D 24         BLT     ADD32     SKIPS IF "DOWN ARROW".
00356 FCB1 2E 03         BGT     CLER      SKIPS FOR NEXT TEST.
00357 FCB3 09            DEX               SUB.  1 FROM CSRPTR.
00358 FCB4 20 25         BRA     ADD2      STORES CURSOR POINTER.
00359                    *
00360                    *
00361                    *
00362 FCB6 81 1F  CLER   CMP A   #$1F      TESTS FOR CTRL BACK ARROW.
00363 FCB8 2D 83         BLT     CLEAR     GOES TO CLEAR SCREEN.
00364 FCBA 27 41         BEQ     LFTJST    MOVES CSR TO LEFT OF SCREEN.
00365                    *
00366                    *
00367                    * ALL OTHER CHARACTERS FALL THRU TO PUTCHR.
00368                    *
```

```
00370                    *       PUTCHR DISPLAYS A CHARACTER ON THE CRT DISPLAY AND
00371                    *         INCREMENTS THE CURSOR POINTER AS WELL AS CHECKING
00372                    *         AND HANDLING CARRIAGE RETURNS.
00373                    *
00374                    *
00375                    *
00376 FCBC DE 1C   PUTCHR LDX     CSRPTR    LOADS OLD CSRPTR.
00377 FCBE 81 0D         CMP A   #$0D      TESTS FOR EXTERNAL C. R.
00378 FCC0 27 04         BEQ     CRLF1     SKIPS TO DO A C. R.  L. F.
00379 FCC2 A7 00         STA A   0,X       DISPLAYS CHAR ON SCREEN.
00380 FCC4 81 60         CMP A   #$60      TESTS FOR INTERNAL C. R.
00381 FCC6 27 4C   CRLF1  BEQ     CRLF      SKIPS FOR CR.  LF.
00382 FCC8 08      PUTCH1 INX               INCREMENTS CSRPTR.
00383 FCC9 20 10         BRA     ADD2      TESTS FOR OVRFLO & UNDRFLO.

00385 FCCB DE 1C   SUB32  LDX     CSRPTR    LOADS CURRENT CRSR POSITION.
00386 FCCD C6 20         LDA B   #32       LOADS LOOP COUNT.
00387 FCCF 09      SUB32A DEX               DECREMENTS CSRPTR.
00388 FCD0 5A            DEC B   -         DECREMENTS LOOP COUNTR.
00389 FCD1 26 FC         BNE     SUB32A    SKIPS BACK IF NOT DONE.
00390 FCD3 20 06         BRA     ADD2      SKIPS TO CHECK UNDRFLO.

00392 FCD5 C6 20   ADD32  LDA B   #32       LOADS LOOP COUNTER.
00393 FCD7 08      ADD32A INX               INCRE. CSRPTR IN INDEX.
00394 FCD8 5A            DEC B   -         DCEREMENTS LOOP COUNTER.
00395 FCD9 26 FC         BNE     ADD32A    SKIPS BACK IF NOT DONE.
00396 FCDB DF 1C   ADD2   STX     CSRPTR    SAVES CSRPTR.

00398                    * NDRFLO (UNDERFLOW) CHECKS FOR THE CURSOR GOING OFF THE
00399                    *  TOP OF THE SCREEN.  THE INDEX*REG. CONTAINS THE CURSOR
00400                    *  POINTER WHEN THE ROUTINE IS ENTERED.
00401                    *
00402                    *
00403 FCDD 8C E000 NDRFLO CPX    #$E000    TESTS IF CSRPTR >= DFFF.
00404 FCE0 2C 04         BGE     OVRFLO    SKIPS IF CSRPTR GREATER.
00405 FCE2 8D B1         BSR     INSRT1    SCROLLS DOWN & MOVES LINE.
00406 FCE4 8D 32         BSR     MOVE3     MOVES BUFFLO TO TOP OF CRT.
```

```
00408                    *
00409                    *
00410                    *     OVERFLOW CHECKS FOR SCROLLING UP (CURSOR IS OFF
00411                    * THE BOTTON OF THE SCREEN); INDEX CONTAINS THE CURSOR
00412                    * POINTER UPON ENTRY.
00413                    *
00414                    *
00415 FCE6 8C E200 OVRFLO CPX     #$E200   TESTS AND EXITS IF
00416 FCE9 2B 18          BMI     OVREXT   CURSOR ON SCREEN.
00417 FCEB 8D 17          BSR     OVR1     DOES OVR1 CHECKING.
00418 FCED D6 32    OVR3  LDA B   EDIT     TESTS IF EDIT IS ON.
00419 FCEF 27 12          BEQ     OVREXT   EXITS IF IT IS OFF.
00420 FCF1 DE 22          LDX     BUFFHI   LOADS HI TEXT POINTR.
00421 FCF3 9C 0E          CPX     BUFEND   TESTS IF PTRS NOT EQU.
00422 FCF5 27 0C          BEQ     OVREXT   EXITS IF NO TEXT.
00423 FCF7 8D 3C          BSR     MOVE1A   MOVES CHRS TO LAST LINE.
00424 FCF9 DE 14          LDX     SRCADR   RESETS NEW BUFFHI
00425 FCFB DF 22          STX     BUFFHI   LOCATION.
00426                    *
00427                    *
00428                    *
00429                    *
00430                    *     FOLLOWING ROUTINE MOVES THE CURSER TO THE LEFT.
00431                    *
00432 FCFD D6 1D    LFTJST LDA B  CSRPTR+1 LOADS LOW BYTE OF PTR.
00433 FCFF C4 E0          AND B   #$E0     TRUNCATES TO LEFT OF LINE.
00434 FD01 D7 1D          STA B   CSRPTR+1 SAVES L. J. ED PTR.
00435 FD03 39       OVREXT RTS             RETURNS TO EDITOR.
00436                    *
00437                    *
00438                    *
00439                    *
00440                    *     OVR1 DOES ACTUAL SCROLLING UP.
00441                    *
00442 FD04 D6 32    OVR1  LDA B   EDIT     TESTS IF EDIT IS ON.
00443 FD06 27 0A          BEQ     OVR1A    SKIPS IF EDIT OFF.
00444 FD08 DE 20          LDX     BUFFLO   LOADS TEXT PTR LOW.
00445 FD0A DF 16          STX     DSTADR   DESTINATION OF TEXT MOVE.
00446 FD0C DE 24          LDX     SCNPTR   SOURCE FOR MOVE.
00447 FD0E 8D 26          BSR     MOVE1    MOVES LIN1 TO BUFFFLO.
00448 FD10 DF 20          STX     BUFFLO   SAVES NEW BUFFLO PTR.
00449 FD12 20 4B    OVR1A BRA     SCRLUP   SCROLLS SCREEN UP 1.



00451                    *     FOLLOWING ROUTINE MOVES THE CURSOR.
00452                    *
00453 FD14 8D BF    CRLF  BSR     ADD32    LINE FEED.
00454 FD16 20 E5          BRA     LFTJST   CARRIAGE RETURN.
```

```
00456                    *     MOVE INSTRUCTIONS MOVE FROM ONE BUFFER AREA TO
00457                    * ANOTHER BUFFER AREA.
00458                    *
00459                    *
00460                    *
00461                    *     MOVE3 CALCULATES THE SOURCE ADDRESS OF THE DATA IN
00462                    * BUFFLO (IF IT EXISTS) FOR MOVING TO THE FIRST LINE ON
00463                    * THE CRT.   MOVE 1 IS THEN ENTERED TO DO THE MOVING.
00464                    *
00465                    *
00466 FD18 DE 24    MOVE3  LDX     SCNPTR   CSRPTR GETS E000 (HOME).
00467 FD1A DF 16           STX     DSTADR   SETS MOVE ADDRESS.
00468 FD1C DE 20           LDX     BUFFLO   LOADS LO BUFFR ADDR.
00469 FD1E 9C 0C           CPX     BUFADR   TESTS IF STRING EXISTS.
00470 FD20 27 23           BEQ     MOVEXT   EXITS IF EMPTY.
00471 FD22 09              DEX              MOVES BACK FROM BLANK.
00472 FD23 9C 0C    MV31   CPX     BUFADR   TESTS IF SRCADR = BUFFADR.
00473 FD25 27 08           BEQ     MV32     MOVES IF START OF LINE.
00474 FD27 09              DEX              NEXT LOWER CHAR.
00475 FD28 E6 00           LDA B   0,X      GETS SOURCE CHAR FOR TEST.
00476 FD2A C1 60           CMP B   #$60     TESTS FOR "C.R.".
00477 FD2C 26 F5           BNE     MV31     SKIPS BACK UNTIL "C.R.".
00478 FD2E 08              INX              POINTS BACK TO FIRST CHAR.
00479 FD2F DF 20    MV32   STX     BUFFLO   SAVES LO ADDRESS.
00480 FD31 20 03           BRA     MOVE1    MOVES DATA.


