11 February 1977

PACE ASSEMBLER - BASIC INTERPRETER

```
                        .TITLE  BASIC,'INTERPRETER  11 FEB 77'
                ;
                ; VERSION SELECT FLAGS:
0000 A DEBUG    =       0               ; INCLUDE DUMP COMMAND
0000 A ROM      =       0               ; ROM VERSION
0001 A BIG      =       1 ! ROM         ; INCLUDE DISK FEATURES
0001 A PACE     =       1
0000 A IMP16    =       1-PACE
8001 A LOPT     =       08001           ; BASIC LIST OPTION FOR .LIST
8001 A          .LIST   LOPT
;
;*******************************************************************************
;
;                       PACE BASIC INTERPRETER
;
;
;       N   N  AAA  TTTTT  III   000  N   N  AAA  L
;       N   N A   A   T     I   O   O N   N A   A L
;       NN  N A   A   T     I   O   O NN  N A   A L
;       N N N A   A   T     I   O   O N N N A   A L
;       N  NN AAAAA   T     I   O   O N  NN AAAAA L
;       N   N A   A   T     I   O   O N   N A   A L
;       N   N A   A   T    III   000  N   N A   A LLLLL
;
;
;                 BBBB   AAA   SSS   III   CCC
;                 B   B A   A S   S   I   C   C
;                 B   B A   A S       I   C
;                 BBBB  A   A  SSS    I   C
;                 B   B AAAAA     S   I   C
;                 B   B A   A S   S   I   C   C
;                 BBBB  A   A  SSS   III   CCC
;
;
;
;                           ** NOTICE **
;       'BASIC' IS A REGISTERED TRADEMARK OF DARTMOUTH COLLEGE.
;
;
;       THIS PROGRAM WAS WRITTEN BY BRUCE J. EDMUNDSON
;       (C) COPYRIGHT 1977, NATIONAL SEMICONDUCTOR CORP.
;
;*******************************************************************************
;
;       DEFINITIONS
;
0000 A AC0      =       0
0001 A AC1      =       1
0002 A AC2      =       2
0003 A AC3      =       3
0000 A SFL      =       0               ; STACK FULL
0001 A ZRO      =       1               ; AC0 = 0
0002 A POS      =       2               ; AC0 >= 0
0003 A BIT0     =       3               ; AC0(0) = 1
```

```
 55        0004 A BIT1     =       4           ; ACO(1) = 1
 56        0005 A NZRO     =       5           ; ACO NEQ 0
 57        000A A CRY      =       10          ; CARRY SET
 58        000B A NEG      =       11          ; ACO < 0
 59        000C A OVF      =       12          ; OVERFLOW SET
 60   ;
 61        0006 A OVFLW    =       6           ; OVERFLOW FLAG
 62        0007 A CARRY    =       7           ; CARRY FLAG
 63        0008 A LINK     =       8           ; LINK FLAG
 64        0027 A REMCHAR  =       ''''/256    ; REMARK CHARACTER
 65        002C A COMMA    =       ','/256
 66        003D A EQUAL    =       '='/256
 67        0020 A BLANK    =       ' '/256     ; RIGHT JUSTIFIED SPACE CHARACTER
 68        0020 A B        =       BLANK
 70        0000 A DUMP     =       0           ; NOT DEBUG: DUMP NOT INCLUDED
 72   ;
 73        7E3B A GETC     =       07E3B       ; IPC-16P FIRMWARE ADDRESSES
 74        7E44 A PUTC     =       07E44
 75        7E59 A GECO     =       07E59
 76        7ECC A INTEST   =       07ECC
 77        D404 A PGFR     =       0D404       ; DISK FIRMWARE ADDRESSES
 78        D406 A DRESET   =       0D406
 79        D40E A OPENR    =       0D40E
 80        D410 A OPENW    =       0D410
 81        D416 A CLOSE    =       0D416
 82        D422 A RWORD    =       0D422
 83        D424 A WWORD    =       0D424
 84        D430 A DERROR   =       0D430
 85   ;
 86   ;
 87                        .GLOBL  ENTRY,USER,PROT
 88   ;
 89   ;
 90   ;         *** BASE SECTOR VARIABLES ***
 91   ;
 92 0000                   .ASECT
 93 0000 9895 A           JMP     @START
 94 0001                  .=2
 95 0002 0427 T           .WORD   STKINT
 96 0003                  .=010
 97   ;
 98 0010          USER:   .=.+1   ; ADDRESS OF USER FUNCTION USR(X)
 99 0011          PROT:   .=.+1   ; POKE PROTECTION FLAG (INIT=1)
100 0012          XQTFLG: .=.+1   ; EXECUTE FLAG (EXEC STATEMENT)
101 0013          MODE:   .=.+1   ; 1=EDIT, 2=CMND, 4=PASS1, 8=PASS2
102 0014          CURLIN: .=.+1   ; CURRENT LINE NUMBER
103 0015          TRMODE:  =.+1   ; LINE EXEC TRACE MODE (1=ON)
104 0016          TXTMOD:  =.+1   ; 0=TTY INPUT; 1=TAPE/DISK; 2=AUTO
105 0017          BRKMOD: .=.+1   ; BREAK CAPABILITY: 0=OFF, 1=ON
106 0018          RUNFLG: .=.+1   ; RUN FLAG: 0=INPUT MODE, 1=RUN
107 0019          ERRFLG: .=.+1   ; ERROR FLAG: 1 TO INHIBIT PASS 2
108   ;
109   ;         I/O FLAGS
110   ;                 UNIT:   0=KB,  1=PT, 2=DISK
```

```
                ;               OUTDVC: 0=TTY, 1=PT, 2=HSP, 4=STRING, 8=DISK
                ;
001A            UNIT:   .=.+1   ; INPUT UNIT
001B            OUTDVX: .=.+1   ; DEFAULT OUTPUT UNIT
001C            OUTDVC: .=.+1   ; OUTPUT UNIT
001D            OUTCOL: .=.+1   ; OUTPUT COLUMN
001E            LINLEN: .=.+1   ; PRINT LINE LENGTH
001F            LZONE:  .=.+1   ; LAST PRINT ZONE
0020            ENDBUF: .=.+1   ; END OF INPUT BUFFER
0021            BRKFLG: .=.+1   ; FLAG TO INDICATE TTY KEY INTERRUPT
                ;
                ;       SOFTWARE STACK POINTERS
                ;
0022            STKPTR: .=.+1   ; HARDWARE STACK EXTENSION POINTER
0023            STPTR:  .=.+1   ; 0: REGISTER SAVE STACK
0024                    .=.+1   ; 1: SYNTAX ANALYZER STACK
0025                    .=.+2   ; 2,3: EXPRESSION EVALUATION STACKS
0027                    .=.+1   ; 4: USER FUNCTION STACK
0028                    .=.+1   ; 5: GOSUB STACK
0029                    .=.+1   ; 6: FOR-NEXT STACK
                ;
                ;       STACK INTERRUPT HANDLER SCRATCH LOCATIONS
                ;
002A            SK0:    .=.+1   ; REGISTER SAVE LOCATIONS
002B            SK1:    .=.+1
002C            SK2:    .=.+1
002D            SKRET:  .=.+1   ; RETURN ADDRESS
002E            STK5:   .=.+5   ; TEMP
                ;
                ;       EDITOR POINTERS
                ;
0033            ESTART: .=.+1   ; START OF EDIT STORAGE
0034            ECUR:   .=.+1   ; CURRENT LINE LOCATION
0035            EEND:   .=.+1   ; END OF EDIT STORAGE (LAST+1)
0036            ELEN:   .=.+1   ; LINE LENGTH (0=DELETE)
0037            NXTLIN: .=.+1   ; ADDRESS OF NEXT LINE
0038            LAST:   .=.+1   ; LAST USED ADDRESS IN MEMORY
0039            TOPRAM: .=.+1   ; LAST ADDRESS IN MEMORY
                ;
                ;       SCANNER VARIABLES
                ;
003A            NTYPE:  .=.+1   ; NUMBER TYPE EXPECTED: 0=CONST, 1=SEQNO
003B            STYPE:  .=.+1   ; STRING EXPECTED: 0=NORMAL,
                                ;     1=DATA/INPUT, 2=LINPUT
003C            SVSCAN: .=.+2   ; SAVE OF LSTART,COL
003E            LSTART: .=.+1   ; LINE START ADDRESS
003F            COL:    .=.+1   ; COLUMN NUMBER
0040            NEXTAD: .=.+1   ; ADDRESS OF NEXT LINE
0041            CLASS:  .=.+1   ; SEMANTIC TYPE
0042            SYMB:   .=.+1   ; SYMBOL VALUE
0043            SYMA:   .=.+1   ; SYMBOL ADDRESS
0044            PREC:   .=.+1   ; OPERATOR PRECEDENCE
0045            IDENT:  .=.+2   ; IDENTIFIER TEMP.
0047            VALUE:  .=.+2   ; FLT. PT.  VALUE
```

```
165 0049          EXPN:    .=.+1    ; EXPONENT
166 004A          DIGIT:   .=.+2    ; TEMP. FOR NUMBER SCANNING
167 004C          PLC:     .=.+2    ; FRACTIONAL PART PLACES
168 004E          ESIGN:   .=.+1    ; EXPONENT SIGN
169 004F          .PFLAG:  .=.+1    ; PERIOD FLAG, TEMP FOR ALLOCATOR,PSCAN
170       000F A NWDS      =        .-CLASS   ; # WORDS TO PRESET TO 0
171  ;
172  ;         SYNTAX ANALYZER RETURN ADDRESS
173  ;
174 0050          SYNRTN:  .=.+1
175  ;
176  ;         SYMBOL TABLE POINTERS
177  ;
178       0038 A SY1       =        LAST      ; HIGH ADDR
179 0051          SY2:     .=.+1              ; LOW - 1
180 0052          FNLINE:  .=.+1              ; LINE NUMBER OF FUNCTION
181                                           ;    (USED FOR DUMMY PARAMETERS)
182  ;
183  ;         DATA POINTERS
184  ;
185 0053          DSTART:  .=.+1              ; LINE POINTER
186 0054          DPTR:    .=.+1              ; COL POINTER
187       0067 A DTEMP     =        TEMP+2    ; COMMA FLAG
188       0068 A DSIGN     =        TEMP+3    ; +/1 SIGN
189       0069 A TYPFLG    =        TEMP+4    ; NUMERIC/STRING FLAG
190 0055          ANEXT:   .=.+1              ; NEXT LOC FOR ARRAYS/STRINGS
191 0056          TNEXT:   .=.+1              ; NEXT LOC FOR TEMP STORAGE
192  ;
193  ;         STRING PROCESSING VARIABLES
194  ;
195 0057          SSTART:  .=.+1    ; STRING BUFFER START
196 0058          SCOL:    .=.+1    ; STRING COLUMN (NEXT)
197 0059          MAXSTR:  .=.+1    ; MAX CHAR LENGTH OF CURRENT STRING
198 005A          NSTRC:   .=.+1    ; NUMBER CHAR PER STRING
199 005B          NSTRW:   .=.+1    ; NUMBER WORDS PER STRING
200  ;
201  ;         TEMPORARY FOR EXPRESSION EVALUATION
202  ;
203 005C          OP1:     .=.+2    ; FIRST OPERAND
204 005E          OP2:     .=.+2    ; SECOND OPERAND
205 0060          OP3:     .=.+1    ; TEMPORARY FOR ADDRESS
206 0061          G:       .=.+1    ; CURRENT ITEM PRECEDENCE
207  ;
208  ;         TEMPORARY FOR COMMANDS & EXECUTION ROUTINES
209  ;
210 0062          LIST1:   .=.+1    ; FIRST OPERAND
211 0063          LIST2:   .=.+1    ; SECOND OPERAND
212 0064          TMPADR:  .=.+1    ; ADDRESS
213 0065          TEMP:    .=.+8    ; VALUES - TO REMAIN DURING SINGLE LINE
214                                 ;    FOR INPUT, FOR-NEXT, FLDEC
215 006D          TAC0:    .=.+1    ; TEMPORARY FOR WITHIN SUBROUTINES ONLY
216 006E          TAC1:    .=.+1
217 006F          TAC2:    .=.+1
218 0070          TAC3:    .=.+1
```

```
219  ;
220  ;        SCRATCH LOCATIONS FOR FLOATING POINT ROUTINES
221  ;
222 0071          X:        .=.+2    ; ARGUMENT
223 0073          Y:        .=.+2    ; SETUP: EXPONENTS
224       0074 A  Y2        =          Y+1
225 0075          FLSIGN:   .=.+1    ; SETUP: SIGN SAVE
226 0076          FRSIGN:   .=.+1    ; MULT: SIGN SAVE
227 0077          XX:       .=.+2    ; EPSER: ARGUMENT
228 0079          ALAST:    .=.+1    ; EPSER,FLDEC: LAST COEFFICIENT ADDRESS
229 007A          COUNT:    .=.+1    ; EPSER,FLDEC: COEFFICIENT COUNT
230 007B          SCR:      .=.+2    ; SCRATCH FOR TRANSCENDENTAL FUNCTIONS,
231 007D          SCRA:     .=.+2    ;    FLMOD, AND FLINT
232 007F          SCRB:     .=.+1    ; FOR LN EXPONENT; IFIX/ROUND BIT
233 0080          XTEMP:    .=.+1    ; FLNORM: EXPONENT
234 0081          TT1:      .=.+1    ; FOR FLDIV
235 0082          TT2:      .=.+2
236 0084          TT3:      .=.+2
237  ;
238 0086          RNDNUM:   .=.+1    ; RANDOM NUMBER (BEFORE FLT PT CONV)
239  ;
240  ;
245 0087          FST:      .=.+14   ; DISK FILE STATUS TABLE
247  ;
248  ;        FIXED POINTERS
249  ;
250 0095 0125 T START:    .PTR     BEGIN
251 0096 0153 T           .PTR     READIN,GETS,PUTS,SAVE,RESTOR
252 009B 0488 T           .PTR     PUSH,PULL,PUTW,OUT1,CRLF,XCRLF,MESG
253 00A2 0576 T           .PTR     GETCH,SCAN,SKL,STRTMP,STKSTR,STR3
254 00A8 0D91 T           .PTR     CALCUL,GETVAL,GETFCN,STROP,RPAREN
255 00AD 027A T           .PTR     RDLINE,PACK,EFIND,FINDCR,INBUF,PBUF,SKL
256 00B4 12AD T           .PTR     IFIX,ROUND,INTFL,FLINT,MULT,IEXP,FLDEC
257 00BB 10E4 T           .PTR     FLCMP,FLADD,FLMULT,FLDIV,NEGATE
258 00C0 1210 T           .PTR     ADSHR,CLRSTK,CLRST1,ERROR,WARNING
259 00C5 0124 T           .PTR     RUNERR,SYNERR,EXPERR,OPERR,SYSERR
261 00CA 0087 A           .PTR     FST,DSKERR
263 00CC 0000 A           .PTR     0,2,9,0A,0D,0F,020,07F,080,0FF
264 00D6 7FFF A           .PTR     07FFF,08000,0C000,0F000,0FF00,0FFFF
265 00DC 4000 A FL1:      .PTR     04000,1    ; 1.0
266       0049 A NPTR     =          .-START  ; # POINTERS
267  ;
268  ;
269                       .NOBAS             ; NO ASM-GENERATED PTRS PLEASE
274 0000                  .BSECT             ; SET BSECT ORIGIN TO SKIP ASECT
275 0000                  .=.+FL1+2
276 0000                  .TSECT
277 0000                  .=.+0100           ; START TSECT AT 0100 (ABS LOADER)
279  ;
```

```
                        .PAGE   'MACROS'
;
;       DEFINITIONS FOR SYNTAX TABLE
;
;       .FORM   CHK1,13,1,2(0)
;       .FORM   CHK2,13,1,2(1)
;       .FORM   CHK3,1(1),12,1,2(2)
;       .FORM   ERR,1(0),13,2(2)
;       .FORM   CTL1,1,7,4,2,2(3)
;       .FORM   CTL2,7,7,2
;
                        .MACRO  ARGCHK,ARG,VAL
;       CHECK ARGUMENTS FOR VALID RANGE
                        .IF     ARG<0
                        .ERROR  'NEG ARGUMENT'
                        .ENDIF
                        .IF     ARG>VAL
                        .ERROR  'TOO BIG: ' ARG
                        .ENDIF
                        .ENDM
;