00482                    *          MOVE1
00483                    *
00484                    *     MOVE1 MOVES A SET OF CHARACTERS FROM EITHER THE TOP
00485                    * LINE OF THE SCREEN TO BUFFLO OR FROM BUFFHI TO THE
00486                    * BOTTOM LINE OF THE SCREEN.   THE SOURCE ADDRESS IS PASSED
00487                    * IN THE INDEX REG., THE DESTINATION ADDRESS IN DSTADR,
00488                    * AND THE MOVE IS TERMINATED BY A "C.R." IN THE LINE OF
00489                    * TEXT BEING MOVED.
00490                    *
00491                    *
00492                    *
00493                    *
00494 FD33 DE 14    MOVE   LDX     SRCADR   LOADS SOURCE ADDRESS INTO X.E
00495 FD35 08       MOVE1A INX              POINTS TO NEXT SOURCE CHAR.
00496 FD36 E6 00    MOVE1  LDA B   0,X      LOADS SOURCE CHARACTER.
00497 FD38 DF 14           STX     SRCADR   SAVES THE SOURCE POINTER.
00498 FD3A DE 16           LDX     DSTADR   LOADS DESTINATION ADDRESS.
00499 FD3C E7 00           STA B   0,X      STORES CHAR. IN DESTINATION.
00500 FD3E 08              INX              NEXT DESTINATION ADDRESS.
00501 FD3F DF 16           STX     DSTADR   SAVES DESTINATION PTR.
00502 FD41 C1 60           CMP B   #$60     TESTS IF MOVE FINSHED (CR).
00503 FD43 26 EE           BNE     MOVE     SKIPS BACK IF NOT DONE.
00504 FD45 39       MOVEXT RTS              RETURNS TO CALLER.
```

```
00506                    *          MOVE2 SUBROUTINE
00507                    *
00508                    *     THE MOVE2 SUBROUTINE MOVES THE LAST LINE ON THE
00509                    * SCREEN TO THE HIGH AREA OF THE BUFFER (BUFFHI) DURING
00510                    * SCROLLING.  THE TEXT IS TEMPORARILY STORED ON THE STACK
00511                    * DURING THE MOVE.  THE MOVE IS TERMINATED BY A "C. R. ".
00512                    * THE TEXT IS STORED AT BUFFHI ON DOWN.
00513                    *
00514                    *
00515                    *
00516                    *
00517 FD46 CE E1E0 MOVE2  LDX     #LASTLN  X GETS ADDR OF LAST LINE.
00518 FD49 5F          CLR B    -        SETS TERMINATION FOR
00519 FD4A 37       MV21  PSH B    STACK    POPPING.
00520 FD4B E6 00          LDA B    0,X      LOADS SOURCE CHAR.
00521 FD4D 08             INX               POINTS TO NEXT CHAR.
00522 FD4E C1 60          CMP B    #$60     TESTS IF LINE TO "C. R. "
00523 FD50 26 F8          BNE      MV21     MOVED TO STACK.
00524 FD52 DE 22    MV22  LDX      BUFFHI   INIT. DESTINATION.
00525 FD54 E7 00    MV23  STA B    0,X      STORES CHAR.
00526 FD56 09             DEX               POINTS TO NEXT LOCATION.
00527 FD57 DF 22          STX      BUFFHI   UPDATES BUFFER PTR.
00528 FD59 33             PUL B    -        GETS NEXT CHAR.
00529 FD5A C1 00          CMP B    #00      TESTS IF ALL CHRS STORED.
00530 FD5C 26 F6          BNE      MV23     SKIPS BACK IF NOT STORED.
00531 FD5E 39       MOVEX RTS               RETURNS TO CALLER.


00533                    *     SCROLLUP MOVES ALL LINES UP 1, & CLEARS LAST LINE.
00534                    *                    #HOMECH
00535 FD5F CE E000 SCRLUP LDX     #$E000   SETS CRT HOME POSITION.
00536 FD62 E6 20    SCRP1 LDA B    $20,X    GETS CHAR FROM NEXT LINE.
00537 FD64 E7 00          STA B    00,X     STORES CHAR ON PREV. LINE.
00538 FD66 08             INX               POINTS TO NEXT LINE.
00539 FD67 8C E1E0        CPX      #LASTLN  TESTS IF MOVE DONE.
00540 FD6A 26 F6          BNE      SCRP1    GOES BACK IF NOT DONE.
00541 FD6C DF 1C          STX      CSRPTR   SETS CSRPTR TO LAST LINE.
00542 FD6E DF 16          STX      DSTADR   INIT DEST FOR NEXT MOVE.
00543 FD70 BD FC3D        JSR      CLEAR    CLEARS LAST LINE.
00544 FD73 39             RTS               EXITS.
```

LDX ___
PSH A
LDA A 0 EX60
Loop CMP B 0,X
Loop BEQ
INC B
BNE
INX
BRA Loop

```
00546                    *       SCRLDOWN MOVES ALL LINES DOWN ONE AND
00547                    * CLEARS THE TOP LINE ON THE SCREEN.
00548                    *
00549 FD74 CE E1DF SCRLDN LDX     #LASTLN-1  INITIALIZES THE POINTER.
00550 FD77 E6 00    SCRD1  LDA B  0,X        LOADS DATA TO BE MOVED.
00551 FD79 E7 20           STA B  $20,X      MOVES DATA DOWN ONE LINE.
00552 FD7B DF 1C           STX    CSRPTR     SAVES CURSOR.
00553 FD7D 09              DEX               POINTS TO NEXT BYTE.
00554 FD7E 8C DFFF         CPX    #$DFFF     TESTS IF MOVE FINISHED.
00555 FD81 26 F4           BNE    SCRD1      SKIPS BACK IF NOT DONE.
00556 FD83 C6 60           LDA B  #$60       LOADS BLANK TO CLEAR LINE.
00557 FD85 08       SCRD2  INX               POINTS TO NEXT CHARACTER.
00558 FD86 E7 00           STA B  0,X        CLEARS BYTE ON LINE 1.
00559 FD88 8C E01F         CPX    #FRSTLN    TESTS IF LINE 1 CLEARED.
00560 FD8B 26 F8           BNE    SCRD2      SKIPS BACK IF NOT CLEARED.
00561 FD8D 39              RTS               RETURNS.
```

```
00563                    *       OUTSTRING PRINTS OUT THE STRING BETWEEN THE
00564                    *       OUTBUF POINTER AND THE BUFEND POINTER.
00565                    *
00566 FD8E DE 1A    OUTSTR LDX     OUTBUF    BUFPTR GETS START OF TEXT.
00567 FD90 A6 00    OUT1   LDA A   0,X       LOADS CHAR TO BE PUT OUT.
00568 FD92 DF 1E           STX     BUFPTR    SAVES SOURCE POINTER.
00569 FD94 BD FCBC         JSR     PUTCHR    PRINTS CHARACTER.
00570 FD97 DE 1E           LDX     BUFPTR    RESTORES POINTER.
00571 FD99 9C 0A           CPX     OUTEND    TESTS FOR END-OF-TEXT.
00572 FD9B 27 03           BEQ     OUT2      EXITS IF END OF TEXT.
00573 FD9D 08              INX               INCRE. PTR TO NEXT CHAR.
00574 FD9E 20 F0           BRA     OUT1      GOES BACK FOR NEXT CHAR.
00575 FDA0 39       OUT2   RTS               EXITS ROUTINE.
00576                    *
00577                    *       END OF EDITOR PROGRAM.
00578                    *
```

```
00580                *          THE MINI-ASSEMBLER
00581                *
00582                *
00583                *     THE MINI-ASSEMBLER IS A FIXED-FIELD ONE INSTRUCTION
00584                * PER LINE 2 PASS ASSEMBLER.  THE MINI-ASSSEMBLER FORMAT
00585                * IS DESCRIBED ON PAGES 9-2 AND 9-3 OF THE SPHERE
00586                * OPERATORS REFERENCE MANUAL.
00587                *     THE TWO PASSES ARE REQUIRED TO FORM THE LABEL
00588                * ADDRESSES.  THE SECOND PASS EQUATES THE ADDRESS FOR
00589                * LABELS REFRENCED BEFORE THEY ARE DEFINED IN THE PROGRAM.
00590                *
00591                *
00592                * ON ENTRY:
00593                * SRCASM = ADDRESS OF SOURCE TEXT TO BE ASSEMBLED.
00594                * BUFFLO = ADDRESSED OF OBJECT CODE PRODUCED.
00595                *
00596                * ON EXIT:
00597                * PCVAL (PROGRAM COUNTER VALUE) = LAST LOCATION OF
00598                * THE ASSEMBLED OBJECT PROGRAM.
00599                *
00600                *
00601                *
00602                *
00603                * ALGORITHM:
00604                *
00605                *ASMBLR:  SET PASS COUNT TO ZERO; SET PCVAL TO DSTASM;
00606                *ASM1A:  OPERAND VALUE FORMED IN "ONDVAL":
00607                * A GETS CHAR IN X6 (OPERAND TYPE); X GETS X+7;
00608                * IF CHAR X6 IS A "@" THEN ONDVAL GETS VALUE FROM SYMBOL
00609                *   TABLE ELSE ONDVAL GETS VALUE FROM ASCBIN CONVERSION;
00610                *SYMBL:  EQUATES SYMBOL (PC VALUE IS THE " " SYMBOL
00611                *[LABEL]) TO A LABEL VALUE:
00612                * SYMVAL GETS PCVAL;
00613                * IF X(1) IS AN "=" THEN SYMVAL GETS ONDVAL;
00614                * IF X(1) IS NOT A "=" OR A SPACE THEN IF SECOND PASS THEN
00615                *   EXIT ELSE START SECOND PASS;
00616                * LABEL ENTRY IN SYMBOL TABLE GETS SYMVAL;
00617                *LDOP:  PUT OPERATION CODE INTO THE OBJECT CODE:
00618                * CONVERT X(2)-X(3) INTO BINARY;
00619                * SAVE PCVAL;
00620                * P.C. GETS P.C. +1;
00621                *OPRND:  FORM OPERAND IN OBJECT CODE:
00622                * FORM ONDVAL INTO PROPER SIZE BASED ON CODE IN X(6);
00623                * STORE NEW OPERAND VALUE IN MEMORY;
00624                * P.C. GETS P.C. +1 OR 2;
00625                * GET NEXT LINE OF SOURCE;
00626                * GO TO ASM1A;
00627                *
00628                *
```

```
00630 FDA1 7F 0004 ASMBLR CLR     AR3       INIT. PASS CTR TO FRST PASS. )
00631 FDA4 DE 20   ASM1   LDX     BUFFLO    SETS PC CNTR TO START OF
00632 FDA6 DF 40          STX     PCVAL     OBJECT CODE.
00633 FDA8 DE 26          LDX     SRCASM    LOADS ADDR FOR FIRST LINE.
00634 FDAA DF 02   ASM1A  STX     TMP1      SAVES ADDR OF CURRENT LINE.
00635 FDAC A6 08          LDA A   8,X       LOADS SYMBOL (LABEL).
00636 FDAE E6 07          LDA B   7,X       LOADS OPERAND TYPE CODE.
00637 FDB0 C1 40          CMP B   #$'@      IF @, LOADS DATA IN SYMBOL
00638 FDB2 27 6B          BEQ     INDADR    ADDRESS, GOES TO SYMBL.
00639 FDB4 08             INX               SETS INDEX TO START OF
00640 FDB5 08             INX               OPERAND NUMBER.
00641 FDB6 08             INX
00642 FDB7 08             INX
00643 FDB8 08             INX
00644 FDB9 08             INX
00645 FDBA 08             INX
00646 FDBB BD FF22        JSR     ASCBIN    CONVRTS # TO BINARY IN B-A.
00647 FDBE D7 2A   ASM1B  STA B   ONDVAL    STORES OPERAND VALUE IN
00648 FDC0 97 2B          STA A   ONDVAL+1  ONDVAL.
00649                     *
00650                     *
00651                     *
00652                     * FOLLOWING FORMS THE VALUE FOR THE LABEL.
00653                     *
00654 FDC2 DE 02   SYMBL  LDX     TMP1      LOADS ORIG LINE PTR INTO X.
00655 FDC4 A6 00          LDA A   0,X       LOADS SYMBOL (LABEL).
00656 FDC6 E6 01          LDA B   1,X       LOADS LABEL CONTROL CHAR.
00657 FDC8 DE 40          LDX     PCVAL     LABEL VALUE GETS PCVAL.
00658 FDCA DF 2C          STX     SYMVAL    .
00659 FDCC C1 3D          CMP B   #$'=      TESTS IF LABLE IS EQUATED.
00660 FDCE 26 06          BNE     ASM2      SKIPS IF NOT EQUATED.
00661 FDD0 DE 2A          LDX     ONDVAL    LABEL VALUE (SYMVAL) GETS
00662 FDD2 DF 2C          STX     SYMVAL    THE OPERAND VALUE.
00663 FDD4 20 0E          BRA     ASM3      CONTINUES EVALUATION.
00664 FDD6 C1 20   ASM2   CMP B   #$'       TESTS FOR END-OF-PROGRAM.
00665 FDD8 27 0A          BEQ     ASM3      SKIPS IF SPACE (NOT END).
00666 FDDA 7D 0004        TST     AR3       TESTS IF SECOND PASS.
00667 FDDD 27 01          BEQ     ASM2A     EXITS IF SECOND PASS.
00668 FDDF 39             RTS               EXITS THE ASSEMBLER.
00669 FDE0 D7 04   ASM2A  STA B   AR3       SETS CTR TO SECOND PASS.
00670 FDE2 20 C0          BRA     ASM1      GOES BACK FOR SECOND PASS.
00671                     *
00672                     *
00673                     *
00674                     * FOLLOWING PUTS THE LABEL VALUE IN THE SYMBOL TABLE.
00675                     *
00676 FDE4 8D 41   ASM3   BSR     SYMPTR    X GETS SYMBL TABL ENTRY ADR. E
00677 FDE6 96 2C          LDA A   SYMVAL    STORES THE LABEL
00678 FDE8 A7 00          STA A   0,X       ADDRESS (SYMVAL) INTO THE
00679 FDEA 96 2D          LDA A   SYMVAL+1  SYMBOL TABEL.
00680 FDEC A7 01          STA A   1,X
```