                        .MACRO  CHKX            ; GEN WORD FOR CHK1,2,3
                        ARGCHK  #1,07FF
                        ARGCHK  #2,1
                        .WORD   #1*2+#2*4!#3
                        .ENDM
;
                        .MACRO  CHK1
                        CHKX    #1,#2,0
                        .ENDM
;
                        .MACRO  CHK2
                        CHKX    #1,#2,1
                        .ENDM
;
                        .MACRO  CHK3
                        CHKX    #1,#2,08002
                        .ENDM
;
                        .MACRO  ERR
                        ARGCHK  #1,07F
                        .WORD   #1*4+2
                        .ENDM
;
                        .MACRO  CTL1
                        ARGCHK  #1,1
                        ARGCHK  #2,07F
                        ARGCHK  #3,0F
                        ARGCHK  #4,3
                        .WORD   #1*128+#2*16+#3*4+#4*4+3
                        .ENDM
;
                        .MACRO  CTL2
                        ARGCHK  #1,07F
```

```
                        ARGCHK  #2,07F
                        ARGCHK  #3,3
                        .WORD   #1*128+#2*4+#3
                        .ENDM
;
;
;       DLD     REG,VALUE       DOUBLE LOAD
;       DST     REG,VALUE       DOUBLE STORE
;
                        .MACRO  DLD,REG,VAL
                        ARGCHK  REG,2
                        LD      REG,VAL
                        LD      REG+1,VAL+1
                        .ENDM
;
                        .MACRO  DST,REG,VAL
                        ARGCHK  REG,2
                        ST      REG,VAL
                        ST      REG+1,VAL+1
                        .ENDM
;
```

```
                        .PAGE   'INTERPRET'
      8011 A            .LIST   LOPT!010
 ;
0100 1924 A FIRST:      JMP     BEGIN           ; *** RESTART ***
 ;
0101 0102 T $S1:        .WORD   .+1
0102 19C3 T             .WORD   STACK0          ; STACK LOCATIONS
0103 19E3 T             .WORD   STACK1
0104 19F2 T             .WORD   STACK2
0105 1A10 T             .WORD   STACK3
0106 1A24 T             .WORD   STACK4
0107 1A2E T             .WORD   STACK5
0108 1A38 T             .WORD   STACK6
 ;
 ;
 ;          CLEAR SOFTWARE STACKS: NUMBER IN AC1 (4 OR 7)
 ;
0109 5000 A CLRST1:     LI      AC0,0           ; SET FNLINE=0
010A D052 A             ST      AC0,FNLINE
010B 5200 A             LI      AC2,0
010C CDF4 A             LD      AC3,$S1
010D C300 A $S2:        LD      AC0,(AC3)
010E D223 A             ST      AC0,STPTR(AC2)
010F 7B01 A             AISZ    AC3,1
0110 7A01 A             AISZ    AC2,1
0111 79FF A             AISZ    AC1,-1
0112 19FA A             JMP     $S2
0113 8000 A             RTS     0
 ;
 ;
 ;          RESET POINTERS - COMES HERE AFTER INITIALIZATION
 ;                           AND UPON SCRATCH COMMAND
 ;
0114 C033 A SCRATCH:LD      AC0,ESTART      ; RESET EDIT STORAGE POINTERS
0115 D035 A             ST      AC0,EEND
0116 D034 A             ST      AC0,ECUR
0117 C038 A RESET:      LD      AC0,LAST        ; CLEAR SYMBOL TABLE
0118 D051 A             ST      AC0,SY2         ;    (RESET POINTERS)
0119 C035 A             LD      AC0,EEND
011A D055 A             ST      AC0,ANEXT
011B D056 A             ST      AC0,TNEXT
011C 5000 A             LI      AC0,0           ; RESET UNITS & MODES
011D D015 A             ST      AC0,TRMODE
011E D018 A             ST      AC0,RUNFLG
011F 5001 A             LI      AC0,1
0120 D017 A             ST      AC0,BRKMOD
0121 94C1*A             JSR     CLRSTK
0122 957B*A             JSR     RSDATA
0123 1901 A             JMP     BEGIN
 ;
 ;
 ;          RUNERR: COMES HERE AFTER EXECUTION TIME ERRORS
 ;          BEGIN:  TYPE 'READY' AND AWAIT COMMAND
 ;
```

```
 415 0124 94C3*A RUNERR: JSR     ERROR       ; RUN ERRORS - TYPE ERROR MSG TH
 416 0125 5104 A BEGIN:  LI      AC1,4
 417 0126 15E2 A         JSR     CLRST1      ; CLEAR 4 SOFTWARE STACKS
 418 0127 5000 A         LI      AC0,0       ; INITIALIZE MESG UNIT
 419 0128 D01C A         ST      AC0,OUTDVC
 420 0129 D01B A         ST      AC0,OUTDVX
 421 012A D012 A         ST      AC0,XQTFLG
 422 012B C104 A         LD      AC0,$JUMPI  ; INITIALIZE RESTART LOC
 423 012C D000 A         ST      AC0,0
 424 012D 94A1*A         JSR     MESG
 425 012E 186D T         .WORD   READY       ; TYPE 'READY'
 426 012F 1923 A         JMP     READIN
 427 0130 9895 A $JUMPI: JMP     @START
 428  ;
 429  ;       SET INPUT DEVICE AND READ FROM IT
 430  ;
 432 0131 5001 A EXEC:   LI      AC0,1       ; SET EXECUTE FLAG
 433 0132 D012 A         ST      AC0,XQTFLG
 434 0133 C033 A LOAD:   LD      AC0,ESTART  ; SCRATCH CURRENT PROGRAM
 435 0134 D035 A         ST      AC0,EEND
 436 0135 9569*A MERGE:  JSR     DRESET      ; READ FROM DISK
 437 0136 98CB*A         JMP     DSKERR
 438 0137 C8CA*A         LD      AC2,=FST
 439 0138 9567*A         JSR     OPENR
 440 0139 98CB*A         JMP     DSKERR
 441 013A 5002 A         LI      AC0,2
 442 013B 1901 A         JMP     .+2
 444 013C 5001 A PTAPE:  LI      AC0,1       ; INPUT FROM PAPER TAPE
 445 013D D01A A         ST      AC0,UNIT
 446 013E 5001 A         LI      AC0,1       ; SET MODE FOR NON-KEYBOARD DEVI
 447 013F 1918 A         JMP     MODEIN
 448  ;
 449 0140 5210 A AUTO:   LI      AC2,010     ; START LINE
 450 0141 5310 A         LI      AC3,010     ; INCREMENT
 451 0142 C060 A         LD      AC0,OP3     ; CHECK # ARGUMENTS ON COMMAND
 452 0143 9CDD*A         SKG     AC0,=1
 453 0144 1901 A         JMP     .+2
 454 0145 CC63 A         LD      AC3,LIST2
 455 0146 9CCC*A         SKG     AC0,=0
 456 0147 1901 A         JMP     .+2
 457 0148 C862 A         LD      AC2,LIST1
 458 0149 D862 A         ST      AC2,LIST1   ; LIST1 = BCD LINE NUMBER
 459 014A DC63 A         ST      AC3,LIST2   ; LIST2 = BCD INCREMENT
 460 014B 5300 A         LI      AC3,0
 461 014C 9554*A         JSR     BCDADD      ; CHECK BCD LEGALITY
 462 014D 1905 A         JMP     READIN      ;    OVERFLOW - ABORT AUTO MODE
 463 014E D462 A         ST      AC1,LIST1
 464 014F 5C40 A         RCPY    AC1,AC0
 465 0150 4102 A         BOC     ZRO,READIN  ; 0 LINE NUMBER NOT ALLOWED
 466 0151 5002 A         LI      AC0,2       ; SET AUTO LINE NUMBER MODE
 467 0152 1905 A         JMP     MODEIN
 468  ;
 469 0153 5000 A READIN: LI      AC0,0       ; PROMPT FOR TELETYPE INPUT
 470 0154 D01A A         ST      AC0,UNIT
```

```
 471 0155 D01C A          ST      AC0,OUTDVC
 472 0156 D01B A          ST      AC0,OUTDVX
 473 0157 D012 A          ST      AC0,XQTFLG
 474 0158 D016 A MODEIN:  ST      AC0,TXTMOD
 475 0159 94C1*A NEXTIN:  JSR     CLRSTK          ; BE SURE STACK IS CLEAR
 476 015A 94A0*A          JSR     XCRLF           ; START ON NEW LINE IF NOT THERE
 477 015B C016 A          LD      AC0,TXTMOD      ; CHECK FOR AUTO MODE
 478 015C F0CD*A          SKNE    AC0,=2
 479 015D 190C A          JMP     $A1
 480 015E 94AD*A          JSR     RDLINE          ; READ INPUT LINE
 481 015F 19F3 A          JMP     READIN          ;    ABORT
 482 0160 1904 A          JMP     $EOF            ;    EOF OR CONTROL-C
 483 0161 9540*A          JSR     PSCAN           ; PRESCAN & COMPRESS BUFFER
 484 0162 19F6 A          JMP     NEXTIN          ;    ERROR
 485 0163 19F5 A          JMP     NEXTIN          ;    BLANK LINE
 486 0164 1912 A          JMP     $CHK
 487 0165 C012 A $EOF:    LD      AC0,XQTFLG      ; END OF FILE: CHECK XQT FLAG
 488 0166 41EC A          BOC     ZRO,READIN
 489 0167 5000 A          LI      AC0,0           ; RUN THE PROGRAM NOW
 490 0168 D062 A          ST      AC0,LIST1
 491 0169 193E A          JMP     RUN
 492  ;
 493                                              ; AUTO MODE INPUT:
 494 016A C462 A $A1:     LD      AC1,LIST1       ; PROMPT WITH LINE NUMBER
 495 016B 9537*A          JSR     TYPLIN
 496 016C 94AD*A          JSR     RDLINE          ; READ INPUT LINE
 497 016D 19EB A          JMP     NEXTIN          ;    ABORT - RETRY
 498 016E 19E4 A          JMP     READIN          ;    CTL-C - END AUTO
 499 016F 9532*A          JSR     PSCAN           ; PRESCAN & COMPRESS
 500 0170 19E8 A          JMP     NEXTIN          ;    ERROR
 501 0171 19E1 A          JMP     READIN          ;    BLANK - END AUTO
 502 0172 C014 A          LD      AC0,CURLIN      ; CHECK IF LINE# ENTERED
 503 0173 4501 A          BOC     NZRO,.+2        ;    (OVERRIDES AUTO NUMBER)
 504 0174 C062 A          LD      AC0,LIST1       ; SAVE CURRENT LINE NUMBER
 505 0175 D014 A          ST      AC0,CURLIN
 506 0176 D062 A          ST      AC0,LIST1
 507  ;
 508 0177 C0B2*A $CHK:    LD      AC0,=PBUF       ; PREPARE TO SCAN
 509 0178 D03E A          ST      AC0,LSTART
 510 0179 5000 A          LI      AC0,0
 511 017A D03F A          ST      AC0,COL
 512 017B D021 A          ST      AC0,BRKFLG
 513 017C C014 A          LD      AC0,CURLIN      ; CHECK IF INPUT IS COMMAND OR T
 514 017D 4506 A          BOC     NZRO,NEWLIN
 515 017E 5002 A          LI      AC0,2           ; COMMAND MODE
 516 017F D013 A          ST      AC0,MODE
 517 0180 9523*A          JSR     SYNTAX          ; SCAN COMMAND
 518 0181 5C00 A          NOP
 519 0182 5C00 A          NOP
 520 0183 19D5 A          JMP     NEXTIN
 521  ;
 522  ;        EVALUATE SYNTAX BEFORE DECIDING TO SAVE
 523  ;
 524 0184 5001 A NEWLIN:  LI      AC0,1           ; EDIT MODE
```

```
  525 0185 D013 A          ST      AC0,MODE
  526 0186 951D*A          JSR     SYNTAX          ; CHECK SYNTAX
  527 0187 1912 A          JMP     $SERR
  528 0188 5C00 A          NOP
  529 0189 951B*A.STLIN:   JSR     ESTOR           ; STORE THE LINE
  530 018A 5000 A          LI      AC0,0
  531 018B D018 A          ST      AC0,RUNFLG      ; RUNFLG = 0
  532 018C C01A A          LD      AC0,UNIT        ; IF INPUT IS DISK, CHECK
  533 018D 78FE A          AISZ    AC0,-2          ;    FOR KEYBOARD INTERRUPT
  534 018E 1902 A          JMP     .+3
  535 018F 9516*A          JSR     INTEST
  536 0190 19C2 A          JMP     READIN
  537 0191 C016 A          LD      AC0,TXTMOD      ; UPDATE LINE NUMBER IN AUTO MOD
  538 0192 78FE A          AISZ    AC0,-2
  539 0193 19C5 A          JMP     NEXTIN
  540 0194 C862 A          LD      AC2,LIST1
  541 0195 CC63 A          LD      AC3,LIST2
  542 0196 950A*A          JSR     BCDADD
  543 0197 19BB A          JMP     READIN
  544 0198 D462 A          ST      AC1,LIST1       ; SAVE NEXT LINE #
  545 0199 19BF A          JMP     NEXTIN
  546 019A C016 A $SERR:   LD      AC0,TXTMOD      ; SYNTAX ERROR - CHECK INPUT DEV
  547 019B 78FF A          AISZ    AC0,-1
  548 019C 19BC A          JMP     NEXTIN          ; KEYBOARD - DON'T STORE
  549 019D 19EB A          JMP     STLIN           ; TAPE/DISK - STORE FOR LATER PR
  550  ;
  551 019E 0F2E T          .POOL   10
      019F D406 A
      01A0 D40E A
      01A1 151B T
      01A2 0594 T
      01A3 0375 T
      01A4 08E5 T
      01A5 04AF T
      01A6 7ECC A
      01A7 0000 A
```

```
552                     .PAGE   'RUN/CONTINUE COMMANDS'
553                     .LOCAL
554   ;
555 01A8 5000 A RUN:    LI      AC0,0           ; INITIALIZE FLAGS
556 01A9 D019 A         ST      AC0,ERRFLG
557 01AA D018 A         ST      AC0,RUNFLG
558 01AB D016 A         ST      AC0,TXTMOD
559 01AC D01A A         ST      AC0,UNIT
560 01AD D014 A         ST      AC0,CURLIN
561 01AE D021 A         ST      AC0,BRKFLG
562 01AF 5001 A         LI      AC0,1
563 01B0 5017 A         LI      AC0,BRKMOD
564 01B1 5004 A         LI      AC0,4           ; MODE=4 (PASS 1)
565 01B2 D013 A         ST      AC0,MODE
566 01B3 C038 A         LD      AC0,LAST        ; CLEAR SYMBOL TABLE
567 01B4 D051 A         ST      AC0,SY2
568 01B5 94C1*A         JSR     CLRSTK          ; RESET ALL STACKS
569 01B6 5107 A         LI      AC1,7
570 01B7 94C2*A         JSR     CLRST1
571 01B8 95E5*A         JSR     RSDATA          ; RESTORE 'DATA' POINTERS
572 01B9 C035 A         LD      AC0,EEND
573 01BA D055 A         ST      AC0,ANEXT
574 01BB D056 A         ST      AC0,TNEXT
575 01BC C833 A         LD      AC2,ESTART      ; INITIALIZE TO START WITH 1ST L
576 01BD F835 A         SKNE    AC2,EEND
577 01BE 9895*A         JMP     BEGIN           ; NO PROGRAM TO RUN - GO BACK TO
578 01BF D83E A NEXT1:  ST      AC2,LSTART      ; SAVE ADDR OF LINE
579 01C0 C200 A         LD      AC0,(AC2)       ; SAVE LINE NUMBER
580 01C1 D014 A         ST      AC0,CURLIN
581 01C2 5002 A         LI      AC0,2
582 01C3 D03F A         ST      AC0,COL
583 01C4 95DF*A         JSR     SYNTAX          ; EVALUATE THE SYNTAX
584 01C5 5C00 A         NOP                     ; KEEP GOING PAST ERRORS ETC.
585 01C6 5C00 A         NOP
586 01C7 C83E A         LD      AC2,LSTART      ; GET ADDR OF NEXT LINE
587 01C8 94B0*A         JSR     FINDCR
588 01C9 5EC0 A         RCPY    AC3,AC2
589 01CA F835 A         SKNE    AC2,EEND        ; CHECK IF AT END
590 01CB 1901 A         JMP     PASS2
591 01CC 19F2 A         JMP     NEXT1
592   ;
593 01CD 5306 A PASS2:  LI      AC3,6           ; CHECK FOR UNTERMINATED FOR-LOO
594 01CE 949C*A         JSR     PULL
595 01CF 1902 A         JMP     $A1             ; FORLOOP STACK EMPTY - OK
596 01D0 501C A         LI      AC0,28          ; UNTERMINATED FOR ON STACK
597 01D1 94C3*A         JSR     ERROR
598   ;
599   ;       ALLOCATE ALL STRINGS AND ARRAYS
600   ;
601 01D2 5000 A $A1:    LI      AC0,0           ; CURLIN=0
602 01D3 D014 A         ST      AC0,CURLIN
603 01D4 C035 A         LD      AC0,EEND        ; SET ARRAY AREA START ADDR
604 01D5 D055 A         ST      AC0,ANEXT
605 01D6 D056 A         ST      AC0,TNEXT
```

```
606 01D7 CC38 A          LD      AC3,SY1       ; LOAD START ADDR OF SYMBOL TABL
607 01D8 FC51 A $NXA:    SKNE    AC3,SY2       ; CHECK IF AT END
608 01D9 192B A          JMP     PASS2B        ;    GO ON FOR PASS2
609 01DA 7BFC A          AISZ    AC3,-4        ; SET AC3 FOR NEXT TABLE ENTRY
610 01DB DC65 A          ST      AC3,TEMP
611 01DC C301 A          LD      AC0,1(AC3)    ; CHECK IF STRING OR ARRAY
612 01DD 4201 A          BOC     POS,.+2
613 01DE 1903 A          JMP     $STR          ; STRING IF BIT 15 IS SET
614 01DF B8D4*A          SKAZ    AC0,=080
615 01E0 1908 A          JMP     $ARY
616 01E1 19F6 A          JMP     $NXA
617 01E2 C05B A $STR:    LD      AC0,NSTRW     ; STRING: ALLOCATE (NSTRW) WORDS
618 01E3 D04A A          ST      AC0,DIGIT     ;    PER ELEMENT
619 01E4 C301 A          LD      AC0,1(AC3)    ; CHECK IF STRING ARRAY
620 01E5 B8D4*A          SKAZ    AC0,=080
621 01E6 1904 A          JMP     $A2
622 01E7 C05B A          LD      AC0,NSTRW     ; ALLOCATE SIMPLE STRING
623 01E8 1913 A          JMP     $A4
624 01E9 5002 A $ARY:    LI      AC0,2         ; ARRAY: ALLOCATE 2 WORDS PER EL
625 01EA D04A A          ST      AC0,DIGIT
626 01EB C302 A $A2:     LD      AC0,2(AC3)    ; CHECK DIMENSIONS
627 01EC C703 A          LD      AC1,3(AC3)
628 01ED 4504 A          BOC     NZRO,$A3
629 01EE 500A A          LI      AC0,10        ; DEFAULT DIMENSIONS ARE (10,0)
630 01EF 5100 A          LI      AC1,0         ;    WHICH IS THE SAME AS (10)
631 01F0 D302 A          ST      AC0,2(AC3)    ; SAVE UPDATED DIMENSIONS
632 01F1 D703 A          ST      AC1,3(AC3)
633 01F2 7801 A $A3:     AISZ    AC0,1         ; ADD 1 TO EACH DIM SINCE
634 01F3 7901 A          AISZ    AC1,1         ;    BOUNDS START AT 0
635 01F4 94B8*A          JSR     MULT          ; DETERMINE TOTAL # OF WORDS REQ
636 01F5 450A A          BOC     NZRO,$AERR    ; PRODUCT MUST BE LESS THAN 1638
637 01F6 C04A A          LD      AC0,DIGIT     ; MULTIPLY BY # WORDS PER ELEMEN
638 01F7 94B8*A          JSR     MULT
639 01F8 4507 A          BOC     NZRO,$AERR
640 01F9 5C40 A          RCPY    AC1,AC0
641 01FA 4105 A          BOC     ZRO,$AERR
642 01FB 4B04 A          BOC     NEG,$AERR
643 01FC 155F A $A4:     JSR     ALOC          ; ALLOCATE STRING/ARRAY SPACE
644 01FD CC65 A          LD      AC3,TEMP      ; SAVE ADDRESS IN SYMBOL TABLE
645 01FE DB04 A          ST      AC2,4(AC3)
646 01FF 19D8 A          JMP     $NXA
647 0200 9499*A $AERR:   JSR     SAVE          ; PRODUCT OF DIMENSIONS IS TOO B
648 0201 5016 A          LI      AC0,22
649 0202 94C3*A          JSR     ERROR
650 0203 949A*A          JSR     RESTOR
651 0204 19D3 A          JMP     $NXA
652  ;
653  ;        NOW RUN THE PROGRAM IF NO ERRORS WERE FOUND
654  ;
655 0205 C019 A PASS2B:  LD      AC0,ERRFLG    ; CONTINUE IF NO ERRORS
656 0206 450A A          BOC     NZRO,$BEGIN
657 0207 949F*A          JSR     CRLF
658 0208 5001 A          LI      AC0,1         ; RUNFLG=1
659 0209 D018 A          ST      AC0,RUNFLG
```

```
 660 020A 5008 A          LI      AC0,8          ; MODE=8 (PASS 2)
 661 020B D013 A          ST      AC0,MODE
 662 020C C462 A          LD      AC1,LIST1      ; START OFF WITH 1ST LINE OR WIT
 663 020D 94AF*A          JSR     EFIND          ;    LINE FROM RUN COMMAND
 664 020E 5C00 A.         NOP
 665 020F D83E A NEXT2:   ST      AC2,LSTART     ; SAVE LINE ADDRESS
 666 0210 F835 A          SKNE    AC2,EEND
 667 0211 9895*A $BEGIN:  JMP     BEGIN          ; PROGRAM DONE
 668 0212 9594*A          JSR     LINIT          ; INITIALIZE FOR NEW LINE
 669 0213 94B0*A          JSR     FINDCR         ; SAVE ADDRESS OF NEXT LINE
 670 0214 DC40 A          ST      AC3,NEXTAD
 671 0215 C200 A          LD      AC0,(AC2)
 672 0216 D014 A          ST      AC0,CURLIN     ; SAVE LINE NUMBER
 673 0217 5002 A          LI      AC0,2
 674 0218 D03F A          ST      AC0,COL
 675 0219 C017 A          LD      AC0,BRKMOD     ; CHECK FOR BREAK
 676 021A 4104 A          BOC     ZRO,$TR        ;    CAPABILITY IS OFF
 677 021B 958A*A          JSR     INTEST
 678 021C 8C21 A          ISZ     BRKFLG
 679 021D C021 A          LD      AC0,BRKFLG
 680 021E 450B A          BOC     NZRO,BREAK
 681 021F C015 A $TR:     LD      AC0,TRMODE     ; CHECK FOR TRACE MODE
 682 0220 4104 A          BOC     ZRO,XQT        ;    NO - EXECUTE THE LINE
 683 0221 94A0*A          JSR     XCRLF          ; CRLF IF NOT AT LEFT MARGIN
 684 0222 94A1*A          JSR     MESG
 685 0223 187F T          .WORD   POINT
 686 0224 9531*A          JSR     LIST           ; LIST THE LINE TO BE EXECUTED
 687 0225 9531*A XQT:     JSR     SYNTAX         ; PROCESS THE LINE
 688 0226 190B A          JMP     $ERR
 689 0227 190D A          JMP     STOP           ; STOP/END
 690 0228 C840 A          LD      AC2,NEXTAD
 691 0229 19E5 A          JMP     NEXT2          ; GO TO NEXT LINE
 692  ;
 693 022A C03E A BREAK:   LD      AC0,LSTART     ; BREAK: SAVE NEXT ADDRESS
 694 022B D040 A          ST      AC0,NEXTAD
 695 022C 9499*A          JSR     SAVE           ; SAVE REG (INLINE WILL RESTORE)
 696 022D 94A0*A          JSR     XCRLF          ; 'BREAK IN ...'
 697 022E 94A1*A          JSR     MESG
 698 022F 188F T          .WORD   BRK
 699 0230 9527*A          JSR     INLINE
 700 0231 9896*A          JMP     READIN
 701  ;
 702 0232 5000 A $ERR:    LI      AC0,0          ; ERROR - RESET RUNFLG
 703 0233 D018 A          ST      AC0,RUNFLG
 704 0234 9895*A          JMP     BEGIN
 705 0235 5000 A STOP:    LI      AC0,0          ; GIVE STOP MESSAGE
 706 0236 D01C A          ST      AC0,OUTDVC     ;    TO TERMINAL
 707 0237 9499*A          JSR     SAVE
 708 0238 94A0*A          JSR     XCRLF
 709 0239 94A1*A          JSR     MESG
 710 023A 188C T          .WORD   STOPM
 711 023B 951C*A          JSR     INLINE
 712 023C 9896*A          JMP     READIN
 713  ;
```

```
714  ;        CONTINUE COMMAND
715  ;
716 023D C018 A CONT:   LD      AC0,RUNFLG   ; CONTINUE ONLY IF RUNFLG=1
717 023E 4503 A         BOC     NZRO,$CNT
718 023F 5002 A         LI      AC0,2        ; NO CONTINUE
719 0240 94C3*A         JSR     ERROR
720 0241 9896*A         JMP     READIN
721 0242 5008 A $CNT:   LI      AC0,8        ; MODE=8 (PASS 2)
722 0243 D013 A         ST      AC0,MODE
723 0244 C840 A         LD      AC2,NEXTAD   ; LOAD LINE ADDRESS
724 0245 C460 A         LD      AC1,OP3      ; CHECK IF OPERAND GIVEN
725 0246 7900 A         AISZ    AC1,0
726 0247 1901 A         JMP     .+2
727 0248 19C6 A         JMP     NEXT2        ; NO OPERAND - CONTINUE WHERE LE
728 0249 C462 A         LD      AC1,LIST1
729 024A 94AF*A         JSR     EFIND        ; FIND LINE FOR CONTINUE
730 024B 5C00 A         NOP
731 024C 19C2 A         JMP     NEXT2        ; GO TO IT
732  ;
733  ;        BYE COMMAND - RETURN TO MONITOR
734  ;                (ERROR IF NO MONITOR PRESENT)
735  ;
737 024D CD0B*A BYE:    LD      AC3,=0D000   ; CHECK IF THERE IS A MONITOR
738 024E C300 A         LD      AC0,(AC3)    ; - LOCATION D000 SHOULD BE 0
739 024F 4504 A         BOC     NZRO,$BYERR
740 0250 C0D9*A         LD      AC0,=0F000   ; LOCATION 0D HAS RETURN ADDRESS
741 0251 A80D A         AND     AC0,0D       ;     (MUST BE 0DXXX)
742 0252 F106*A         SKNE    AC0,=0D000
743 0253 980D A         JMP     @0D
744 0254 5004 A $BYERR: LI      AC0,4        ; NO MONITOR: STMT ERROR
745 0255 98C5*A         JMP     RUNERR
747  ;
748 0256 033B T         .POOL   6
    0257 08E5 T
    0258 03E5 T
    0259 D000 A
    025A 0000 A
    025B 0000 A
```

```
                        .PAGE   'MEMORY ALLOCATION'
  ;
  ;         MEMORY ALLOCATION ROUTINE:
  ;
  ;         ON ENTRY:  AC0 = NUMBER OF WORDS TO ALLOCATE
  ;         ON EXIT:   AC2 = START ADDRESS OF AREA
  ;
  ;         ALOC ALLOCATES 'PERMANENT' STORAGE (FOR DURATION OF PROGRAM)
  ;         TLOC ALLOCATES 'TEMPORARY' STORAGE (FOR DURATION OF LINE)
  ;
  ;         MEMORY AREA IS INITIALIZED TO ZERO
  ;
025C 5255 A ALOC:   LI      AC2,ANEXT
025D 1901 A         JMP     .+2
025E 5256 A TLOC:   LI      AC2,TNEXT
025F D84F A         ST      AC2,PFLAG       ; SAVE ADDR TO HOLD NEXT LOC
0260 CA00 A         LD      AC2,(AC2)       ; ALLOCATE (AC0) WORDS
0261 6880 A         RADD    AC2,AC0
0262 B04F A         ST      AC0,@PFLAG      ; STORE NEW NEXT ADDRESS
0263 D04F A         ST      AC0,PFLAG
0264 C451 A         LD      AC1,SY2         ; CHECK FOR STORAGE OVERFLOW
0265 94A4*A         JSR     SKL             ; CHECK IF ARRAY FITS BELOW SYMB
0266 1907 A         JMP     $AER4
0267 5F80 A         RCPY    AC2,AC3         ; ARRAY FIT IN MEMORY OK
0268 5000 A         LI      AC0,0           ; CLEAR MEMORY AREA
0269 D300 A $CLR:   ST      AC0,(AC3)
026A FC4F A         SKNE    AC3,PFLAG
026B 8000 A         RTS     0
026C 7B01 A         AISZ    AC3,1
026D 19FB A         JMP     $CLR
  ;
026E 5000 A $AER4:  LI      AC0,0           ; ARRAY DOES NOT FIT - QUIT ALLO
026F 98C5*A         JMP     RUNERR
  ;
  ;
  ;         STRTMP:  ALLOCATE A TEMPORARY STRING AND
  ;                  INITIALIZE  PUTS  POINTERS FOR ITS USE
  ;
0270 9499*A STRTMP: JSR     SAVE
0271 C05B A         LD      AC0,NSTRW       ; ALLOCATE (NSTRW) WORDS
0272 15EB A         JSR     TLOC
0273 7A01 A         AISZ    AC2,1           ; STRING STARTS AT 2ND WORD OF A
0274 D857 A         ST      AC2,SSTART
0275 5000 A         LI      AC0,0
0276 D058 A         ST      AC0,SCOL
0277 C05A A         LD      AC0,NSTRC
0278 D059 A         ST      AC0,MAXSTR
0279 989A*A         JMP     RESTOR
  ;
```

```
                        .PAGE   'READ INPUT LINE'
                        .LOCAL
;
;
;       JSR     RDLINE          READ A LINE FROM THE INPUT DEVICE
;       <LINE ABORTED RETURN>
;       <ETX OR EOF RETURN>     (ETX = CONTROL-C)
;       <NORMAL RETURN>
;
;       SPECIAL CHARACTERS ARE:
;          LINE END:            0D (CR)
;          SPECIAL RETURN:      03 (ETX) -- USED TO STOP INPUT
;          BACKSPACE:           08 (BS), 05F (BACK-ARROW)
;          TAB:                 09 (HT)
;          IGNORED:             0 (NULL), 0A (LF), 07F (RUBOUT/DELETE)
;          ALSO IGNORED:        01B (ESC) AND THE CHARACTER FOLLOWING IT
;                               (KEYBOARD INPUT ONLY)
;          OTHER CONTROL CHARACTERS ABORT THE INPUT LINE
;
;       USED BY NEXTIN (COMMAND MODE)
;        AND BY INPUT  (RUN MODE)
;
027A 5020 A RDLINE: LI      AC0,BLANK     ; CLEAR THE INPUT BUFFER TO BLAN
027B CCB1*A         LD      AC3,=INBUF
027C D300 A $RD1:   ST      AC0,(AC3)
027D FC20 A         SKNE    AC3,ENDBUF
027E 1902 A         JMP     $RD2
027F 7B01 A         AISZ    AC3,1
0280 19FB A         JMP     $RD1
0281 500D A $RD2:   LI      AC0,0D        ; STORE CR INTO LAST WORD + 1
0282 D301 A         ST      AC0,1(AC3)
0283 C8B1*A         LD      AC2,=INBUF    ; LOAD BUFFER ADDRESS
0284 C01A A         LD      AC0,UNIT
0285 503F A         LI      AC0,'?'/256   ; PROMPT IF KEYBOARD INPUT
0286 1549 A         JSR     TECHO
0287 5020 A         LI      AC0,BLANK
0288 1547 A         JSR     TECHO
0289 C01A A $IN1:   LD      AC0,UNIT      ; TEST FOR DISK INPUT
028A 78FE A         AISZ    AC0,-2
028B 1904 A         JMP     $GET
028C 95CD*A         JSR     DISKIN
028D 8001 A         RTS     1
028E 8001 A         RTS     1
028F 8002 A         RTS     2
;
;       TERMINAL INPUT PROCESSING - KEYBOARD OR PAPER TAPE
;
0290 C01A A $GET:   LD      AC0,UNIT      ; TEST WHETHER TO ECHO
0291 4102 A         BOC     ZR0,$KB
0292 95C8*A         JSR     GETC
0293 1901 A         JMP     $KB+1
0294 9542*A $KB:    JSR     GECO
0295 A8D3*A         AND     AC0,=07F      ; REMOVE PARITY BIT
0296 41F9 A         BOC     ZR0,$GET      ; IGNORE NULL
```

```
854 0297 F0D3*A          SKNE    AC0,=07F      ; IGNORE RUBOUT
855 0298 19F7 A          JMP     $GET
856 0299 F0CF*A          SKNE    AC0,=0A       ; IGNORE LINE FEED
857 029A 19F5 A          JMP     $GET
858 029B F0D0*A          SKNE    AC0,=0D       ; CR TERMINATES LINE
859 029C 1912 A          JMP     $CRLF
860 029D F13A*A          SKNE    AC0,=3        ; ETX = CONTROL-C
861 029E 1914 A          JMP     $ETX
862 029F F139*A          SKNE    AC0,=8        ; BS = BACKSPACE
863 02A0 191C A          JMP     BACK
864 02A1 F0CE*A          SKNE    AC0,=9        ; ACCEPT TABS
865 02A2 1907 A          JMP     $S1
866 02A3 F136*A          SKNE    AC0,=01B      ; ESC = ESCAPE
867 02A4 1914 A          JMP     $ESC