```
00682                    * FOLLOWING FORMS THE OPERATION CODE.
00683                    *
00684 FDEE DE 02   LDOP   LDX     TMP1      LOADS ORIG LINE POINTER.
00685 FDF0 08             INX               SETS X TO POINT TO
00686 FDF1 08             INX               THE OP CODE CHARS.
00687 FDF2 08             INX
00688 FDF3 A6 00          LDA A   0,X       GETS OP CODE CHAR INTO A.
00689 FDF5 81 20          CMP A   #$'       TESTS IF OP CODE EXISTS.
00690 FDF7 27 0A          BEQ     OPRND     SKIPS IF NONEXISTANT.
00691 FDF9 BD FF22        JSR     ASCBIN    CONVRTS OP CODE TO BINARY.
00692 FDFC DE 40          LDX     PCVAL     LOADS POINTR TO OBJECT CODE.
00693 FDFE A7 00          STA A   0,X       STORS OP INTO OBJECT CODE.
00694 FE00 08             INX               SETS TO NEXT OBJ CODE LOCTN.
00695 FE01 DF 40          STX     PCVAL     SAVES P. C.  POINTER.
00696                    *
00697                    *
00698                    *
00699                    * FOLLOWING STORES INTO THE OBJECT CODE THE SIZED OPERAND.
00700                    *
00701 FE03 DE 02   OPRND  LDX     TMP1      LOADS SOURCE LINE POINTER.
00702 FE05 A6 06          LDA A   6,X       LOADS OPERAND SIZE CHAR.
00703 FE07 DE 40          LDX     PCVAL     LOADS X WITH OBJ CODE PTR.
00704 FE09 81 45          CMP A   #$'E      TESTS LENGTH TYPE.
00705 FE0B 2E 31          BGT     RELTIV    SKIPS IF AN "R" OPERAND.
00706 FE0D 27 21          BEQ     EXTEND    SKIPS IF AN "E" SIZE OPRND.
00707 FE0F 81 44          CMP A   #$'D      TESTS IF SIZE CHR EXISTS.
00708 FE11 27 22          BEQ     DIRECT    SKIPS IF "D"COMND EXISTS.
00709                    *
00710                    *
00711                    *
00712                    * FOLLOWING GETS THE NEXT LINE.
00713                    *
00714 FE13 DE 02   ASM4   LDX     TMP1      LOADS START OF LINE IN
00715 FE15 08      ASM4A  INX               ORDER TO FIND NEXT LINE.
00716 FE16 A6 00          LDA A   0,X       LOADS CHAR FROM SORCE LINE.
00717 FE18 81 60          CMP A   #$60      TESTS FOR A CARRAGE RETURN.
00718 FE1A 26 F9          BNE     ASM4A     SKIPS BAK UNTIL C. R.  FOUND.
00719 FE1C 08             INX               POINTS TO FIRST LINE CHAR.
00720 FE1D 20 8B          BRA     ASM1A     GOES BACK TO ASSM.  NEXT LINE
00721                    *
00722                    *
00723                    *
00724                    * THE FOLLOWING ARE SUBROUTINES USED BY THE MAIN CODE.
00725                    *
00726                    *
00727 FE1F 8D 06   INDADR BSR     SYMPTR    GETS CONTENTS OF
00728 FE21 EE 00          LDX     0,X       SYMBOL LOCATION.
00729 FE23 DF 2A          STX     ONDVAL    STORES AS OPERAND.
00730 FE25 20 9B          BRA     SYMBL     RETURNS TO FIX LABEL VALUE.
00731                    *
00732 FE27 48      SYMPTR ASL A   -         MULT LABEL BY 2 TO FORM
00733 FE28 5F             CLR B   -         POINTR INTO SYMBOL TABLE.
00734 FE29 97 01   LOADX  STA A   TMP+1     LOADS POINTER INTO THE
00735 FE2B D7 00          STA B   TMP       SYMBOL TABLE INTO X.
00736 FE2D DE 00          LDX     TMP       RETURNS TO CALLER.
00737 FE2F 39             RTS               RETURNS.
0
```

```
00739 FE30 D6 2A   EXTEND LDA B  ONDVAL    STORES HI BYTE OF OPERAND
00740 FE32 E7 00          STA B  0,X       INTO OBJECT CODE.
00741 FE34 08             INX              INC PC TO POINT TO NXT WD.
00742 FE35 96 2B   DIRECT LDA A  ONDVAL+1  STORES LO BYTE OF OPERAND
00743 FE37 A7 00          STA A  0,X       INTO OBJECT CODE.
00744 FE39 08             INX              INC & SAVE P.C. TO POINT TO
00745 FE3A DF 40          STX    PCVAL     NEXT BYTE.
00746 FE3C 20 D5          BRA    ASM4      GOES TO WORK ON NEXT LINE.
00747                *
00748 FE3E 08      RELTIV INX              INCREMENT P.C. PTR TO POINT
00749 FE3F DF 40          STX    PCVAL     TO NXT BYTE & SAVE P.C..
00750 FE41 96 2B          LDA A  ONDVAL+1  LOADS LO BYTE OF OPERAND.
00751 FE43 90 41          SUB A  PCVAL+1   FORMS RELATIVE OFFSET.
00752 FE45 09             DEX              INSERTS RELATIVE BYTE INTO
00753 FE46 A7 00          STA A  0,X       OBJECT CODE.
00754 FE48 20 C9          BRA    ASM4      GOES TO ASSMBL NEXT LINE.
00755                *
00756                *       END OF THE ASSEMBLER PROGRAM.
00757                *
```



```
00759                     END
TOTAL ERRORS 00000
```

```
00081                 *            DEBUGGER
00082                 *
00083                 *     THE DEBUGGER FOR THE PDS SYSTEM WAS DESIGNED TO
00084                 * PROVIDE A VERSATILE TOOL FOR USE IN PROGRAM TESTING AND
00085                 * DEBUGGING.  IT ALLOWS FOR BREAKPOINTS, MINI-ASSEMBLER
00086                 * SYMBOL TABLE REFERENCING, STACK MANIPULATION AND
00087                 * INPUT IN EITHER HEXADECIMAL, OCTAL OR DECIMAL.
00088                 * THE DEBUGGER PRINTS A PROMPT CHARACTER ">" ON EVERY NEW
00089                 * LINE.  AN INSTRUCTION CAN BE TYPED IN WHENEVER THE CURSOR
00090                 * IS BLINKING, EXCEPT WHEN A NUMBER IS BEING TYPED IN.
00091                 * THE DEBUGGER CALLS THE EDITOR WHENEVER A NUMBER IS TO BE
00092                 * INPUT, SO CORRECTIONS CAN BE MADE IF THE WRONG DIGIT IS
00093                 * TYPED IN.  THE POINTER "PCVAL" POINTS TO THE CURRENTLY
00094                 * OPENED BYTE LOCATION.  THE DEBUGGER OPERATES ON WHATEVER
00095                 * BYTE IS POINTED TO BY PCVAL.  FOR FURTHER DETAILS SEE THE
00096                 * SECTION ON THE DEBUGGER IN THE OPERATORS REFERENCE MANUAL.
00097                 *
00098                 *     THE DEBUGGER IS IMPLEMENTED BY A SMALL ROUTINE TO SET
00099                 * UP ENTRY (DEBUG) AND A LARGE ROUTINE WHICH DOES A RANGE
00100                 * COMPARE TO FIND THE PROPER COMMAND AND THEN EXECUTES THE
00101                 * COMMAND (RUNBUG).  NOTE THAT SINCE COMMANDS ARE
00102                 * DIFFERENTIATED BY RANGE, ANY KEY STRUCK WILL PRODUCE
00103                 * A COMMAND EXECUTION, SUCH AS A "," BEING INTERPRETED
00104                 * AS A "+" COMMAND.
00105                 *
00106                 *
00107                 *
00108                 * COMMANDS:
00109                 *
00110           ↑M    * "C.R." LINE - PRINTS ">" OUT ON A NEW LINE.
00111                 * " " CHANGE - THE SPACE COMND CHANGES CONTENTS FROM Y TO Z.
00112       ! to 9    * "+" OPNNXT - OPENS NEXT LOCATION.
00113    - to about   * "-" OPNPRE - OPENS PREVIOUS LOCATION.
00114        ↑@ ↑A    * "↑B" BRKSET - SETS A BREAKPOINT AT THE OPENED LOCATION.
00115                 * "↑C" CLRBRK - CLEARS BRKPOINT.  MUST BE DONE BEFORE EXIT.
00116           ↑D ↑F * "↑E" EXIT - PERFORM RTI - EXECUTE AT BRKPOINT LOCATION.
00117                 * "↑G" GOLOCN - STARTS EXECUTION AT OPENED LOCATION.
00118 BS TAB LF ↑H ↑I ↑K ↑L * "↑J" JUMP   - JUMP TO USERS SUBROUTINE.
00119           ↑N ↑P ↑Q * "↑O" OPNLOC - OPENS LOCATION THAT IS TYPED IN AFTER "O".
00120                 * "↑R" OPNREG - OPENS THE TOP-OF-STACK LOCATION.
00121                 * "↑S" SETSTK - SETS THE STACK TO THE OPENED LOCATION.
00122                 * "↑T" OPNTBL - OPENS LOCATION IN SYMBOL TABLE OF NEXT CHR.
00123 ↑U-↑Z (especially CAN ESC [ \ ] ^ del) * "↑X" GOEXEC - EXITS THE DEBUGGER - GOES BACK TO EXEC.
00124                 *
00125                 *
00126                 *
00127                 * SUBROUTINES:
00128                 *
00129                 * INPCHR - INPUTS A CHAR.  INTO A AND PRINTS IT.
00130                 * INPNUM - INPUTS A NUMBER INTO B-A FROM THE KEYBOARD.
00131                 * PNTBYT - PRINTS ACC A AS 2 HEX DIGITS ON THE SCREEN.
00132                 * PNTDIG - PRINTS B-A AS 4 HEX DIGITS ON THE SCREEN.
00133                 * NEWLIN - PRINTS A C.R.  AND A ">" ON THE SCREEN.
00134                 * DSPADR - PRINTS BYTE ADDRESS (XXXX) AND BYTE CONTENTS (YY)
00135                 *   AS >XXXX YY ON THE SCREEN.
00136                 *
```