868 02A5 F135*A          SKNE    AC0,=05F      ; BACKARROW - BACKSPACE
869 02A6 1916 A          JMP     BACK
870 02A7 B934*A          SKAZ    AC0,=060      ; ABORT IF OTHER CONTROL CHARACT
871 02A8 1901 A          JMP     $S1
872 02A9 191A A          JMP     $ERR
873 02AA D200 A $S1:     ST      AC0,(AC2)     ; STORE CHARACTER
874 02AB F820 A          SKNE    AC2,ENDBUF
875 02AC 1902 A          JMP     $CRLF         ; FORCE CR IF PAST SET LENGTH
876 02AD 7A01 A          AISZ    AC2,1
877 02AE 19E1 A          JMP     $GET
878 02AF 500D A $CRLF:   LI      AC0,0D        ; STORE CARRIAGE RETURN
879 02B0 D200 A          ST      AC0,(AC2)
880 02B1 1517 A $OVER:   JSR     CRLF1         ; PRINT CR/LF IF KEYBOARD MODE
881 02B2 8002 A          RTS     2
882  ;
883 02B3 505E A $ETX:    LI      AC0,05E       ; ETX: ECHO ^C IF KEYBOARD INPUT
884 02B4 151B A          JSR     TECHO
885 02B5 5043 A          LI      AC0,'C'/256
886 02B6 1519 A          JSR     TECHO
887 02B7 1511 A          JSR     CRLF1
888 02B8 8001 A          RTS     1
889  ;
890 02B9 C01A A $ESC:    LD      AC0,UNIT      ; ESCAPE - IGNORE IT AND CHARACT
891 02BA 4509 A          BOC     NZRO,$ERR     ;    FOLLOWING (KEYBOARD ONLY)
892 02BB 951B*A          JSR     GECO
893 02BC 19D3 A          JMP     $GET
894  ;
895 02BD 7AFF A BACK:    AISZ    AC2,-1        ; BACK UP BUFFER POINTER
896 02BE 5020 A          LI      AC0,BLANK     ; BLANK CURRENT CHARACTER
897 02BF D200 A          ST      AC0,(AC2)
898 02C0 5C80 A          RCPY    AC2,AC0       ; CHECK FOR UNDERFLOW
899 02C1 9D1B*A          SKG     AC0,=INBUF-1
900 02C2 1901 A          JMP     $ERR
901 02C3 19CC A          JMP     $GET
902  ;
903 02C4 C01A A $ERR:    LD      AC0,UNIT      ; INPUT ABORTED
904 02C5 43B4 A          BOC     BIT0,RDLINE   ; PAPER TAPE - START LINE OVER
905 02C6 505C A          LI      AC0,'\'/256   ; PRINT TWO BACK-SLASHES
906 02C7 1508 A          JSR     TECHO
907 02C8 1507 A          JSR     TECHO         ; CONTINUE ON TO CRLF
```

```
908  ;
909 02C9 500D A CRLF1:  LI      ACO,0D          ; PRINT CR AND LF
910 02CA 1505 A         JSR     TECHO           ;    TO TELETYPE
911 02CB 500A A         LI      ACO,0A
912 02CC 1503 A         JSR     TECHO
913 02CD 5001 A         LI      ACO,1           ; OUTCOL=1
914 02CE D01D A         ST      ACO,OUTCOL
915 02CF 8000 A         RTS     0
916  ;
917 02D0 6000 A TECHO:  PUSH    ACO             ; PRINT TO TTY IF IN KEYBOARD MO
918 02D1 C01A A         LD      ACO,UNIT
919 02D2 4102 A         BOC     ZRO,$TTY
920 02D3 6400 A         PULL    ACO
921 02D4 8000 A         RTS     0
922 02D5 6400 A $TTY:   PULL    ACO             ; RECOVER CHAR AND PRINT
923 02D6 9907*A         JMP     PUTC
924  ;
925 02D7 7E59 A         .POOL   10
    02D8 0003 A
    02D9 0008 A
    02DA 001B A
    02DB 005F A
    02DC 0060 A
    02DD 18BD T
    02DE 7E44 A
    02DF 0000 A
    02E0 0000 A
```

```
  926                       .PAGE   'LIST'
  927                       .LOCAL
  928  ;
  929  ;         PROCESS LIST COMMANDS
  930  ;
  931 02E1 CDFD*A HLIST:  LD      AC3,=08048    ; HIGH SPEED PRINTER LIST
  932 02E2 C301 A         LD      AC0,1(AC3)    ; READ STATUS
  933 02E3 4401 A         BOC     BIT1,.+2      ; LIST ON TTY IF NO PRINTER
  934 02E4 1916 A         JMP     TLIST
  935 02E5 5002 A         LI      AC0,2         ; CHANGE OUTPUT DEVICE
  936 02E6 D01C A         ST      AC0,OUTDVC
  937 02E7 503C A         LI      AC0,60        ; SET 60 LINES/PAGE
  938 02E8 D07A A         ST      AC0,COUNT
  939 02E9 1914 A         JMP     LISTIT
  940 02EA 5001 A PUNCH:  LI      AC0,1         ; PAPER TAPE LIST
  941 02EB D01C A         ST      AC0,OUTDVC
  942 02EC 95F3*A         JSR     GETC          ; WAIT FOR PUNCH TURN ON - HIT A
  943 02ED 5000 A         LI      AC0,0         ; SEND LEADER
  944 02EE 5128 A         LI      AC1,40
  945 02EF 949E*A         JSR     OUT1
  946 02F0 79FF A         AISZ    AC1,-1
  947 02F1 19FD A         JMP     .-2
  948 02F2 190B A         JMP     LISTIT
  950 02F3 9544*A SAVFIL: JSR     DRESET        ; DISK FILE LIST
  951 02F4 98CB*A         JMP     DSKERR
  952 02F5 C8CA*A         LD      AC2,=FST
  953 02F6 9542*A         JSR     OPENW
  954 02F7 98CB*A         JMP     DSKERR
  955 02F8 5008 A         LI      AC0,8
  956 02F9 D01C A         ST      AC0,OUTDVC
  957 02FA 1903 A         JMP     LISTIT
  959 02FB 5000 A TLIST:  LI      AC0,0         ; TELETYPE LIST
  960 02FC D01C A         ST      AC0,OUTDVC
  961 02FD 156D A         JSR     CRLF
  962 02FE 5001 A LISTIT: LI      AC0,1         ; TURN BREAK CAPABILITY ON
  963 02FF D017 A         ST      AC0,BRKMOD
  964 0300 C062 A         LD      AC0,LIST1     ; CHECK TO BE SURE LIST1<=LIST2
  965 0301 C463 A         LD      AC1,LIST2
  966 0302 94A4*A         JSR     SKL
  967 0303 D063 A         ST      AC0,LIST2
  968 0304 C462 A         LD      AC1,LIST1     ; DETERMINE STORAGE ADDRESSES
  969 0305 94AF*A         JSR     EFIND         ;    OF RANGE TO BE LISTED
  970 0306 5C00 A         NOP
  971 0307 D862 A         ST      AC2,LIST1
  972 0308 C463 A         LD      AC1,LIST2
  973 0309 94AF*A         JSR     EFIND
  974 030A 5F80 A         RCPY    AC2,AC3
  975 030B DC63 A         ST      AC3,LIST2
  976 030C C862 A         LD      AC2,LIST1
  977 030D F863 A $LIST:  SKNE    AC2,LIST2
  978 030E 1913 A         JMP     $ENDL
  979 030F 152B A         JSR     LIST
  980 0310 7901 A         AISZ    AC1,1         ; CONVERT CHAR COUNT TO WORD COU
  981 0311 2D02 A         SHR     AC1,1,0       ;    AND COMPUTE NEW LINE ADDRES
```

LIST

```
 982 0312 6A40 A          RADD    AC1,AC2
 983 0313 C01C A          LD      AC0,OUTDVC    ; DO NOT BREAK DISK SAVE
 984 0314 F1C4*A          SKNE    AC0,=8
 985 0315 19F7 A          JMP     $LIST
 986 0316 C021 A          LD      AC0,BRKFLG    ; CONTINUE LIST UNLESS BREAK FLA
 987 0317 450A A          BOC     NZRO,$ENDL
 988 0318 C01C A          LD      AC0,OUTDVC    ; IF OUTPUT IS TO PRINTER,
 989 0319 78FE A          AISZ    AC0,-2        ;    UPDATE LINE COUNT
 990 031A 19F2 A          JMP     $LIST
 991 031B AC7A A          DSZ     COUNT         ; DECREMENT LINE COUNT
 992 031C 19F0 A          JMP     $LIST
 993 031D 503C A          LI      AC0,60        ; RESET COUNT
 994 031E D07A A          ST      AC0,COUNT
 995 031F 500C A          LI      AC0,0C
 996 0320 156E A          JSR     OUT1
 997 0321 19EB A          JMP     $LIST
 998 0322 C01C A $ENDL:   LD      AC0,OUTDVC    ; END OF LIST
 999 0323 F1B5*A          SKNE    AC0,=8        ; DISK FILE ?
1000 0324 150B A          JSR     $ENDD         ;    YES
1001 0325 500A A          LI      AC0,0A        ; END LIST WITH LINE FEED
1002 0326 1568 A          JSR     OUT1
1003 0327 C01C A          LD      AC0,OUTDVC    ; IF DEVICE WAS PUNCH,
1004 0328 78FF A          AISZ    AC0,-1        ;    SEND TRAILER
1005 0329 9896*A          JMP     READIN
1006 032A 5128 A          LI      AC1,40
1007 032B 1563 A          JSR     OUT1
1008 032C 79FF A          AISZ    AC1,-1
1009 032D 19FD A          JMP     .-2
1010 032E 95B1*A          JSR     GETC          ; TURN OFF PUNCH, WAIT FOR ANY K
1011 032F 9896*A          JMP     READIN
1012 0330 501C A $ENDD:   LI      AC0,01C       ; DISK END: 01C=EOF
1013 0331 155D A          JSR     OUT1
1014 0332 C8CA*A          LD      AC2,=FST
1015 0333 9506*A          JSR     CLOSE         ; CLOSE FILE
1016 0334 98CB*A          JMP     DSKERR
1017 0335 5000 A          LI      AC0,0         ; REMAINING OUTPUT TO TERMINAL
1018 0336 D01C A          ST      AC0,OUTDVC
1019 0337 8000 A          RTS     0
1020  ;
1021 0338 D406 A          .POOL   3
     0339 D410 A
     033A D416 A
1022  ;
1023 033B C600 A LIST:    LD      AC1,(AC2)     ; LIST (AC2) BUFFER
1024 033C 5020 A          LI      AC0,BLANK     ; FIRST IS LINE NUMBER
1025 033D 1538 A          JSR     TYPLSP
1026 033E 5102 A          LI      AC1,2         ; SET COL. COUNTER
1027 033F 9497*A $1:      JSR     GETS
1028 0340 F0D0*A          SKNE    AC0,=0D
1029 0341 1929 A          JMP     CRLF
1030 0342 F0CE*A          SKNE    AC0,=9        ; TAB ?
1031 0343 190E A          JMP     $TAB
1032 0344 B8D4*A          SKAZ    AC0,=080
1033 0345 1902 A          JMP     $K0
```

LIST

```
1034 0346 1548 A $1A:    JSR     OUT1          ; PRINT REGULAR CHARACTER
1035 0347 19F7 A         JMP     $1
1036 0348 9499*A $K0:    JSR     SAVE          ; PRINT KEYWORD GIVEN ITS CODE
1037 0349 CD27*A         LD      AC3,=RESTAB   ; LOOKUP CODE IN TABLE
1038 034A F300 A $K1:    SKNE    AC0,0(AC3)
1039 034B 1902 A         JMP     $K2
1040 034C 7B01 A         AISZ    AC3,1
1041 034D 19FC A         JMP     $K1
1042 034E 7B01 A $K2:    AISZ    AC3,1         ; PRINT KEYWORD (ASCII)
1043 034F 9522*A         JSR     MESG3
1044 0350 949A*A         JSR     RESTOR
1045 0351 19ED A         JMP     $1
1046 0352 C01C A $TAB:   LD      AC0,OUTDVC    ; TAB: PRINT SPACES TO NEXT TAB
1047 0353 F185*A         SKNE    AC0,=8        ;    STOP, BUT NOT TO DISK OR PT
1048 0354 190F A         JMP     $T2
1049 0355 F0DD*A         SKNE    AC0,=1
1050 0356 190D A         JMP     $T2
1051 0357 C01D A         LD      AC0,OUTCOL
1052 0358 E0DB*A         ADD     AC0,=-1       ; TABS ARE EVERY 8 COLUMNS
1053 0359 A919*A         AND     AC0,=7        ;    (9,17,25,...)
1054 035A 7001 A         CAI     AC0,1
1055 035B E118*A         ADD     AC0,=8
1056 035C 6100 A         PUSH    AC1
1057 035D 5D00 A         RCPY    AC0,AC1
1058 035E 5020 A         LI      AC0,BLANK
1059 035F 152F A         JSR     OUT1
1060 0360 79FF A         AISZ    AC1,-1
1061 0361 19FD A         JMP     .-2
1062 0362 6500 A         PULL    AC1
1063 0363 19DB A         JMP     $1
1064 0364 5009 A $T2:    LI      AC0,9         ; SEND TAB CHAR TO PT/DISK
1065 0365 19E0 A         JMP     $1A
1066  ;
1067 0366 9499*A XCRLF:  JSR     SAVE          ; PRINT CR AND LF IF NOT AT
1068 0367 C01D A         LD      AC0,OUTCOL    ;    COLUMN 1
1069 0368 F0DD*A         SKNE    AC0,=1
1070 0369 989A*A         JMP     RESTOR        ; ALREADY AT COL 1 - RETURN
1071 036A 1901 A         JMP     CRLF+1
1072  ;
1073 036B 9499*A CRLF:   JSR     SAVE          ; CR/LF TO OUTPUT DEVICE
1074 036C 500D A         LI      AC0,0D
1075 036D 1521 A         JSR     OUT1
1076 036E 500A A         LI      AC0,0A
1077 036F 151F A         JSR     OUT1
1078 0370 989A*A         JMP     RESTOR        ; (RTS 0)
1079  ;
1080 0371 168C T         .POOL   4
     0372 0417 T
     0373 0007 A
     0374 0008 A
```

```
1081                        .PAGE   'OUTPUT ROUTINES'
1082  ;
1083  ;         TYPLIN - TYPE THE LINE NUMBER - BLANK LEADING ZEROES
1084  ;                  AC1 = LINE NUMBER (BCD)
1085  ;         TYPLSP - TYPE THE LINE NUMBER - WITH LEADING CHAR
1086  ;                  AC0 = CHARACTER TO REPLACE LEADING ZEROES
1087  ;                  AC1 = LINE NUMBER (BCD)
1088  ;
1089                        .LOCAL
1090 0375 5000 A TYPLIN:  LI      AC0,0           ; TYPE LINE NO.  PACKED
1091 0376 D04F A TYPLSP:  ST      AC0,PFLAG       ; ENTER HERE TO SET LEADING CHAR
1092 0377 5304 A          LI      AC3,4           ; DIGIT COUNTER
1093 0378 5C40 A          RCPY    AC1,AC0
1094 0379 4501 A          BOC     NZRO,.+2        ; IF NUMBER IS ZERO, SET TO 000F
1095 037A 510F A          LI      AC1,0F          ;    (TO BE PRINTED '   0')
1096 037B 5C40 A $T1:     RCPY    AC1,AC0         ; CHECK FOR LEADING ZEROES
1097 037C 2908 A          SHL     AC1,4,0
1098 037D 2C18 A          SHR     AC0,12,0
1099 037E 4509 A          BOC     NZRO,$T3
1100 037F C04F A          LD      AC0,PFLAG       ; CHECK LEADING ZERO CHARACTER
1101 0380 4101 A          BOC     ZRO,.+2         ;    DON'T PRINT IF NULL
1102 0381 150D A          JSR     OUT1
1103 0382 7BFF A          AISZ    AC3,-1
1104 0383 19F7 A          JMP     $T1
1105 0384 1909 A          JMP     $T4
1106 0385 5C40 A $T2:     RCPY    AC1,AC0
1107 0386 2908 A          SHL     AC1,4,0
1108 0387 2C18 A          SHR     AC0,12,0
1109 0388 F0D1*A $T3:     SKNE    AC0,=0F         ; 0F IS ZERO DIGIT (SEE ABOVE)
1110 0389 5000 A          LI      AC0,0
1111 038A 7830 A          AISZ    AC0,'0'/256     ; CONVERT BCD DIGIT TO ASCII
1112 038B 1503 A          JSR     OUT1            ; PRINT DIGIT
1113 038C 7BFF A          AISZ    AC3,-1
1114 038D 19F7 A          JMP     $T2
1115 038E 5020 A $T4:     LI      AC0,BLANK       ; PRINT BLANK AND EXIT (THROUGH
1116  ;
1117  ;         OUT1 - PRINT 1 CHARACTER ON THE OUTPUT DEVICE
1118  ;
1119 038F D06D A OUT1:    ST      AC0,TAC0
1120 0390 C017 A          LD      AC0,BRKMOD      ; CHECK BREAK CAPABILITY
1121 0391 4102 A          BOC     ZRO,$01         ;    TURNED OFF - SKIP TEST
1122 0392 9531*A          JSR     INTEST          ; CHECK FOR TTY INTERRUPT
1123 0393 8C21 A          ISZ     BRKFLG          ;    YES - SET FLAG FOR LATER TE
1124 0394 C01C A $01:     LD      AC0,OUTDVC      ; DON'T UPDATE COLUMN
1125 0395 B92F*A          SKAZ    AC0,=12         ;    IF OUTDVC IS STRING
1126 0396 1917 A          JMP     $P1             ;    OR DISK
1127 0397 C06D A          LD      AC0,TAC0
1128 0398 A3D3*A          AND     AC0,=07F        ; TEST CHAR BEING PRINTED
1129 0399 F0D0*A          SKNE    AC0,=0D
1130 039A 190D A          JMP     $CR             ; CARRIAGE RETURN - RESET OUTCOL
1131 039B F1D8*A          SKNE    AC0,=08         ; BACKSPACE ?