```
00138 FE4A                    ORG    $FE4A
00139                 *
00140                 *
00141                 *
00142                 *       FOLLOWING IS LOCATION OF ENTRY OF THE BRKPT VECTOR.
00143                 *
00144 FE4A 30         BKENTR  TSX                    INDEX GETS STACK POINTER.
00145 FE4B E6 05              LDA B  5,X             LOADS HI RETURN ADDRESS.
00146 FE4D A6 06              LDA A  6,X             LOADS LOW BYTE OF ADDRESS.
00147 FE4F 80 01              SUB A  #1              SUB 1 FROM RETURN ADDRESS.
00148 FE51 C2 00              SBC B  #0              .
00149 FE53 E7 05              STA B  5,X             RESTORES RETURN ADDR. TO
00150 FE55 A7 06              STA A  6,X             THE BREAKPOINT LOCATION.
00151 FE57 20 0B              BRA    DEBUG           GOES TO THE DEBUGGER.
00152                 *
00153                 *
00154 FE59 81 0D      LINE    CMP A  #$0D            TESTS FOR A C.R. (LINE).
00155 FE5B 2D 7D              BLT    JMPLCN          GOES TO 'JSR' (↑J) ROUTNE.
00156 FE5D 2E 4E              BGT    OPNREG          SKIPS FOR NEXT (↑R) TESTS.
00157 FE5F BD FCCB            JSR    SUB32           MOVES CURSOR UP ONE LINE.
00158 FE62 31         POPLIN  INS                    CLEANS UP STACK FOR
00159 FE63 31                 INS                    DISPLAY OF C.R. >.
00160                 *
00161                 *
00162 FE64 8D 26      DEBUG   BSR    NEWLIN          PRINTS "C.R.>".
00163 FE66 8D 09      DBUG1   BSR    INPCHR          READS IN COMMAND.
00164 FE68 BD FCA5            JSR    RTARRO          INSERT BLANK.
00165 FE6B DE 40              LDX    PCVAL           LOADS CURRENTLY OPENED LOC.
00166 FE6D 8D 09              BSR    RUNBUG          EXECUTES DEBUG COMMAND.
00167 FE6F 20 F5              BRA    DBUG1           GOES BACK FOR NEXT COMND.
00168                 *
00169                 *
00170 FE71 BD FC4A    INPCHR  JSR    GETCHR          READ IN CHAR INTO A.
00171 FE74 BD FCBC            JSR    PUTCHR          DISPLAYS CHARACTER.
00172 FE77 39                 RTS                    RETURNS TO CALLER.
00173                 *
00174                 *
00175                 *
00176 FE78 81 03      RUNBUG  CMP A  #$03            TESTS FOR A "↑C" COMMAND.
00177 FE7A 2D 25              BLT    BRKSET          SKIPS IF A "↑B" COMMAND.
00178 FE7C 2E 45              BGT    EXIT            SKIPS FOR NEXT COMND TEST.
00179 FE7E DE 38      CLRBRK  LDX    BRKADR          GETS ADDRESS OF BREAKPOINT.
00180 FE80 96 2E              LDA A  BRKSAV          LOADS ORIG BYTE CONTENTS.
00181 FE82 A7 00              STA A  0,X             RESTORES BYTE DATA.
00182 FE84 20 66              BRA    DSPADR          GOES TO OPEN THE LOCATION.
00183                 *
00184                 *
00185 FE86 81 20      CHANGE  CMP A  #$ '            TESTS OFR A SPACE COMND.
00186 FE88 2D 12              BLT    EXECTV          SKIPS TO EXIT BACK TO EXEC.
00187 FE8A 2E 09              BGT    OPNPRE          SKIPS FOR OTHER CMND TESTS.
00188 FE8C 8D 56      SPACE   BSR    INPNUM          INPUTS NEW BYTE CONTENTS.
00189 FE8E DE 40              LDX    PCVAL           LOADS OPENED BYTE LOCATION.
00190 FE90 A7 00              STA A  0,X             STORES NEW BYTE CONTENTS.
00191                 *
00192                 *
00193 FE92 08         OPNNXT  INX                    FORMS NEXT LOCATION ADDRES.
00194 FE93 20 57              BRA    DSPADR          GOES TO OPEN LOCATION BYTE.
```

```
00196 FE95 81 2D    OPNPRE CMP A  #$'-      TESTS FOR A "-" COMMAND.
00197 FE97 2D F9           BLT    OPNNXT    SKIPS FOR A "+" COMMAND.
00198 FE99 09              DEX              FORMS PREV. LOCATION ADDR.
00199 FE9A 20 50           BRA    DSPADR    GOES TO OPEN THE LOCATION.
00200                *
00201                *
00202 FE9C 31       EXECTV INS              CLEANS UP THE STACK.
00203 FE9D 31              INS
00204 FE9E 7E FC14         JMP    EXEC      RETURNS TO THE EXECUTIVE.
00205                *
00206                *
00207 FEA1 A6 00    BRKSET LDA A  0,X       LOADS DATA OF OPNED LOCATN.
00208 FEA3 97 2E           STA A  BRKSAV     SAVES DATA OF OPNED BYTE.
00209 FEA5 DF 30           STX    BRKADR     SAVES ADDR. OF BREAKPOINT.
00210 FEA7 86 3F           LDA A  #$3F       LOADS SOFTWARE INTUP COMND.
00211 FEA9 A7 00           STA A  0,X        SETS AN SWI AT OPNED BYTE.
00212 FEAB 20 B5           BRA    POPLIN     GOES TO NEXT LINE FOR COMND.
00213                *
00214                *
00215 FEAD 81 12    OPNREG CMP A  #$12       TESTS FOR "↑R" (STACK TOP).
00216 FEAF 2D 0A           BLT    OPNLOC     GOES TO OPEN A LOCATION.
00217 FEB1 2E 19           BGT    OPNTBL     SKIPS FOR NEXT TEST (↑T).
00218 FEB3 30              TSX              OPENS TOP-OF-STACK.
00219 FEB4 08              INX              PCVAL GETS STACK POINTER.
00220 FEB5 08              INX              (CLEANS UP THE STACK).
00221 FEB6 20 34           BRA    DSPADR     GOES TO DISPLAY THE T-O-S.
00222                *
00223                *
00224 FEB8 35       SETSTK TXS              STACK POINTER GETS PCVAL.
00225 FEB9 20 A9           BRA    DEBUG      RETURNS TO INPUT COMMAND.
00226                *
00227                *
00228 FEBB 8D 27    OPNLOC BSR    INPNUM     LOADS A 16 BIT NUMBER.
00229 FEBD D7 40    OPNLC1 STA B  PCVAL      STORES NEWLY OPENED
00230 FEBF 97 41           STA A  PCVAL+1    LOCATION ADDRESS.
00231 FEC1 20 2B           BRA    DSPAD1     DISPLAYS CONTNTS OF LOCATN.
00232                *
00233                *
00234 FEC3 81 07    EXIT   CMP A  #$07       TESTS IF AN EXIT (↑E) COMD.
00235 FEC5 27 11           BEQ    GOLOCN     SKIPS FOR THE "GO" COMMAND.
00236 FEC7 2E 90           BGT    LINE       SKIPS FOR NEXT COMMD TEST.
00237 FEC9 31              INS              CLEARS UP THE STACK.
00238 FECA 31              INS
00239 FECB 3B              RTI              RETURNS FROM BREAKPOINT.
00240                *
00241                *
00242 FECC 81 14    OPNTBL CMP A  #$14       TESTS IF A "↑T" (TABLE).
00243 FECE 2D E8           BLT    SETSTK     GOES TO SET STACK PTR (↑S).
00244 FED0 2E B4           BGT    CHANGE     SKIPS FOR NEXT TEST (SPACE).
00245 FED2 8D 2D           BSR    INPCHR     LOADS A WITH SYMBOL (LABL).
00246 FED4 48              ASL A             ALIGNS ADDRESS FOR
00247 FED5 5F              CLR B             SYMBOL TABLE ENTRY.
00248 FED6 20 E5           BRA    OPNLC1     SAVES AND DISPLAYS ADDRESS.
00249                *
00250                *
00251 FED8 31       GOLOCN INS              CLEANS UP THE STACK.
00252 FED9 31              INS
00253 FEDA 6E 00    JMPLCN JMP    0,X        JUMPS TO USERS PROGRAM.
```

```
00255                    *
00256                    * FOLLOWING ARE SUBROUTINES USED BY THE DEBUGGER.
00257                    *
00258 FEDC 86 0D    NEWLIN LDA A  #$0D      LOADS A CARRIAGE RETURN.
00259 FEDE 8D 3E           BSR    PNTBF1    PRINTS A CARRIAGE RETURN.
00260 FEE0 86 3E           LDA A  #'>       LOADS A PROMPT CHARACTER.
00261 FEE2 20 3A           BRA    PNTBF1    DISPLAYS PROMPTER CHAR.
00262                    *
00263                    *
00264                    * FOLLOWING INPUTS A 16 BIT NUMBER INTO THEE BA REGISTER.
00265                    *
00266 FEE4 BD FC75 INPNUM JSR     EDITIN    INPUTS A STRING OF DIGITS.
00267 FEE7 DE 24           LDX    SCNPIR    LOADS ADDR. OF FIRST DIGIT.
00268 FEE9 8D 37           BSR    ASCBIN    CONVERTS TO BINARY # IN BA.
00269 FEEB 39              RTS              RETURNS TO CALLER.
00270                    *
00271                    *
00272                    * FOLLOWING DISPLAYS THE LOCATION ADDR. & CONTENTS.
00273                    *
00274 FEEC DF 40    DSPADR STX     PCVAL     SAVES OPENED LOCATION ADDR.
00275 FEEE 8D EC    DSPAD1 BSR     NEWLIN    PRINTS A "C.R." AND ">".
00276 FEF0 8D 0A           BSR    PNTDIG    PRINTS OUT "PCVAL" IN HEX.
00277 FEF2 BD FCA5         JSR    RTARRO    PRINTS A SPACE.
00278 FEF5 DE 40           LDX    PCVAL     LOADS PTR. TO OPEND LOC.
00279 FEF7 A6 00           LDA A  0,X       LOADS DATA FROM LOCATN.
00280 FEF9 8D 07           BSR    PNTBYT    PRINTS DATA IN HEX FORMAT.
00281 FEFB 39              RTS              RETURNS TO INPUT COMMAND.
00282                    *
00283                    *
00284 FEFC 96 40    PNTDIG LDA A  PCVAL     PRINTS THE 2 HI HEX DIGITS
00285 FEFE 8D 02           BSR    PNTBYT    OF OPENED ADDRESS.
00286 FF00 96 41           LDA A  PCVAL+1   PRINTS OUT 2 LOW HEX DIGITS.
00287                    *
00288                    *
00289                    * FOLLOWING PRINTS OUT 2 HEX DIGITS.
00290                    *
00291 FF02 CE 0010 PNTBYT LDX     #16       LOADS 16 FOR BASE.
00292 FF05 DF 04           STX    ARB       STORES FOR CONVERSION.
00293 FF07 5F              CLR B  -         CLEARS HI 2 DIGITS.
00294 FF08 CE 0035         LDX    #IOBUFF   LOADS OUTPUT BUFF ADDRESS.
00295                    *
00296                    *
00297                    * FOLLOWING CONVERTS BYTE TO HEX WITH LEADING ZEROS.
00298                    *
00299 FF0B D7 36    CONVRT STA B  IOBUFF+1  CLERS BYTE FOR SECOND DIGIT..
00300 FF0D BD FF64         JSR    BINASC    CONVERTS TO ASCII DIGITS.
00301 FF10 96 35           LDA A  IOBUFF    LOADS HI DIGIT.
00302 FF12 D6 36           LDA B  IOBUFF+1  TESTS BOTH DIGITS CONVTD.
00303 FF14 26 04           BNE    PNTBUF    SSKIPS IF BOTH DIGITS CONVTD00304 FF16
 97 36          STA A  IOBUFF+1 SETS UP LOW DIGIT.
00305 FF18 86 30           LDA A  #$'0      HIGH DIGIT GETS A "0".
00306                    *
00307 FF1A 8D 02    PNTBUF BSR     PNTBF1    PRINTS OUT HI DIGIT.
00308 FF1C 96 36           LDA A  IOBUFF+1  LOADS LOW DIGI
00309 FF1E BD FCBC PNTBF1  JSR     PUTCHR    DISPLAYS CCHARACTER.
00310 FF21 39              RTS              RETURNS TO CALLING PROGRAM.
00311                    *
00312                    *       END OF DEBUGGER
```