1132 039C 190E A          JMP     $BS
1133 039D B928*A          SKAZ    AC0,=060
1134 039E 1901 A          JMP     .+2
```

```
1135 039F 190E A          JMP     $P1             ; OTHER CONTROL - DON'T CHANGE O
1136 03A0 8C1D A          ISZ     OUTCOL          ; INCREMENT OUTPUT COLUMN COUNT
1137 03A1 C01D A          LD      AC0,OUTCOL
1138 03A2 9C1E A          SKG     AC0,LINLEN      ; CHECK IF PAST END OF LINE
1139 03A3 190A A          JMP     $P1
1140 03A4 C06D A          LD      AC0,TACO        ; NEW LINE NEEDED
1141 03A5 15C5 A          JSR     CRLF
1142 03A6 D06D A          ST      AC0,TACO
1143 03A7 1906 A          JMP     $P1
1144 03A8 5001 A $CR:     LI      AC0,1
1145 03A9 D01D A          ST      AC0,OUTCOL
1146 03AA 1903 A          JMP     $P1
1147 03AB AC1D A $BS:     DSZ     OUTCOL          ; BACKSPACE: DECREMENT COUNT
1148 03AC 1901 A          JMP     $P1             ;    BUT NOT BELOW 1
1149 03AD 8C1D A          ISZ     OUTCOL
1150 03AE C01C A $P1:     LD      AC0,OUTDVC
1151 03AF 4406 A          BOC     BIT1,$PR
1152 03B0 4303 A          BOC     BIT0,$PT
1153 03B1 4506 A          BOC     NZRO,$PS
1154 03B2 C06D A          LD      AC0,TACO
1155 03B3 9913*A          JMP     PUTC            ; TTY
1156 03B4 C06D A $PT:     LD      AC0,TACO        ; PAPER TAPE - AVAILABLE SEPARAT
1157 03B5 9911*A          JMP     PUTC            ;    FOR POSSIBLE FUTURE CHANGES
1158 03B6 C06D A $PR:     LD      AC0,TACO
1159 03B7 194A A          JMP     HSPRT           ; HIGH SPEED PRINTER
1160 03B8 F1BB*A $PS:     SKNE    AC0,=8
1161 03B9 1904 A          JMP     $PD
1162 03BA C06D A          LD      AC0,TACO
1163 03BB 9498*A          JSR     PUTS            ; STRING (FOR FUNCTION)
1164 03BC 5C00 A          NOP
1165 03BD 8000 A          RTS     0
1166 03BE C06D A $PD:     LD      AC0,TACO
1167 03BF 9499*A          JSR     SAVE
1168 03C0 C8CA*A          LD      AC2,=FST        ; DISK FILE OUTPUT
1169 03C1 9506*A          JSR     WCHAR
1170 03C2 98CB*A          JMP     DSKERR
1171 03C3 989A*A          JMP     RESTOR
1172 03C4 7ECC A          .POOL   6
     03C5 000C A
     03C6 0060 A
     03C7 7E44 A
     03C8 1AA7 T
     03C9 0000 A
1173  ;
1174  ;        XXXX ERROR IN NNNN
1175  ;                ERROR NUMBER IS IN AC0 ON ENTRY
1176  ;                LINE NUMBER IS IN CURLIN
1177  ;
1178 03CA 9499*A ERROR:   JSR     SAVE
1179 03CB 159A A          JSR     XCRLF           ; CR/LF IF NOT AT START OF LINE
1180 03CC 5101 A          LI      AC1,1
1181 03CD D419 A          ST      AC1,ERRFLG      ; SET ERRFLG TO SIGNAL ERROR
1182 03CE C41C A          LD      AC1,OUTDVC      ; CHECK OUTPUT DEVICE -
1183 03CF 5200 A          LI      AC2,0           ;    SET TO TTY
```

```
1184 03D0 F4CD*A          SKNE    AC1,=2          ;     IF NOT PRINTER
1185 03D1 D81C A          ST      AC2,OUTDVC
1186 03D2 5F00 A          RCPY    AC0,AC3         ; GET ADDRESS OF ERROR NAME
1187 03D3 EDF5*A          ADD     AC3,=ERRTAB
1188 03D4 C300 A          LD      AC0,(AC3)
1189 03D5 1517 A          JSR     OUT2
1190 03D6 C301 A          LD      AC0,1(AC3)
1191 03D7 1515 A          JSR     OUT2
1192 03D8 1538 A          JSR     MESG            ; ' ERROR'
1193 03D9 1886 T          .WORD   ERM1
1194 03DA 190A A          JMP     INLINE
1195  ;
1196 03DB 9499*A WARNING: JSR     SAVE
1197 03DC 1589 A          JSR     XCRLF           ; CR/LF IF NOT AT START OF LINE
1198 03DD 5F00 A          RCPY    AC0,AC3         ; GET ADDRESS OF ERROR NAME
1199 03DE EDEA*A          ADD     AC3,=ERRTAB
1200 03DF C300 A          LD      AC0,(AC3)
1201 03E0 150C A          JSR     OUT2
1202 03E1 C301 A          LD      AC0,1(AC3)
1203 03E2 150A A          JSR     OUT2
1204 03E3 152D A          JSR     MESG            ; ' WARNING'
1205 03E4 1881 T          .WORD   WARNM
1206  ;
1207 03E5 C014 A INLINE:  LD      AC0,CURLIN      ; TYPE LINE NO.  IF NON-ZERO
1208 03E6 4104 A          BOC     ZR0,$NOLIN
1209 03E7 1529 A          JSR     MESG            ; 'IN ...'
1210 03E8 188A T          .WORD   ERM2
1211 03E9 C414 A          LD      AC1,CURLIN
1212 03EA 158A A          JSR     TYPLIN
1213 03EB 949F*A $NOLIN:  JSR     CRLF            ; PRINT CR/LF AND THEN RTS
1214 03EC 989A*A          JMP     RESTOR          ; (SAVED BY BREAK/STOP/ERROR/WAR
1215  ;
1216 03ED 2010 A OUT2:    ROL     AC0,8,0         ; SEND TWO CHARACTERS FROM AC0
1217 03EE 15A0 A          JSR     OUT1            ;     TO OUTPUT DEVICE
1218 03EF 2010 A          ROL     AC0,8,0
1219 03F0 199E A          JMP     OUT1
1220  ;
1221  ;       PUTW  -  PRINT (AC2) IN HEX WITH LEADING BLANK
1222  ;       PUTW4 -  PRINT AC2 IN HEX, NO LEADING BLANK
1223  ;                (USED BY HEX$ FUNCTION ONLY)
1224  ;
1225 03F1 1576 A PUTW4:   JSR     SAVE            ; SAVE REGISTERS
1226 03F2 1903 A          JMP     $PW1            ; PRINT 4 HEX DIGITS
1227 03F3 1574 A PUTW:    JSR     SAVE            ; SAVE REGISTERS
1228 03F4 5020 A          LI      AC0,BLANK       ; PRINT 1 BLANK FIRST
1229 03F5 1599 A          JSR     OUT1
1230 03F6 5104 A $PW1:    LI      AC1,4
1231 03F7 5CC0 A $PW:     RCPY    AC2,AC0         ; CONVERT 4 BITS TO ASCII
1232 03F8 2A08 A          SHL     AC2,4,0
1233 03F9 2C18 A          SHR     AC0,12,0
1234 03FA 7830 A          AISZ    AC0,'0'/256
1235 03FB 9D12*A          SKG     AC0,='9'/256
1236 03FC 1901 A          JMP     .+2
1237 03FD 7807 A          AISZ    AC0,'A'-'9'/256-1
```

```
1238 03FE 1590 A          JSR     OUT1
1239 03FF 79FF A          AISZ    AC1,-1
1240 0400 19F6 A          JMP     $PW
1241 0401 1977 A          JMP     RESTOR          ; RESTORE REG AND RETURN
1242  ;
1243  ;        SEND CHARACTER IN BITS 0-7 OF AC0 (AND TAC0)
1244  ;        TO THE HIGH SPEED PRINTER (CENTRONICS 306)
1245  ;
1246 0402 1565 A HSPRT:   JSR     SAVE            ; WRITE 1 ASCII CHAR ON PRINTER
1247 0403 CD0B*A          LD      AC3,=08048
1248 0404 C301 A $L1:     LD      AC0,1(AC3)      ; READ STATUS
1249 0405 A906 A          AND     AC0,$8006       ; TEST ONLY FAULT*,PRP*,WAIT*
1250 0406 C506 A          LD      AC1,$8002       ; COMPARE TO EXPECTED STATUS
1251 0407 5840 A          RXOR    AC1,AC0
1252 0408 45FB A          BOC     NZR0,$L1
1253 0409 C06D A          LD      AC0,TAC0        ; RELOAD CHARACTER TO SEND
1254 040A D307 A          ST      AC0,7(AC3)      ; SEND THE CHARACTER
1255 040B 196D A          JMP     RESTOR          ; RESTORE AC3 AND RETURN
1256  ;
1257 040C 8006 A $8006:   .WORD   08006
1258 040D 8002 A $8002:   .WORD   08002
1259 040E 0039 A          .POOL   3
     040F 8048 A
     0410 0000 A
1260  ;
1261  ;
1262  ;
1263  ;        MESSAGE PRINTING ROUTINES
1264  ;
1265  ;        JSR     MESG            JUMP TO SUBROUTINE
1266  ;        .WORD   MSG             MESSAGE ADDRESS
1267  ;        <RETURN HERE>
1268  ;
1269  ;        LD      AC3,MSGAD       MESSAGE ADDRESS IN AC3
1270  ;        JSR     MESG3
1271  ;        <RETURN HERE>
1272  ;
1273  ; NOTE:  THIS VERSION DOES NOT SEND CARRIAGE RETURN
1274  ;        OR LINE FEED UNLESS THEY ARE IN THE BUFFER.
1275  ;        THE MESSAGE IS TERMINATED BY A NULL BYTE.
1276  ;
1277                    .LOCAL
1278 0411 1556 A MESG:    JSR     SAVE            ; SAVE REGISTERS
1279 0412 6700 A          PULL    AC3             ; GET RETURN ADDRESS
1280 0413 ECDD*A          ADD     AC3,=1          ; INCREMENT RETURN ADDRESS
1281 0414 6300 A          PUSH    AC3             ;    TO SKIP PARAMETER
1282 0415 CFFF A          LD      AC3,-1(AC3)     ; GET MESSAGE ADDRESS
1283 0416 1901 A          JMP     $M1
1284 0417 1550 A MESG3:   JSR     SAVE
1285 0418 C300 A $M1:     LD      AC0,(AC3)
1286 0419 2C10 A          SHR     AC0,8,0         ; CHECK LEFT BYTE
1287 041A 4107 A          BOC     ZR0,$MSGEND
1288 041B 949E*A          JSR     OUT1
1289 041C C300 A          LD      AC0,(AC3)       ; DO RIGHT BYTE
```

```
1290 041D A8D5*A          AND     AC0,=0FF
1291 041E 4103 A          BOC     ZR0,$MSGEND
1292 041F 949E*A          JSR     OUT1
1293 0420 7B01 A          AISZ    AC3,1
1294 0421 19F6 A          JMP     $M1
1295 0422 1956 A $MSGEND: JMP     RESTOR
1296  ;
1297 0423 0000 A          .POOL   4
     0424 0000 A
     0425 0000 A
     0426 0000 A
```

```
1298                        .PAGE   'STACK INTERRUPTS'
1299  ;
1300  ;         STKINT: STACK INTERRUPT HANDLER
1301  ;         STKINT MAINTAINS THE SOFTWARE STACK
1302  ;         AND CHECKS FOR UNDERFLOW/OVERFLOW
1303  ;
1304 0427 D02A A STKINT: ST      AC0,SK0       ; SAVE REGISTERS
1305 0428 D42B A         ST      AC1,SK1
1306 0429 D82C A         ST      AC2,SK2
1307 042A 6400 A         PULL    AC0
1308 042B D02D A         ST      AC0,SKRET
1309 042C 4013 A         BOC     SFL,$FULL     ; STACK FULL OR EMPTY ?
1310 042D C022 A         LD      AC0,STKPTR    ; STACK EMPTY
1311 042E 9DE1*A         SKG     AC0,=STACK+3  ; CHECK FOR UNDERFLOW
1312 042F 190E A         JMP     $UNFL
1313 0430 5104 A         LI      AC1,4         ; RESTORE 4 WORDS
1314 0431 AC22 A $EMP:   DSZ     STKPTR
1315 0432 A022 A         LD      AC0,@STKPTR
1316 0433 6000 A         PUSH    AC0
1317 0434 79FF A         AISZ    AC1,-1
1318 0435 19FB A         JMP     $EMP
1319 0436 C02D A $REST:  LD      AC0,SKRET     ; RESTORE REGISTERS AND
1320 0437 6000 A         PUSH    AC0           ;    RE-ENABLE INTERRUPT
1321 0438 C02A A         LD      AC0,SK0
1322 0439 C42B A         LD      AC1,SK1
1323 043A C82C A         LD      AC2,SK2
1324 043B 3100 A         PFLG    1
1325 043C 3180 A         SFLG    1
1326 043D 7C00 A         RTI
1327 043E 151A A $UNFL:  JSR     CLRSTK        ; *** UNDERFLOW ***
1328 043F 94C9*A         JSR     SYSERR
1329  ;
1330 0440 5105 A $FULL:  LI      AC1,5         ; STACK FULL:
1331 0441 C9E1*A         LD      AC2,=STK5     ; REMOVE 5 WORDS
1332 0442 6400 A $LP1:   PULL    AC0
1333 0443 D200 A         ST      AC0,(AC2)
1334 0444 7A01 A         AISZ    AC2,1
1335 0445 79FF A         AISZ    AC1,-1
1336 0446 19FB A         JMP     $LP1
1337 0447 5104 A         LI      AC1,4         ; PUT LAST 4 WORDS ON
1338 0448 6400 A $LP2:   PULL    AC0           ;    SOFTWARE STACK
1339 0449 B022 A         ST      AC0,@STKPTR
1340 044A 8C22 A         ISZ     STKPTR
1341 044B C022 A         LD      AC0,STKPTR    ; CHECK FOR OVERFLOW
1342 044C 9DD7*A         SKG     AC0,=STACK+20
1343 044D 1902 A         JMP     $OK
1344 044E 150A A         JSR     CLRSTK        ; *** OVERFLOW ***
1345 044F 94C9*A         JSR     SYSERR
1346 0450 79FF A $OK:    AISZ    AC1,-1
1347 0451 19F6 A         JMP     $LP2
1348 0452 5105 A         LI      AC1,5         ; RESTORE TOP 5 WORDS
1349 0453 7AFF A $LP3:   AISZ    AC2,-1
1350 0454 C200 A         LD      AC0,(AC2)
1351 0455 6000 A         PUSH    AC0
```

```
1352 0456 79FF A          AISZ    AC1,-1
1353 0457 19FB A          JMP     $LP3
1354 0458 19DD A          JMP     $REST
1355  ;
1356  ;        CLEAR STACKS (HARDWARE STACK AND SOFTWARE EXTENSION)
1357  ;
1358  ;        WARNING:  CALL ONLY FROM MAIN PROGRAM!
1359  ;                  REGISTERS ARE NOT SAVED.