```
00314                *      UTILITY PROGRAMS
00315                *
00316                *
00317                *
00318                *
00319                *           ASCII TO BINARY CONVERSION.
00320                *
00321                *     THE ASCII TO BINARY ROUTINE CONVERTS FROM AN ASCII
00322                * NUMBER STRING POINTED TO BY X TO AN UNSIGNED 16 BIT
00323                * BINARY NUMBER IN BA (ACC B HAS THE HI BYTE, ACC A HAS
00324                * THE LO BYTE).  THE ASCII STRING IS TERMINATED BY A NON
00325                * HEXADECIMAL CHARACTER.  UPON EXITING, THE INDEX REGISTER
00326                * WILL POINT TO THE NEXT CHARACTER AFTER THE NUMBER
00327                * STRING.  THE BASE OF THE NUMBER STRING IS PASSED TO
00328                * THE ROUTINE IN ARA (ARA IS THE ARITHMETIC REGESTER A
00329                * LOCATED IN BYTES 06 AND 07 OF LOW MEMORY).  IF THE
00330                * ROUTINE IS ENTERED WITH A KNOWN BASE, PUT THE BASE
00331                * (BETWEEN 2 AND 16) IN ARA AND ENTER THE ROUTINE AT
00332                * THE ENTRY POINT ENTR2.
00333                *
00334                *
00335                *
00336                *     CONVERSION FORMULA:
00337                * ASCII NUMBBER STRING X[4],X[3],X[2],X[1] IN
00338                *BASE Y;
00339                * BINARY NUMBER =
00340                *     X[4]*Y↑3+X[3]*Y↑2+X[2]*Y↑1+X[1]*Y↑0          OR
00341                * BINARY NUMBER =
00342                *     (((0*Y+X[4])*Y+X[3])*Y+X[2])*Y+X[1]
00343                *WHERE ↑ IS THE EXPONENT OPERATOR,
00344                * X IS A CHARACTER & Y IS THE BASE.
00345                *
00346                *
00347                *
00348                *     ALGORITHM:
00349                *ASCBIN:  FORM THE BASE IN ARA BASED ON THE FIRST CHAR.
00350                *  OF THE NUMBER STRING; INCREMENT CHAR. PTR. IN X;
00351                *ENTR2:  NUMBER (IN BA) GETS 0;
00352                *NXTCHR:  IF THE CURRENT CHAR. POINTED TO BY X IS NOT A
00353                *  DIGIT THEN EXIT ELSE INCREMT CHARACTER PTR IN INDEX;
00354                *  CONVERT DIGIT TO BINARY;
00355                *  NUMBER GETS NUMBER * BASE;
00356                *  NUMBER GETS NUMBER + DIGIT;
00357                *  GO TO OPERATE ON THE NEXT DIGIT (NXTCHR);
00358                *
```

```
00360 FF22 A6 00   ASCBIN LDA A  0,X      GETS CHR TO FORM BASE.
00361 FF24 81 2E           CMP A  #$'.     TESTS FOR DECML STRNG.
00362 FF26 2D 06           BLT    OCT      SKIPS IF BASE 8 (*).
00363 FF28 2E 09           BGT    HEX      SKIPS IF BASE 16.
00364 FF2A 86 0A           LDA A  #10      LOADS BASE 10 FOR CONVERSN.
00365 FF2C 20 02           BRA    ASC1     SKIPS TO INC. TEXT POINTR.
00366 FF2E 86 08   OCT     LDA A  #8       LOADS BASE 8 FOR CONVERSION.
00367 FF30 08      ASC1    INX             INCREMENT PTR TO NEXT CHAR.
00368 FF31 20 02           BRA    ASC2     SKIPS TO SAVE BASE.
00369 FF33 86 10   HEX     LDA A  #16      LOADS BASE 16 FOR CONVERN.
00370 FF35 97 07   ASC2    STA A  AR0      SAVES BASE IN BASE#.
00371              *
00372              *
00373 FF37 5F      ENTR2   CLR B  NUMBER   GETS 0.
00374 FF38 37              PSH B  -        (LOW NUMBER ON STACK).
00375 FF39 D7 06           STA B  AR1      CLEARS HI OF BASE.
00376 FF3B A6 00   NXTCHR  LDA A  0,X      GETS CHAR TO CONVERT.
00377 FF3D 08              INX             INC TO NEXT CHARACTER
00378 FF3E 81 30           CMP A  #$'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.
```

```
00401                    *          BINARY TO ASCII
00402                    *
00403                    *
00404                    *     THE BINARY TO ASCII CONVERSION ROUTINE CONVERTS
00405                    * A 16 BIT BINARY NUMBER IN THE BA REGISTR (REG B & REG A)
00406                    * TO A STRING OF ASCII DIGITS.  THE ASCII STRING CAN BE IN
00407                    * ANY BASE FROM BASE 2 THROUGH BASE 41.  THE VALUE OF THE
00408                    * BASE IS LOCATED IN THE ARITHMETIC PSEUDO-REGISTER ARB
00409                    * (ARB IS LOOCATED IN BYTE AR3 [LOC 4] AND AR2 [LOC 5]).
00410                    * WHEN THE ROUTINE IS ENTERED, THE POINTER TO THE OUTPUT
00411                    * LOCATION IS PASSED IN THE INDEX REG.  WHEN THE ROUTINE
00412                    * EXITS, THE INDEX POINTS TO THE LAST DIGIT IN THE STRING
00413                    * PLUS ONE.
00414                    *     CONVERSION IS DONE BY THE METHOD OF REPEATED
00415                    * DIVISION.  THE LOW ORDER DIGIT IS FORMED FIRST.  THE
00416                    * DIGITS ARE THEN PLACED ON THE STACK UNTIL CONVERSION IS
00417                    * COMPLETED.  THE DIGITS ARE THEN POPPED OFF THE STACK
00418                    * AND PLACED IN THE OUTPUT STRING.  THE TOP-OF-STACK IS
00419                    * INITIALIZED TO HEX FF TO TELL WHEN ALL THE DIGITS
00420                    * HAVE BEEN POPPED OFF THE STACK.  AFTER THE DIVISION, THE
00421                    * DIGIT (THE REMAINDER OF THE DIVISION OPERATION)
00422                    * IS LOCATED IN THE AR 0 PART OF ARA (BYTE 7).  WHEN
00423                    * THE QUOTIENT OF THE DIVISION IS 0, THEN THE CONVERSION
00424                    * IS COMPLEATED.
00425                    *
00426                    *
00427                    *
00428 FF64 DF 00    BINASC STX    TMP       SAVES OUTPUT POINTER.
00429 FF66 34               DES             /SETS THE TOP-OF-STACK TO
00430 FF67 30               TSX             /ALL ONES TO TELL END OF
00431 FF68 6F 00            CLR    0,X      /CHAR STRING (LAST CHAR IS
00432 FF6A 63 00            COM    0,X      /PUT ON STACK FIRST).
00433 FF6C DE 04    BIN1    LDX    ARB      RESTORES DIVISOR (BASE).
00434 FF6E DF 06            STX    ARA
00435 FF70 8D 3D            BSR    DIVIDE   * QUOTIENT IN BA GETS THE
00436                    * REMAINDER OF # TO BE CONVERTED; REMAINDER IN ARA GETS
00437                    * THE LOW ORDER DIGIT.
00438 FF72 97 02            STA A  TMP1     SAVES A OF BA.
00439 FF74 96 07            LDA A  AR0      LOAD DIGIT (REMAINDER).
00440 FF76 36               PSH A  -        STACK DIGIT (REVERSE ORDER). G
00441 FF77 96 02            LDA A  TMP1     RESTORES A OF BA.
00442 FF79 4D               TST A  -        /TESTS IF QUOTIENT IS = 0
00443 FF7A 26 F0            BNE    BIN1     /(SIGNIFYING THAT
00444 FF7C 5D               TST B  -        /THE CONVERRSION
00445 FF7D 26 ED            BNE    BIN1     /IS DONE).
00446                    *
00447 FF7F DE 00    BINSTR LDX    TMP       RESTORES OUTPUT POINTER.
00448 FF81 32       BIN3    PUL A  -        UNSTACK A DIGIT.
00449 FF82 4D               TST A  -        TESTS IF NEG (END?).
00450 FF83 2A 01            BPL    BIN4     SKIPS IF A DIGIT.
00451 FF85 39               RTS             EXITS FROM SUBROUTINE.
00452 FF86 81 09    BIN4    CMP A  #9       TESTS IF RESULT IS HEX.
00453 FF88 2F 02            BLE    BIN5     SKIPS IF DIGIT NOT HEX.
00454 FF8A 8B 07            ADD A  #7       FORMS HEX VALUE OF DIGIT.
00455 FF8C 8B 30    BIN5    ADD A  #$'0     FORMS DECIMAL CHARACTER.
00456 FF8E A7 00            STA A  0,X      OUTPUTS CHARACTER.
00457 FF90 08               INX             POINTS TO NEXT CHARACTER.
00458 FF91 20 EE            BRA    BIN3     GOES BACK FOR NEXT DIGIT.
```