1360  ;
1361 0459 3100 A CLRSTK: PFLG    1              ; TURN OFF INTERRUPTS
1362 045A 3900 A          PFLG    9
1363 045B 6600 A          PULL    AC2            ; PULL RETURN ADDRESS
1364 045C 5109 A          LI      AC1,9          ; CLEAR HARDWARE STACK
1365 045D 6400 A          PULL    AC0
1366 045E 79FF A          AISZ    AC1,-1
1367 045F 19FD A          JMP     .-2
1368 0460 C1C4*A          LD      AC0,=STKINT    ; RESTORE POINTERS
1369 0461 D002 A          ST      AC0,2
1370 0462 C1C3*A          LD      AC0,=STACK
1371 0463 D022 A          ST      AC0,STKPTR
1372 0464 6200 A          PUSH    AC2            ; DUMMY WORD
1373 0465 3180 A          SFLG    1              ; RE-ENABLE INTERRUPTS
1374 0466 3980 A          SFLG    9
1375 0467 1A00 A          JMP     (AC2)
1376  ;
```

```
1377                        .PAGE   'SOFTWARE STACK ROUTINES'
1378  ;
1379  ;         SAVE ALL 4 REGISTERS ON SOFTWARE STACK
1380  ;
1381 0468 D06D A.SAVE:    ST      AC0,TAC0      ; SAVE AC0 TEMPORARILY
1382 0469 C023 A          LD      AC0,STPTR
1383 046A 9D1C A          SKG     AC0,$ST1L     ; CHECK FOR POSSIBLE OVERFLOW
1384 046B 1901 A          JMP     .+2
1385 046C 1917 A          JMP     $SOVF
1386 046D C06D A          LD      AC0,TAC0
1387 046E B023 A          ST      AC0,@STPTR
1388 046F 5CC0 A          RCPY    AC3,AC0
1389 0470 CC23 A          LD      AC3,STPTR
1390 0471 D701 A          ST      AC1,1(AC3)
1391 0472 DB02 A          ST      AC2,2(AC3)
1392 0473 D303 A          ST      AC0,3(AC3)
1393 0474 7B04 A          AISZ    AC3,4
1394 0475 DC23 A          ST      AC3,STPTR
1395 0476 5F00 A          RCPY    AC0,AC3
1396 0477 C06D A          LD      AC0,TAC0
1397 0478 8000 A          RTS     0
1398  ;
1399  ;         RESTORE 4 REGISTERS FROM SOFTWARE STACK
1400  ;
1401 0479 C023 A RESTOR:  LD      AC0,STPTR     ; CHECK FOR UNDERFLOW
1402 047A 9D0B A          SKG     AC0,$ST1B
1403 047B 1909 A          JMP     $SUNF
1404 047C CC23 A          LD      AC3,STPTR
1405 047D 7BFC A          AISZ    AC3,-4
1406 047E DC23 A          ST      AC3,STPTR
1407 047F C300 A          LD      AC0,0(AC3)
1408 0480 C701 A          LD      AC1,1(AC3)
1409 0481 CB02 A          LD      AC2,2(AC3)
1410 0482 CF03 A          LD      AC3,3(AC3)
1411 0483 8000 A          RTS     0
1412  ;
1413 0484 94C9*A $SOVF:   JSR     SYSERR        ; STACK ERROR #1
1414 0485 94C9*A $SUNF:   JSR     SYSERR        ; STACK ERROR #2
1415  ;
1416 0486 19C6 T $ST1B:   .WORD   STACK0+3      ; ADDRESSES FOR COMPARING WHEN
1417 0487 19DF T $ST1L:   .WORD   STACK0+28     ;    STORING 4 LOCATIONS
1418  ;
1419  ;         PUSH,PULL:  PUSH OR PULL FROM A SOFTWARE STACK
1420  ;                   AC0 = VALUE LOCATION BEFORE OR AFTER
1421  ;                   AC3 = STACK NUMBER (0=REGISTER STACK)
1422  ;
1423  ;                   RETURN:  RTS 0 -- ERROR (EMPTY OR OVERFLOW)
1424  ;                            RTS 1 -- NORMAL
1425  ;
1426 0488 D06D A PUSH:    ST      AC0,TAC0      ; CHECK FOR SOFTWARE STACK
1427 0489 DC70 A          ST      AC3,TAC3      ;    OVERFLOW BEFORE PUSHING
1428 048A C323 A          LD      AC0,STPTR(AC3)
1429 048B ED13 A          ADD     AC3,STLAST
1430 048C 9F00 A          SKG     AC0,(AC3)
```

```
1431 048D 1901 A          JMP     .+2
1432 048E 1905 A          JMP     $PERR         ; ERROR EXIT - OVERFLOW
1433 048F C06D A          LD      AC0,TAC0
1434 0490 CC70 A          LD      AC3,TAC3
1435 0491 B323 A          ST      AC0,@STPTR(AC3)
1436 0492 8F23 A          ISZ     STPTR(AC3)
1437 0493 8001 A          RTS     1
1438 0494 CC70 A $PERR:   LD      AC3,TAC3      ; PUSH/PULL ERROR - RESTORE AC3
1439 0495 8000 A          RTS     0
1440  ;
1441 0496 C323 A PULL:    LD      AC0,STPTR(AC3)  ; CHECK FOR UNDERFLOW
1442 0497 DC70 A          ST      AC3,TAC3
1443 0498 ED0E A          ADD     AC3,STFIRST
1444 0499 9F00 A          SKG     AC0,(AC3)
1445 049A 19F9 A          JMP     $PERR         ; ERROR EXIT - STACK IS EMPTY
1446 049B CC70 A          LD      AC3,TAC3
1447 049C AF23 A          DSZ     STPTR(AC3)
1448 049D A323 A          LD      AC0,@STPTR(AC3)
1449 049E 8001 A          RTS     1
1450  ;
1451 049F 04A0 T STLAST:  .WORD   .+1
1452 04A0 19E2 T          .WORD   STACK1-1      ; LAST WORD OF EACH BUFFER
1453 04A1 19F1 T          .WORD   STACK2-1
1454 04A2 1A0F T          .WORD   STACK3-1
1455 04A3 1A23 T          .WORD   STACK4-1
1456 04A4 1A2D T          .WORD   STACK5-1
1457 04A5 1A37 T          .WORD   STACK6-1
1458 04A6 1A41 T          .WORD   STACKX-1
1459  ;
1460 04A7 04A8 T STFIRST: .WORD   .+1
1461 04A8 19C3 T          .WORD   STACK0        ; FIRST WORD OF EACH STACK
1462 04A9 19E3 T          .WORD   STACK1
1463 04AA 19F2 T          .WORD   STACK2
1464 04AB 1A10 T          .WORD   STACK3
1465 04AC 1A24 T          .WORD   STACK4
1466 04AD 1A2E T          .WORD   STACK5
1467 04AE 1A38 T          .WORD   STACK6
1468  ;
```

```
1469                        .PAGE   'EDITOR ROUTINES'
1470                        .LOCAL
1471   ;
1472   ;        ESTOR - PUT PBUF LINE INTO EDIT MEMORY
1473   ;
1474 04AF C8B2*A ESTOR:   LD      AC2,=PBUF     ; GET BUFFER ADDRESS
1475 04B0 5100 A          LI      AC1,0
1476 04B1 9497*A          JSR     GETS          ; GET NEXT CHAR
1477 04B2 F0D0*A          SKNE    AC0,=0D       ; CHECK IF CARRIAGE RETURN
1478 04B3 1903 A          JMP     EDEL          ;    YES - DELETE LINE
1479 04B4 7AFF A          AISZ    AC2,-1        ; POINT AC2 TO 'LINE NO.' (FAKE
1480 04B5 1579 A          JSR     FINDCR        ; DETERMINE LINE LENGTH
1481 04B6 1901 A          JMP     .+2           ; AC1 NOW HAS LENGTH
1482 04B7 5100 A EDEL:    LI      AC1,0
1483 04B8 D436 A          ST      AC1,ELEN
1484 04B9 C414 A          LD      AC1,CURLIN
1485 04BA 154F A          JSR     EFIND         ; FIND MATCHING LINE #
1486 04BB 1932 A          JMP     $NOTF
1487 04BC D837 A          ST      AC2,NXTLIN    ; LINE FOUND - REPLACE OR DELETE
1488 04BD C300 A $DN:     LD      AC0,(AC3)     ; MOVE UPPER STORAGE DOWN
1489 04BE D200 A          ST      AC0,(AC2)
1490 04BF FC35 A          SKNE    AC3,EEND
1491 04C0 1903 A          JMP     $IN1
1492 04C1 7A01 A          AISZ    AC2,1
1493 04C2 7B01 A          AISZ    AC3,1
1494 04C3 19F9 A          JMP     $DN
1495 04C4 D835 A $IN1:    ST      AC2,EEND      ; SAVE NEW VALUE OF EEND
1496 04C5 C033 A $IN:     LD      AC0,ESTART    ; RESET ECUR TO BEGINNING
1497 04C6 D034 A          ST      AC0,ECUR      ;    (WILL REMAIN IF DELETING LI
1498 04C7 C038 A          LD      AC0,LAST      ; CLEAR SYMBOL TABLE
1499 04C8 D051 A          ST      AC0,SY2
1500 04C9 C036 A          LD      AC0,ELEN      ; STORE NEW LINE
1501 04CA 411F A          BOC     ZR0,$FIN      ; ZERO LENGTH - DONE
1502 04CB C035 A          LD      AC0,EEND      ; MOVE UP (ELEN) WORDS
1503 04CC E036 A          ADD     AC0,ELEN      ; CHECK IF ROOM IN MEMORY
1504 04CD C451 A          LD      AC1,SY2
1505 04CE 1558 A          JSR     SKL
1506 04CF 1920 A          JMP     $STERR
1507 04D0 C835 A          LD      AC2,EEND
1508 04D1 CC35 A          LD      AC3,EEND
1509 04D2 EC36 A          ADD     AC3,ELEN
1510 04D3 C200 A $UP:     LD      AC0,(AC2)     ; MOVE UPPER STORAGE UP
1511 04D4 D300 A          ST      AC0,(AC3)
1512 04D5 7BFF A          AISZ    AC3,-1
1513 04D6 F837 A          SKNE    AC2,NXTLIN    ; CHECK IF DONE
1514 04D7 1902 A          JMP     $STORE
1515 04D8 7AFF A          AISZ    AC2,-1
1516 04D9 19F9 A          JMP     $UP
1517 04DA C035 A $STORE:  LD      AC0,EEND      ; UPDATE END ADDRESS
1518 04DB E036 A          ADD     AC0,ELEN
1519 04DC D035 A          ST      AC0,EEND
1520 04DD CCB2*A          LD      AC3,=PBUF     ; COPY LINE INTO MEMORY
1521 04DE C014 A          LD      AC0,CURLIN
1522 04DF D200 A          ST      AC0,(AC2)
```

```
1523 04E0 D834 A          ST      AC2,ECUR      ; UPDATE CURRENT LOCATION
1524 04E1 7A01 A          AISZ    AC2,1
1525 04E2 C436 A          LD      AC1,ELEN      ; (ELEN >= 2)
1526 04E3 79FF A          AISZ    AC1,-1
1527 04E4 C300 A $ST:     LD      AC0,(AC3)
1528 04E5 D200 A          ST      AC0,(AC2)
1529 04E6 7A01 A          AISZ    AC2,1
1530 04E7 7B01 A          AISZ    AC3,1
1531 04E8 79FF A          AISZ    AC1,-1
1532 04E9 19FA A          JMP     $ST
1533 04EA C035 A $FIN:    LD      AC0,EEND
1534 04EB D055 A          ST      AC0,ANEXT
1535 04EC D056 A          ST      AC0,TNEXT
1536 04ED 8000 A          RTS
1537  ;
1538 04EE D837 A $NOTF:   ST      AC2,NXTLIN    ; NOT FOUND:  INSERT OR IGNORE
1539 04EF 19D5 A          JMP     $IN
1540  ;
1541 04F0 5000 A $STERR:  LI      AC0,0         ; OUT OF STORAGE
1542 04F1 98C5*A          JMP     RUNERR        ;    LINE NOT SAVED
1543  ;
1544  ;
1545  ;
1546  ;        DELETE COMMAND PROCESSING - DELETE A LINE RANGE
1547  ;
1548 04F2 5000 A DELETE:  LI      AC0,0         ; LINE LENGTH = 0 (NO REPLACE)
1549 04F3 D036 A          ST      AC0,ELEN
1550 04F4 C060 A          LD      AC0,OP3       ; CHECK # OF ARGUMENTS
1551 04F5 4502 A          BOC     NZRO,$DEL1
1552 04F6 5008 A          LI      AC0,8         ; NO ARG - SYNTAX ERROR
1553 04F7 98C5*A          JMP     RUNERR
1554 04F8 78FF A $DEL1:   AISZ    AC0,-1        ; 1 ARG - DELETE ONLY ONE LINE
1555 04F9 1902 A          JMP     $DEL2
1556 04FA C062 A          LD      AC0,LIST1
1557 04FB D063 A          ST      AC0,LIST2
1558 04FC C062 A $DEL2:   LD      AC0,LIST1     ; CHECK TO BE SURE LIST1<=LIST2
1559 04FD C463 A          LD      AC1,LIST2
1560 04FE 1528 A          JSR     SKL
1561 04FF D063 A          ST      AC0,LIST2
1562 0500 5D00 A          RCPY    AC0,AC1       ; FIND LOCATION OF FIRST LINE TO
1563 0501 1508 A          JSR     EFIND         ;    BE DELETED
1564 0502 5C00 A          NOP
1565 0503 D862 A          ST      AC2,LIST1
1566 0504 C463 A          LD      AC1,LIST2     ; FIND LAST LOCATION TO DELETE
1567 0505 1504 A          JSR     EFIND
1568 0506 5F80 A          RCPY    AC2,AC3
1569 0507 C862 A          LD      AC2,LIST1
1570 0508 D837 A          ST      AC2,NXTLIN
1571 0509 19B3 A          JMP     $DN           ; MOVE UPPER MEMORY DOWN
1572  ;
1573  ;
1574  ;
1575  ;        FIND LINE IN PROGRAM AREA
1576  ;
```

```
1577  ;         JSR     EFIND
1578  ;         <LINE NOT FOUND>  AC2 = LOC TO INSERT
1579  ;         <LINE FOUND>      AC2 = ADDRESS OF LINE
1580  ;                           AC3 = ADDRESS OF NEXT LINE
1581  ;                           AC1 = WORD COUNT OF LINE
1582  ;
1583  ;         A MATCH IS ATTEMPTED FOR THE LINE NUMBER IN (AC1)
1584  ;
1585        007B A $LINE    =       SCR             ; USE SCRATCH SPACE TO HOLD LINE
1586  ;
1587 050A D47B A EFIND:   ST      AC1,$LINE       ; SAVE LINE NUMBER TO BE FOUND
1588 050B C835 A          LD      AC2,EEND        ; CHECK IF ANY PROGRAM EXISTS
1589 050C F833 A          SKNE    AC2,ESTART      ;    NO PROGRAM - INSERT AT END
1590 050D 8000 A          RTS     0
1591 050E C834 A          LD      AC2,ECUR        ; START INITIAL SEARCH FROM CURR
1592 050F F835 A          SKNE    AC2,EEND
1593 0510 1907 A          JMP     $FOR
1594 0511 C200 A          LD      AC0,(AC2)       ; CHECK LINE NUMBER OF CURRENT L
1595 0512 F07B A          SKNE    AC0,$LINE       ; EQUAL LINE# - RETURN DATA
1596 0513 1910 A          JMP     $FND
1597 0514 C47B A          LD      AC1,$LINE       ; CHECK IF CURRENT < $LINE
1598 0515 1511 A          JSR     SKL
1599 0516 1901 A          JMP     $FOR            ;    NO - START SCAN FROM BEGINN
1600 0517 1901 A          JMP     $LOOP           ;    YES - START SCAN FROM CURRE
1601 0518 C833 A $FOR:    LD      AC2,ESTART      ; SCAN FORWARD FROM START
1602 0519 F835 A $LOOP:   SKNE    AC2,EEND
1603 051A 8000 A          RTS     0               ; NO LINES >= $LINE IN MEMORY
1604 051B C200 A          LD      AC0,(AC2)
1605 051C F07B A          SKNE    AC0,$LINE
1606 051D 1906 A          JMP     $FND            ; LINE FOUND
1607 051E C47B A          LD      AC1,$LINE
1608 051F 1507 A          JSR     SKL
1609 0520 8000 A          RTS     0               ; FOUND LINE WITH HIGHER LINE NU
1610 0521 150D A          JSR     FINDCR          ; TRY NEXT LINE
1611 0522 5EC0 A          RCPY    AC3,AC2
1612 0523 19F5 A          JMP     $LOOP
1613 0524 D834 A $FND:    ST      AC2,ECUR        ; LINE FOUND - RETURN WITH DATA
1614 0525 1509 A          JSR     FINDCR
1615 0526 8001 A          RTS     1
1616  ;
1617  ;         SKL     SKIP IF (AC0) < (AC1)
1618  ;                 TEST IS ON 16-BIT POSITIVE MAGNITUDE
1619  ;
1620 0527 D06D A SKL:     ST      AC0,TAC0        ; SAVE REGISTERS
1621 0528 D46E A          ST      AC1,TAC1
1622 0529 3780 A          SFLG    CARRY
1623 052A 906E A          SUBB    AC0,TAC1        ; TEST IS ON AC0-AC1 (MAY SET CA
1624 052B C06D A          LD      AC0,TAC0        ; RESTORE AC0
1625 052C 4A01 A          BOC     CRY,.+2
1626 052D 8001 A          RTS     1               ; IF CARRY NOT SET, AC0<AC1
1627 052E 8000 A          RTS     0               ; IF CARRY IS  SET, AC0>=AC1
1628  ;
1629  ;         FINDCR - FIND CARRIAGE RETURN
1630  ;         ON ENTRY: AC2 = ADDR OF 1ST WORD OF LINE (LINE NO.)
```

```
1631  ;        ON EXIT:  AC2 PRESERVED; AC3 = ADDR OF NEXT LINE
1632  ;                  AC1 = WORD COUNT OF LINE (INCLUDING LINE NO.)
1633  ;
1634 052F 5F80 A FINDCR: RCPY    AC2,AC3       ; USE AC3 FOR SCAN ADDRESS
1635 0530 7B01 A.$1:     AISZ    AC3,1         ; CHECK NEXT LINE (SKIP LINE NO.
1636 0531 1507 A         JSR     CKCR
1637 0532 1901 A         JMP     $2
1638 0533 19FC A         JMP     $1
1639 0534 7B01 A $2:     AISZ    AC3,1         ; AC3 = ADDR OF NEXT LINE
1640 0535 5D80 A         RCPY    AC2,AC1       ; AC1 = (AC3)-(AC2)
1641 0536 7101 A         CAI     AC1,1
1642 0537 69C0 A         RADD    AC3,AC1
1643 0538 8000 A         RTS     0
1644  ;
1645  ;        CKCR - CHECK IF 0(AC3) HAS CARRIAGE RETURN IN EITHER BYTE
1646  ;                 RTS 0 IF TRUE; RTS 1 IF NO CR
1647  ;
1648 0539 C300 A CKCR:    LD      AC0,(AC3)     ; CHECK LEFT BYTE
1649 053A 2C10 A         SHR     AC0,8,0
1650 053B F0D0*A         SKNE    AC0,=0D
1651 053C 8000 A         RTS     0
1652 053D C300 A         LD      AC0,(AC3)     ; CHECK RIGHT BYTE
1653 053E A8D5*A         AND     AC0,=0FF
1654 053F F0D0*A         SKNE    AC0,=0D
1655 0540 8000 A         RTS     0
1656 0541 8001 A         RTS     1             ; GO ON TO NEXT WORD
1657  ;
```

```
1658                    .PAGE   'SUBSTRING ROUTINES'
1659                    .LOCAL
1660  ;
1661  ;       PUTS  -  ADD A CHARACTER TO A STRING
1662  ;          .
1663  ;       STRING ORIGIN = (SSTART)
1664  ;       CHARACTER NO.  = SCOL (INCREMENTED ON EXIT)
1665  ;       AC0 - CHARACTER IN BITS 0-7
1666  ;
1667  ;       RTS 0 -- STRING OVERFLOW (CHARACTER NOT STORED)
1668  ;       RTS 1 -- NORMAL RETURN
1669  ;
1670 0542 D86F A PUTS:   ST      AC2,TAC2      ; SAVE REGISTERS
1671 0543 D46E A         ST      AC1,TAC1
1672 0544 D06D A         ST      AC0,TAC0
1673 0545 C857 A         LD      AC2,SSTART
1674 0546 C458 A         LD      AC1,SCOL
1675 0547 F459 A         SKNE    AC1,MAXSTR    ; CHECK IF ALREADY AT MAX LENGTH
1676 0548 1918 A         JMP     $RST          ;    IF SO, THEN SKIP THIS ROUTI
1677 0549 2D02 A         SHR     AC1,1,0       ; CALCULATE ACTUAL WORD ADDRESS
1678 054A 6A40 A         RADD    AC1,AC2
1679 054B C458 A         LD      AC1,SCOL
1680 054C A8D5*A         AND     AC0,=0FF      ; MASK CHARACTER TO 8 BITS
1681 054D 6D00 A         RXCH    AC0,AC1
1682 054E 4306 A         BOC     BIT0,$RIGHT   ; DETERMINE RIGHT OR LEFT
1683 054F 6D00 A         RXCH    AC0,AC1
1684 0550 2810 A         SHL     AC0,8,0
1685 0551 D070 A         ST      AC0,$TEMP
1686 0552 C200 A         LD      AC0,(AC2)
1687 0553 A8D5*A         AND     AC0,=0FF      ; ZERO POSITION OF NEW CHARACTER
1688 0554 1905 A         JMP     $BOTH
1689 0555 6D00 A $RIGHT: RXCH    AC0,AC1
1690 0556 A8D5*A         AND     AC0,=0FF
1691 0557 D070 A         ST      AC0,$TEMP
1692 0558 C200 A         LD      AC0,(AC2)
1693 0559 A8DA*A         AND     AC0,=0FF00
1694 055A A470 A $BOTH:  OR      AC0,$TEMP     ; MERGE NEW CHARACTER INTO WORD
1695 055B D200 A         ST      AC0,(AC2)     ; SAVE BACK INTO BUFFER
1696 055C 8C58 A         ISZ     SCOL
1697 055D C06D A         LD      AC0,TAC0      ; RESTORE REGISTERS
1698 055E C46E A         LD      AC1,TAC1
1699 055F C86F A         LD      AC2,TAC2
1700 0560 8001 A         RTS     1             ; NORMAL PUTS RETURN
1701 0561 C06D A $RST:   LD      AC0,TAC0      ; RESTORE REGISTERS
1702 0562 C46E A         LD      AC1,TAC1
1703 0563 C86F A         LD      AC2,TAC2
1704 0564 8000 A         RTS     0
1705  ;
1706  ;       GETS  -  GET A CHARACTER FROM A PACKED STRING
1707  ;
1708  ;       AC2 = STRING ORIGIN
1709  ;       AC1 = CHARACTER NO. (0=LEFTMOST), INCREMENTED ON EXIT
1710  ;       AC0 = CHARACTER (RETURNED IN BITS 0-7)
1711  ;
```

```
1712  ;
1713 0565 D86F A GETS:   ST      AC2,TAC2      ; SAVE REGISTERS
1714 0566 D46E A         ST      AC1,TAC1
1715 0567 8C6E A         ISZ     TAC1          ; INCREMENT RETURNED AC1 VALUE
1716 0568 5C40 A GETS2:  RCPY    AC1,AC0
1717 0569 2D02 A         SHR     AC1,1,0       ; CALCULATE ACTUAL WORD ADDRESS
1718 056A 6A40 A         RADD    AC1,AC2
1719 056B C600 A         LD      AC1,(AC2)     ; GET 2 PACKED CHARACTERS
1720 056C 4301 A         BOC     BIT0,.+2
1721 056D 2D10 A         SHR     AC1,8,0       ; SHIFT RIGHT IF USING LEFT BYTE
1722 056E 5C40 A         RCPY    AC1,AC0
1723 056F A8D5*A         AND     AC0,=0FF      ; MASK TO RETURN ONLY RIGHT BYTE
1724 0570 D06D A         ST      AC0,TAC0
1725 0571 19EF A         JMP     $RST          ; RESTORE REGISTERS & RETURN
1726  ;
1727      0070 A $TEMP   =       TAC3
1728  ;
1729  ;       GETUC  -  GET 1 CHAR FROM UNPACKED INBUF FOR PRESCANNER
1730  ;                (CONVERT LOWER CASE TO UPPER CASE)
1731  ;                AC3 = NEXT WORD ADDRESS
1732  ;
1733  ;       GETCH  -  GET 1 CHAR FROM CURRENT LINE
1734  ;
1735  ;       LSTART = LINE ORIGIN
1736  ;       COL    = CHARACTER NO., INCREMENTED ON EXIT
1737  ;
1738  ;       RETURN:  RTS 0  IF A SPACE (OR TAB)
1739  ;                RTS 1  IF A DIGIT  (0-9)
1740  ;                RTS 2  IF A LETTER (A-Z)
1741  ;                RTS 3  OTHERWISE (PUNCTUATION, ETC.)
1742  ;
1743 0572 C300 A GETUC:  LD      AC0,(AC3)     ; READ UNPACKED CHAR (PRESCAN)
1744 0573 1515 A         JSR     CHKLC
1745 0574 7B01 A         AISZ    AC3,1
1746 0575 1907 A         JMP     GETS3
1747 0576 D46E A GETCH:  ST      AC1,TAC1      ; SAVE REGISTERS
1748 0577 D86F A         ST      AC2,TAC2
1749 0578 C43F A         LD      AC1,COL
1750 0579 8C3F A         ISZ     COL
1751 057A C83E A         LD      AC2,LSTART
1752 057B 15EC A         JSR     GETS2         ; GET CHAR
1753 057C 41F9 A         BOC     ZR0,GETCH     ; TRY AGAIN IF NULL (FILLER)
1754 057D F0D2*A GETS3:  SKNE    AC0,=BLANK    ; IS IT A SPACE?
1755 057E 8000 A         RTS     0             ;    YES
1756 057F F0CE*A         SKNE    AC0,=9        ; TAB ?
1757 0580 8000 A         RTS     0             ;    YES
1758 0581 9D0D*A         SKG     AC0,='9'/256  ; IS IT A DIGIT?
1759 0582 9D0D*A         SKG     AC0,='0'/256-1
1760 0583 1901 A         JMP     .+2           ;    NO
1761 0584 8001 A         RTS     1             ;    YES
1762 0585 9D0B*A         SKG     AC0,='Z'/256  ; IS IT A LETTER?
1763 0586 9D0B*A         SKG     AC0,='A'/256-1
1764 0587 8003 A         RTS     3             ;    NO
1765 0588 8002 A         RTS     2             ;    YES
```

```
1766  ;
1767 0589 A8D3*A CHKLC:  AND     AC0,=07F     ; CONVERT LOWER CASE CHAR TO UPP
1768 058A 9D08*A         SKG     AC0,=05F
1769 058B 1901 A         JMP     .+2          ; ALREADY UPPER CASE
1770 058C 78E0 A         AISZ    AC0,-020     ; CONVERT LOWER TO UPPER CASE
1771 058D D04F A         ST      AC0,PFLAG    ; SAVE CHAR (FOR PSCAN)
1772 058E 8000 A         RTS     0
1773  ;
1774 058F 0039 A         .POOL   5
     0590 002F A
     0591 005A A
     0592 0040 A
     0593 005F A
```

```
1775                        .PAGE   'PRESCANNER'
1776                        .LOCAL
1777  ;
1778  ;       PRESCAN - CONVERT ALL KEYWORDS TO SINGLE BYTE CODE
1779  ;
1780  ;               RTS 0 -- ERROR
1781  ;               RTS 1 -- BLANK LINE
1782  ;               RTS 2 -- NORMAL RETURN
1783  ;
1784  ;       LINE IS COPIED FROM INBUF TO PBUF
1785  ;
1786 0594 CC20 A PSCAN:  LD      AC3,ENDBUF    ; REMOVE EXCESS TRAILING BLANKS
1787 0595 1901 A         JMP     .+2
1788 0596 7BFF A $TRIM:  AISZ    AC3,-1
1789 0597 FD47*A         SKNE    AC3,=INBUF-1
1790 0598 8001 A         RTS     1             ; COMPLETELY BLANK LINE
1791 0599 C300 A         LD      AC0,(AC3)
1792 059A F0D2*A         SKNE    AC0,=BLANK
1793 059B 19FA A         JMP     $TRIM         ; IGNORE TRAILING BLANKS, TABS,
1794 059C F0CE*A         SKNE    AC0,=9
1795 059D 19F8 A         JMP     $TRIM
1796 059E F0D0*A         SKNE    AC0,=0D
1797 059F 19F6 A         JMP     $TRIM
1798 05A0 500D A         LI      AC0,0D        ; PUT CR AFTER LAST SIGNIF CHAR
1799 05A1 D301 A         ST      AC0,1(AC3)
1800 05A2 CCB1*A         LD      AC3,=INBUF    ; SET POINTERS TO COPY
1801 05A3 C0B2*A         LD      AC0,=PBUF     ;    FROM INBUF TO PBUF
1802 05A4 D057 A         ST      AC0,SSTART    ;    (USING GETUC & PUTS)
1803 05A5 C13A*A         LD      AC0,=133
1804 05A6 D059 A         ST      AC0,MAXSTR
1805 05A7 5100 A         LI      AC1,0
1806 05A8 D458 A         ST      AC1,SCOL
1807 05A9 D414 A         ST      AC1,CURLIN    ; INITIALIZE LINE NUMBER
1808 05AA 15C7 A $P0:    JSR     GETUC         ; LOOK FOR LINE NUMBER
1809 05AB 1903 A         JMP     $P00          ;    SPACE
1810 05AC 1905 A         JMP     $P1           ;    DIGIT
1811 05AD 1935 A         JMP     $PL1          ;    LETTER
1812 05AE 1934 A         JMP     $PL1          ;    PUNCTUATION
1813 05AF 1592 A $P00:   JSR     PUTS          ; SAVE SPACE FOR AUTO MODE
1814 05B0 5C00 A         NOP
1815 05B1 19F8 A         JMP     $P0
1816 05B2 6D00 A $P1:    RXCH    AC0,AC1       ; FORM LINE # - CHECK SIZE
1817 05B3 B8D9*A         SKAZ    AC0,=0F000
1818 05B4 1914 A         JMP     $VLERR
1819 05B5 6D00 A         RXCH    AC0,AC1
1820 05B6 2908 A         SHL     AC1,4,0
1821 05B7 A8D1*A         AND     AC0,=0F
1822 05B8 6900 A         RADD    AC0,AC1
1823 05B9 D414 A         ST      AC1,CURLIN
1824 05BA 5000 A         LI      AC0,0         ; RESET OUTPUT COLUMN
1825 05BB D058 A         ST      AC0,SCOL      ;    IN CASE SAVED SPACES
1826 05BC 15B5 A         JSR     GETUC         ; GET MORE DIGITS OR FOLLOWING C
1827 05BD 1903 A         JMP     $P2
1828 05BE 19F3 A         JMP     $P1
```

```
1829 05BF 1923 A          JMP     $PL1
1830 05C0 1922 A          JMP     $PL1
1831 05C1 15B0 A $P2:     JSR     GETUC          ; NEW MAIN ITEM: GET NEXT CHAR
1832 05C2 1903 A          JMP     $PP1
1833 05C3 1902 A          JMP     $PP1
1834 05C4 191E A          JMP     $PL1
1835 05C5 191D A          JMP     $PL1
1836 05C6 9498*A $PP1:    JSR     PUTS           ; SAVE SPACE/DIGIT
1837 05C7 5C00 A          NOP
1838 05C8 19F8 A          JMP     $P2
1839 05C9 500A A $VLERR:  LI      AC0,10         ; TOO MANY DIGITS IN NUMBER
1840 05CA 98C3*A          JMP     ERROR
1841 05CB C300 A $STRNG:  LD      AC0,(AC3)      ; COPY ALL OF STRING TO OUTPUT
1842 05CC 7B01 A          AISZ    AC3,1
1843 05CD 9498*A          JSR     PUTS
1844 05CE 5C00 A          NOP
1845 05CF F0D0*A          SKNE    AC0,=0D
1846 05D0 8002 A          RTS     2
1847 05D1 F10F*A          SKNE    AC0,=022
1848 05D2 19EE A          JMP     $P2
1849 05D3 19F7 A          JMP     $STRNG
1850 05D4 159D A $HEX:    JSR     GETUC          ; GET & SAVE HEX NUMBER
1851 05D5 19F0 A          JMP     $PP1
1852 05D6 1905 A          JMP     $HX2
1853 05D7 1901 A          JMP     $HX1
1854 05D8 190A A          JMP     $PL1
1855 05D9 9D08*A $HX1:    SKG     AC0,='F'/256   ; CHECK HEX LETTER (A-F)
1856 05DA 1901 A          JMP     $HX2
1857 05DB 1907 A          JMP     $PL1
1858 05DC 9498*A $HX2:    JSR     PUTS           ; SAVE HEX DIGIT/LETTER
1859 05DD 5C00 A          NOP
1860 05DE 19F5 A          JMP     $HEX
1861 05DF 18BD T          .POOL   4
     05E0 0085 A
     05E1 0022 A
     05E2 0046 A
1862  ;
1863 05E3 C458 A $PL1:    LD      AC1,SCOL       ; SAVE SCOL (START OF SEQUENCE)
1864 05E4 D47B A          ST      AC1,SCR
1865 05E5 9498*A          JSR     PUTS
1866 05E6 5C00 A          NOP
1867 05E7 F0D0*A          SKNE    AC0,=0D        ; CHECK IF CR
1868 05E8 8002 A          RTS     2              ;    YES - DONE
1869 05E9 F1F7*A          SKNE    AC0,=022       ; CHECK FOR STRINGS
1870 05EA 19E0 A          JMP     $STRNG
1871 05EB F172*A          SKNE    AC0,=REMCHAR   ; CHECK IF REMARK CHARACTER
1872 05EC 9972*A          JMP     $REM
1873 05ED F172*A          SKNE    AC0,='$'/256   ; CHECK IF HEX NUMBER
1874 05EE 19E5 A          JMP     $HEX
1875 05EF 2810 A          SHL     AC0,8,0        ; CHECK FOR KEYWORD OR 2-CHAR PU
1876 05F0 D045 A $PL1A:   ST      AC0,IDENT      ; INITIALIZE IDENT
1877 05F1 5000 A          LI      AC0,0
1878 05F2 D046 A          ST      AC0,IDENT+1
1879 05F3 956D*A          JSR     GETUC          ; CHECK FOR 2ND CHAR
```

```
1880 05F4 19D1 A          JMP     $PP1
1881 05F5 19D0 A          JMP     $PP1
1882 05F6 1900 A          JMP     $PL2
1883 05F7 9498*A $PL2:    JSR     PUTS          ; 2ND LETTER/PUNCTUATION
1884 05F8 5C00 A          NOP
1885 05F9 F0D0*A          SKNE    AC0,=0D       ; CHECK IF CR
1886 05FA 8002 A          RTS     2             ;    YES - DONE
1887 05FB F1E5*A          SKNE    AC0,=022      ; CHECK FOR STRINGS
1888 05FC 19CE A          JMP     $STRNG
1889 05FD F160*A          SKNE    AC0,=REMCHAR  ; CHECK IF REMARK CHARACTER
1890 05FE 1978 A          JMP     $REM
1891 05FF F160*A          SKNE    AC0,='$'/256  ; CHECK IF HEX NUMBER
1892 0600 19D3 A          JMP     $HEX
1893 0601 A445 A          OR      AC0,IDENT
1894 0602 D045 A          ST      AC0,IDENT
1895 0603 C95E*A          LD      AC2,=RESTAB-1   ; SCAN RSVD WORD TABLE
1896 0604 1567 A $PL2A:   JSR     NXTRES
1897 0605 190B A          JMP     $PL2B         ;    NOT IN TABLE
1898 0606 F045 A          SKNE    AC0,IDENT
1899 0607 1901 A          JMP     .+2           ;    FOUND
1900 0608 19FB A          JMP     $PL2A
1901 0609 7900 A          AISZ    AC1,0         ; CHECK IF 2-CHAR ID
1902 060A 190A A          JMP     $PL3
1903 060B C07B A          LD      AC0,SCR       ; 2 CHAR ID FOUND - COPY CODE
1904 060C D058 A          ST      AC0,SCOL
1905 060D C200 A          LD      AC0,0(AC2)
1906 060E 9498*A          JSR     PUTS
1907 060F 5C00 A          NOP
1908 0610 19B0 A          JMP     $P2           ; GET NEXT (NOT DATA OR REM)
1909 0611 C045 A $PL2B:   LD      AC0,IDENT     ; RESTART KEYWORD SCAN WITH
1910 0612 2810 A          SHL     AC0,8,0       ;    NEW CHAR
1911 0613 8C7B A          ISZ     SCR
1912 0614 19DB A          JMP     $PL1A
1913 0615 C045 A $PL3:    LD      AC0,IDENT
1914 0616 F14C*A          SKNE    AC0,='GO'     ; IF IDENT IS 'GO',
1915 0617 153A A          JSR     $CKGO         ;    CHECK IF 'GO' OR 'GOTO'/'GO
1916 0618 9548*A          JSR     GETUC         ; CHECK 3RD CHAR - IF NOT LETTER
1917 0619 19AC A          JMP     $PP1          ;    THEN NOT A KEYWORD YET
1918 061A 19AB A          JMP     $PP1
1919 061B 1901 A          JMP     $PL3A
1920 061C 19C6 A          JMP     $PL1
1921 061D 9498*A $PL3A:   JSR     PUTS
1922 061E 5C00 A          NOP
1923 061F 2810 A          SHL     AC0,8,0
1924 0620 D046 A          ST      AC0,IDENT+1
1925 0621 C940*A          LD      AC2,=RESTAB-1   ; SCAN RSVD WORD TABLE
1926 0622 1549 A $PL3B:   JSR     NXTRES
1927 0623 191E A          JMP     $PL3C         ;    NOT IN TABLE
1928 0624 F045 A          SKNE    AC0,IDENT
1929 0625 1901 A          JMP     .+2
1930 0626 19FB A          JMP     $PL3B
1931 0627 F446 A          SKNE    AC1,IDENT+1
1932 0628 1901 A          JMP     .+2
1933 0629 19F8 A          JMP     $PL3B
```

```
1934 062A C07B A          LD      AC0,SCR       ; KEYWORD FOUND - (1ST 3 CHAR)
1935 062B D058 A          ST      AC0,SCOL
1936 062C C200 A          LD      AC0,0(AC2)    ; REPLACE IT WITH CODE
1937 062D 9498*A          JSR     PUTS
1938 062E 5C00 A          NOP
1939 062F D07B A          ST      AC0,SCR
1940 0630 C300 A $PL3R:   LD      AC0,0(AC3)    ; SKIP ANY FOLLOWING MATCHING
1941 0631 9532*A          JSR     CHKLC         ;    CHAR OF KEYWORD
1942 0632 C202 A          LD      AC0,2(AC2)    ; COMPARE TO RIGHT CHAR IN TABLE
1943 0633 A8D3*A          AND     AC0,=07F
1944 0634 F04F A          SKNE    AC0,PFLAG
1945 0635 1901 A          JMP     .+2
1946 0636 1910 A          JMP     $PL4
1947 0637 7A01 A          AISZ    AC2,1
1948 0638 7B01 A          AISZ    AC3,1
1949 0639 C300 A          LD      AC0,0(AC3)    ; COMPARE CHAR TO LEFT CHAR IN T
1950 063A 9529*A          JSR     CHKLC
1951 063B C202 A          LD      AC0,2(AC2)
1952 063C 2C10 A          SHR     AC0,8,0
1953 063D F04F A          SKNE    AC0,PFLAG
1954 063E 1901 A          JMP     .+2
1955 063F 1907 A          JMP     $PL4
1956 0640 7B01 A          AISZ    AC3,1
1957 0641 19EE A          JMP     $PL3R
1958 0642 7BFE A $PL3C:   AISZ    AC3,-2        ; 3 CHAR NOT IN TABLE -
1959 0643 C07B A          LD      AC0,SCR       ;    BACK UP 2 CHAR
1960 0644 7801 A          AISZ    AC0,1
1961 0645 D058 A          ST      AC0,SCOL
1962 0646 991E*A          JMP     $P2
1963 0647 C07B A $PL4:    LD      AC0,SCR       ; CHECK KEYWORD
1964 0648 F0D4*A          SKNE    AC0,=128      ; REM ?