```
00460                    *          MULTIPLY ROUTINE
00461                    *
00462                    *     THE MULTIPLY ROUTINE MULTIPLIES TWO 16 BIT BINARY
00463                    * NUMBERS TOGETHER TO PRODUCE A 16 BIT RESULT.  THE BA
00464                    * REGISTERS AND ARA (BYTES 6 & 7) REGISTER ARE USED.
00465                    * THE CONTENTS OF ARA ARE UNCHANGED UPON PROGRAM EXIT.
00466                    *
00467                    *
00468                    *  BA GETS BA * ARA
00469                    *
00470                    *
00471                    *     MULTIPLYING IS ACCOMPLISHED BY REPEATED ADDITIONS
00472                    * OF ONE OF THE OPERATORS (OPERATOR ARA) INTO THE RESULT.
00473                    * THE RESULT STARTS OUT WITH A ZERO VALUE AND IS SHIFTED
00474                    * OVER ONE AFTER EACH ADDITION.  THE HIGHEST ORDER VALUE
00475                    * IS ADDED IN FIRST AND THEN, GOING TO THE RIGHT,
00476                    * (THUS SHIFTING THE ANSWER LEFT ONE TO BRING IN THE NEXT
00477                    * RIGHTMOST DIGIT) GETTING THE NEXT LOWERMOST SIGNIFICANT
00478                    * DIGIT.  THE NEXT RIGHTMOST BIT OF THE OTHER OPERAND
00479                    * (THE ONE ORIGINALLY IN BA) IS TESTED, AND IF ONE,
00480                    * ANOTHER ADDITION TAKES PLACE.  THIS IS REPEATED UNTIL
00481                    * THE FINAL SUM IS FORMED.
00482                    *
00483                    *
00484                    *
00485                    *  MULTIPLY ALGORITHM:
00486                    *
00487                    *MULT:  STACK BA; BA GETS 0; SET COUNT VALUE TO 16;
00488                    *MUL1:  SHIFT BA LEFT 1;
00489                    *  SHIFT LEFT ORIG BA VALUE ON STACK INTO CARRY;
00490                    *  IF CARRY = 0 THEN GO TO MUL2
00491                    *  BA GETS BA + ARA;
00492                    *MUL2:  DECREMENT COUNT;
00493                    *  IF COUNT # 0 THEN GO TO MUL1 ELSE EXIT.
00494                    *
00495                    *
00496                    *
00497
00498 FF93 36            MULT   PSH A   -          PUTS THE ORIGINAL CONTENTS
00499 FF94 37                   PSH B   -          OF BA ONTO THE STACK.
00500 FF95 86 10                LDA A   #16        LOADS COUNT VALUE
00501 FF97 36                   PSH A   -          ONTO THE STACK.
00502 FF98 4F                   CLR A   -          BA GETS ZEROED.
00503 FF99 5F                   CLR B
00504 FF9A 30                   TSX                SET INDEX TO STACK.
00505 FF9B 48            MUL1   ASL A   -          SHIFT LEFT BA.
00506 FF9C 59                   ROL B
00507 FF9D 68 02                ASL    2,X         SHIFTS ORIG. BA OPERAND
00508 FF9F 69 01                ROL    1,X         ONE LEFT INTO CARRY.
00509 FFA1 24 04                BCC    MUL2        SKIPS ADDING IF CARRY = 0.
00510 FFA3 9B 07                ADD A  AR0         BA GETS BA + ARA.
00511 FFA5 D9 06                ADC B  AR1
00512 FFA7 6A 00         MUL2   DEC    0,X         TESTS IF DONE.
00513 FFA9 26 F0                BNE    MUL1        GOES BACK IF NOT DONE.
00514 FFAB 31                   INS                CLEANS UP THE STACK.
00515 FFAC 31                   INS
00516 FFAD 31                   INS
00517 FFAE 39                   RTS                EXITS ROUTINE.
```

```
00519                    *          DIVIDE ROUTINE
00520                    *
00521                    *     THE DIVIDE SUBROUTINE DIVIDES THE 16 BIT NUMBER
00522                    * IN THE BA REGISTERS BY THE 16 BIT NUMBER IN THE PSEUDO
00523                    * REGESTER ARA (LOCATED IN BYTES 6 & 7).  UPON EXITING,
00524                    * BA WILL CONTAIN THE QUOTIENT OF THE DIVISION AND ARA
00525                    * WILL CONTAIN THE REMAINDER.  THE DIVIDEND BA IS DIVIDED
00526                    * BY THE DIVISOR ARA (I.E.  BA/ARA ).
00527                    *
00528                    *
00529                    *
00530                    *
00531                    *  DIVIDE ALGORITHM:
00532                    *
00533                    *DIVIDE:  X3,4 GETS BA (BA IS PUT ON THE STACK);
00534                    *  X1,2 GETS THE 16 BIT ARA VALUE (ARA PUT ON THE STACK);
00535                    *  COUNT GETS 1 + THE NUMBER OF NONSIGNIFICANT BITS IN
00536                    * THE DIVISOR [LEFT JUSTIFY X1,2 TO FIRST 1 BIT, COUNT
00537                    * GETS 1 + THE # OF LEADING ZEROS IN X1,2] [COUNT WILL BE
00538                    * FROM 1 TO 17];
00539                    *  BA GETS X3,4 [RESTORES BA];
00540                    *  X3,4 GETS 0 [INITIALIZES THE QUOTIENT];
00541                    *DIV3:  BA GETS BA - X1,2;
00542                    *  IF THERE WAS A BORROW [I.E. DIVIDEND IN BA < DIVISOR
00543                    * IN X1,2 (ARA)] THEN BA GETS BA + X1,2 [ORIGINAL BA
00544                    * VALUE RESTORED] & CARRY CLEARED ELSE CARRY IS SET;
00545                    *DIV5:  X3,4 [QUOTIENT] GETS CARRY LEFT SHIFTED IN;
00546                    *  X1,2 [DIVISOR] GETS SHIFTED RIGHT ONE PLACE WITH ZERO
00547                    * FILLED IN FROM THE LEFT SIDE;
00548                    *  DECREMENT COUNT;
00549                    *  EXIT IF DONE ELSE GO TO DIV3;
00550                    *
00551                    *
00552                    *
00553                    *
00554                    *
00555 FFAF 36            DIVIDE PSH A  -        LOADS DIVIDEND INTO X3,4.
00556 FFB0 37                   PSH B
00557 FFB1 96 06                LDA A  AR1     LOADS DIVISOR FROM ARA.
00558 FFB3 D6 07                LDA B  AR0
00559 FFB5 37                   PSH B  -       PUTS DIVISOR INTO X1,2.
00560 FFB6 36                   PSH A
00561 FFB7 34                   DES            SET UP SPACE FOR COUNT.
00562 FFB8 30                   TSX            INDEX GETS STACK POINTER.
00563 FFB9 86 01                LDA A  #1      INITIALIZE COUNT.
00564 FFBB 6D 01                TST    1,X     TESTS FOR HI DIVISR BIT ON.
00565 FFBD 2B 0B                BMI    DIV2    SKIPS IF ON.
00566 FFBF 4C            DIV1   INC A  -       COUNTS LEADING ZEROS.
00567 FFC0 68 02                ASL    2,X     LEFT JUSTIFIES X1,2.
00568 FFC2 69 01                ROL    1,X
00569 FFC4 2B 04                BMI    DIV2    SKIPS IF NO LEADING ZERO.
00570 FFC6 81 11                CMP A  #17     TESTS FOR ALL ZERO DIVISOR.
00571 FFC8 26 F5                BNE    DIV1    GOES BACK IF BITS LEFT.
00572 FFCA A7 00         DIV2   STA A  0,X     SETS COUNTER.
00573 FFCC E6 03                LDA B  3,X     BA GETS ORIGINAL
00574 FFCE A6 04                LDA A  4,X     DIVIDEND VALUE.
00575 FFD0 6F 03                CLR    3,X     CLEARS X3,4 FOR FORMATION
00576 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 FFAE                *
00601   49                *
00602                     *
00603 FFF8 0104    IRQ    FDB    $0104    INTERRUPT REQUEST VECTOR.
00604 FFFA FE4F    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
```