1965 0649 192D A          JMP     $REM
1966 064A F11B*A          SKNE    AC0,=144      ; DATA ?
1967 064B 192B A          JMP     $REM
1978 064C F11A*A          SKNE    AC0,=206      ; LOAD ?
1979 064D 1929 A          JMP     $REM
1980 064E 9D19*A          SKG     AC0,=215      ; SAVE/MERGE/EXEC ?
1981 064F 9D19*A          SKG     AC0,=212
1982 0650 9914*A          JMP     $P2           ;    NONE OF THOSE - RESUME SCAN
1983 0651 1925 A          JMP     $REM          ;    'DATA' TYPE KEYWORD
1984 0652 C300 A $CKGO:   LD      AC0,0(AC3)    ; CHECK CHAR AFTER 'GO'
1985 0653 F116*A          SKNE    AC0,='S'/256  ; 'S' OR 'T' - OK
1986 0654 8000 A          RTS
1987 0655 F115*A          SKNE    AC0,='T'/256
1988 0656 8000 A          RTS
1989 0657 C07B A          LD      AC0,SCR       ; KEYWORD IS 'GO'
1990 0658 D058 A          ST      AC0,SCOL
1991 0659 C168*A          LD      AC0,=157
1992 065A 9498*A          JSR     PUTS
1993 065B 5C00 A          NOP
1994 065C 6400 A          PULL    AC0
1995 065D 9907*A          JMP     $P2
1996 065E 0027 A          .POOL   14
     065F 0677 T
```

```
      0660 0024 A
      0661 0572 T
      0662 168B T
      0663 474F A
      0664 0589 T
      0665 05C1 T
      0666 0090 A
      0667 00CE A
      0668 00D7 A
      0669 00D4 A
      066A 0053 A
      066B 0054 A
1997   ;
1998 066C 7A01 A NXTRES:  AISZ    AC2,1          ; GET FIRST 3 CHAR FROM NEXT RSV
1999 066D C200 A          LD      AC0,(AC2)      ;    WORD TABLE ENTRY INTO AC0,A
2000 066E 4501 A          BOC     NZRO,.+2
2001 066F 8000 A          RTS     0              ; 0 -> NO MORE ENTRIES
2002 0670 B8DA*A          SKAZ    AC0,=0FF00
2003 0671 19FA A          JMP     NXTRES
2004 0672 C601 A          LD      AC1,1(AC2)
2005 0673 C202 A          LD      AC0,2(AC2)
2006 0674 A8DA*A          AND     AC0,=0FF00
2007 0675 6D00 A          RXCH    AC0,AC1
2008 0676 8001 A          RTS     1
2009   ;
2010 0677 C300 A $REM:    LD      AC0,(AC3)      ; REM OR DATA - COPY REMAIN CHAR
2011 0678 9498*A          JSR     PUTS           ;    ON LINE WITHOUT PROCESSING
2012 0679 5C00 A          NOP
2013 067A 7B01 A          AISZ    AC3,1
2014 067B F0D0*A          SKNE    AC0,=0D
2015 067C 8002 A          RTS     2
2016 067D 19F9 A          JMP     $REM
2017   ;
2033   ;
2034   ;
2035   ;        PACK INPUT BUFFER (FOR INPUT COMMAND)
2036   ;
2037 067E C8B1*A PACK:    LD      AC2,=INBUF
2038 067F 5F80 A          RCPY    AC2,AC3
2039 0680 C200 A $PAK:    LD      AC0,(AC2)
2040 0681 2810 A          SHL     AC0,8,0
2041 0682 F820 A          SKNE    AC2,ENDBUF
2042 0683 1903 A          JMP     $PK2
2043 0684 7A01 A          AISZ    AC2,1
2044 0685 C600 A          LD      AC1,(AC2)
2045 0686 5840 A          RXOR    AC1,AC0
2046 0687 D300 A $PK2:    ST      AC0,(AC3)
2047 0688 F820 A          SKNE    AC2,ENDBUF
2048 0689 1903 A          JMP     $PK3
2049 068A 7A01 A          AISZ    AC2,1
2050 068B 7B01 A          AISZ    AC3,1
2051 068C 19F3 A          JMP     $PAK
2052 068D 500D A $PK3:    LI      AC0,0D
2053 068E D301 A          ST      AC0,1(AC3)
```

```
 2054 068F 8000 A          RTS     0
 2055  ;
```

```
2056                    .PAGE   'SCANNER'
2057                    .LOCAL
2058  ;
2059  ;       LEXICAL SCANNER - OBTAINS THE NEXT TOKEN
2060  ;       FROM THE CURRENT LINE BEING SCANNED.
2061  ;
2062  ;       LSTART POINTS TO THE LINE BEING SCANNED AND
2063  ;       COL IS THE COLUMN COUNTER.
2064  ;
2065  ;       RETURNED VARIABLES (BASE PAGE):
2066  ;               CLASS = SEMANTIC TYPE
2067  ;               SYMB = SYMBOL OR POINTER
2068  ;               SYMA = ADDRESS
2069  ;               VALUE = CONSTANT VALUE (2 WORDS)
2070  ;
2071  ;       RETURN: RTS 0  IF ERROR (MSG PRINTED)
2072  ;               RTS 1  NORMAL RETURN
2073  ;
2074  ;       SEMANTIC CLASSES ARE:
2075  ;
2076  ;            CLASS    TYPE                    EXAMPLES
2077  ;            -----    ----                    --------
2078  ;
2079  ;               1   OPERATORS (UNARY)        +  -
2080  ;               2   BINARY OPERATORS         *  /  ** (UPARROW)  MAX  MIN
2081  ;               3   UNARY LOGICAL OP         NOT
2082  ;               4   BINARY LOGICAL OP        AND  OR  XOR
2083  ;               5   RELATIONS                <  <=  =  >=  >  <>  #
2084  ;               6   SYSTEM FUNCTIONS         SIN  SQR  ABS  ...
2085  ;               7   USER FUNCTIONS           FNA ... FNZ
2086  ;               8   PRINT FUNCTION           TAB  SPC  LIN
2087  ;               9   FUNCTION (STRING OP)     LEN
2088  ;               10  LEFT PAREN.              (
2089  ;               11  RIGHT PAREN.             )
2090  ;               12  DELIMITERS               ,  ;
2091  ;               13  END OF STMT              (CR)  ' (APOSTROPHE-COMMENT)
2092  ;               14  STMT SEPARATOR           :
2093  ;               15  IDENTIFIER              A ... Z ... A0 ... Z9 ... AA
2094  ;               16  ARRAY ID                 A( ... Z9(
2095  ;               17  STRING ID                A$ ... Z9$
2096  ;               18  STRING ARRAY ID          A$( ... Z9$(
2097  ;               19  SUBSTRING FUNCTION       LEFT$ MID$ RIGHT$
2098  ;               20  STMT NO. (BCD, HEX)      1 ... 9999
2099  ;               21  NUMERIC CONST            0 ... 9 ... 14.7E-5  (FLT PT)
2100  ;               22  STRING CONSTANT          (QUOTED OR UNQUOTED)
2101  ;               23  STRING FUNCTION
2102  ;               24  SPECIAL IDENTIFIER       RND
2103  ;               25  KEYWORDS                 LET  IF  STEP  ...
2104  ;               27  SYSTEM COMMANDS          LIST  RUN  ...
2105  ;               28  DISK COMMANDS            LOAD  SAVE  ...
2106  ;               30  UNKNOWN PUNCTUATION
2107  ;
2108  ;
2109 0690 5000 A SCAN:   LI      AC0,0          ; INITIALIZE VARIABLES
```

```
2110 0691 D058 A          ST      AC0,SCOL
2111 0692 510F A          LI      AC1,NWDS      ; (NWDS) WORDS STARTING WITH CLA
2112 0693 C92F*A          LD      AC2,=CLASS    ;    PRESET TO ZERO
2113 0694 D200 A          ST      AC0,(AC2)
2114 0695 7A01 A.         AISZ    AC2,1
2115 0696 79FF A          AISZ    AC1,-1
2116 0697 19FC A          JMP     .-3
2117 0698 C12B*A          LD      AC0,='  '
2118 0699 D045 A          ST      AC0,IDENT
2119 069A D046 A          ST      AC0,IDENT+1
2120 069B C17A*A          LD      AC0,=IDENT
2121 069C D057 A          ST      AC0,SSTART
2122 069D C05A A          LD      AC0,NSTRC
2123 069E D059 A          ST      AC0,MAXSTR
2124 069F C03B A          LD      AC0,STYPE     ; CHECK IF SCANNING STRING INPUT
2125 06A0 4102 A          BOC     ZRO,SCAN2
2126 06A1 9900 A          JMP     @.+1
2127 06A2 078A T          .WORD   GETDST
2128 06A3 94A2*A SCAN2:   JSR     GETCH         ; GET NEXT CHAR
2129 06A4 19FE A          JMP     SCAN2         ; IGNORE SPACES BETWEEN THINGS
2130 06A5 1903 A          JMP     $D1           ; DIGIT: LINE# OR CONSTANT
2131 06A6 191E A          JMP     $L1           ; LETTER: IDENTIFIER
2132 06A7 9900 A          JMP     @.+1          ; KEYWORD CODE OR PUNCTUATION
2133 06A8 076E T          .WORD   $SPEC
2134 06A9 C43A A $D1:     LD      AC1,NTYPE     ; CHECK TYPE FLAG FOR KIND OF NU
2135 06AA 7900 A          AISZ    AC1,0
2136 06AB 1902 A          JMP     $SEQNO        ; 1: SEQUENCE NUMBER (BCD)
2137 06AC 9900 A          JMP     @.+1          ; 0: FLOATING POINT CONSTANT
2138 06AD 07E1 T          .WORD   CONST
2139 06AE 5100 A $SEQNO:  LI      AC1,0         ; PROCESS A LINE NUMBER
2140 06AF 1905 A          JMP     $SQ2          ;    (4 DIGIT B.C.D.)
2141 06B0 94A2*A $SQ1:    JSR     GETCH
2142 06B1 190C A          JMP     $SQ4
2143 06B2 1902 A          JMP     $SQ2
2144 06B3 1909 A          JMP     $SQ3
2145 06B4 1908 A          JMP     $SQ3
2146 06B5 A8D1*A $SQ2:    AND     AC0,=0F
2147 06B6 2908 A          SHL     AC1,4,0
2148 06B7 5900 A          RXOR    AC0,AC1
2149 06B8 7A01 A          AISZ    AC2,1
2150 06B9 5C40 A          RCPY    AC1,AC0
2151 06BA B8D9*A          SKAZ    AC0,=0F000
2152 06BB 1902 A          JMP     $SQ4
2153 06BC 19F3 A          JMP     $SQ1
2154 06BD AC3F A $SQ3:    DSZ     COL           ; TERMINATING CHAR NOT YET PROCE
2155 06BE D443 A $SQ4:    ST      AC1,SYMA      ; SYMA = BCD NUMBER
2156 06BF 5014 A          LI      AC0,20        ; CLASS = 20
2157 06C0 D041 A          ST      AC0,CLASS
2158 06C1 8001 A          RTS     1
2159  ;
2160 06C2 009D A          .POOL   3
     06C3 0041 A
     06C4 2020 A
2161  ;
```

```
2162  ;       CHECK SYMBOL, FUNCTION NAME, KEYWORD, OR COMMAND
2163  ;
2164 06C5 C43B A $L1:    LD      AC1,STYPE     ; CHECK IF IN SPECIAL STRING MOD
2165 06C6 7900 A         AISZ    AC1,0
2166 06C7 994F*A         JMP     GETDST        ; PROCESS STRING FOR DATA OR INP
2167 06C8 5104 A         LI      AC1,4
2168 06C9 D459 A         ST      AC1,MAXSTR    ; MAX NAME LENGTH = 4
2169 06CA 9498*A $L2:    JSR     PUTS          ; SAVE 1ST CHARACTER
2170 06CB 5C00 A         NOP
2171 06CC 94A2*A         JSR     GETCH         ; GET & CHECK 2ND CHAR
2172 06CD 1919 A         JMP     $LSP          ;    SPACE
2173 06CE 1902 A         JMP     $L2A          ;    DIGIT OK FOR 2ND CHAR
2174 06CF 1908 A         JMP     $L2B          ;    LETTER
2175 06D0 190D A         JMP     $L3           ;    PUNCTUATION
2176 06D1 9498*A $L2A:   JSR     PUTS          ; DIGIT: LAST CHAR OF NAME
2177 06D2 5C00 A         NOP
2178 06D3 94A2*A         JSR     GETCH         ; CHECK FOLLOWING
2179 06D4 1912 A         JMP     $LSP
2180 06D5 190C A         JMP     $LC1
2181 06D6 190B A         JMP     $LC1
2182 06D7 1906 A         JMP     $L3
2183 06D8 9498*A $L2B:   JSR     PUTS          ; 2ND OR LATER LETTERS - SAVE
2184 06D9 5C00 A         NOP
2185 06DA 94A2*A         JSR     GETCH         ; GET & CHECK NEXT
2186 06DB 190B A         JMP     $LSP
2187 06DC 1905 A         JMP     $LC1
2188 06DD 19FA A         JMP     $L2B
2189 06DE F181*A $L3:    SKNE    AC0,='$'/256  ;    PUNCTUATION - CHECK IF $
2190 06DF 1904 A         JMP     $LDOL
2191 06E0 F137*A $LP1:   SKNE    AC0,='('/256  ; CHECK FOR LEFT PAREN
2192 06E1 190A A         JMP     $LP2
2193 06E2 ACBF A $LC1:   DSZ     COL           ; DECREMENT COUNT TO BEFORE
2194 06E3 190B A         JMP     $LC2          ;    TERMINATING CHARACTER
2195 06E4 C045 A $LDOL:  LD      AC0,IDENT     ; $ AT END: SET BIT 15 OF 1ST WO
2196 06E5 A4D7*A         OR      AC0,=08000    ;    OF NAME AS A STRING FLAG
2197 06E6 D045 A         ST      AC0,IDENT
2198 06E7 94A2*A $LSP:   JSR     GETCH         ; $ OR SPACE: CHECK IF ( FOLLOWS
2199 06E8 19FE A         JMP     $LSP          ;    SKIP SPACES
2200 06E9 19F8 A         JMP     $LC1          ;    DIGIT
2201 06EA 19F7 A         JMP     $LC1          ;    LETTER
2202 06EB 19F4 A         JMP     $LP1          ;    PUNCTUATION - CHECK IF (
2203 06EC C0D4*A $LP2:   LD      AC0,=080      ; ( AFTER NAME - SET FLAG BIT
2204 06ED A445 A         OR      AC0,IDENT
2205 06EE D045 A         ST      AC0,IDENT
2206 06EF C045 A $LC2:   LD      AC0,IDENT     ; CHECK IDENTIFIER TYPE
2207 06F0 F128*A         SKNE    AC0,=080+'FN'  ; FNX( ETC.: USER FUNCTION
2208 06F1 191C A         JMP     $FN
2209 06F2 B8D4*A         SKAZ    AC0,=080
2210 06F3 1908 A         JMP     $AY
2211 06F4 4202 A         BOC     POS,$NORM
2212 06F5 5011 A         LI      AC0,17        ; SIMPLE STRING
2213 06F6 1909 A         JMP     $CKARY
2214 06F7 500F A $NORM:  LI      AC0,15        ; SIMPLE VARIABLE
2215 06F8 D041 A $SAVE:  ST      AC0,CLASS     ; SAVE CLASS TYPE
```

```
2216 06F9 C045 A          LD      AC0,IDENT     ; 1ST WORD OF IDENT IS IDENTIFIE
2217 06FA D042 A          ST      AC0,SYMB
2218 06FB 991E*A          JMP     LOOKUP        ; CHECK SYMBOL TABLE
2219 06FC 4202 A $AY:     BOC     POS,$AY2
2220 06FD 5012 A.         LI      AC0,18        ; STRING ARRAY
2221 06FE 1901 A          JMP     $CKARY
2222 06FF 5010 A $AY2:    LI      AC0,16        ; NUMERIC ARRAY
2223 0700 D041 A $CKARY:  ST      AC0,CLASS
2224 0701 C045 A          LD      AC0,IDENT
2225 0702 D042 A          ST      AC0,SYMB
2226 0703 9516*A          JSR     LOOKUP        ; CHECK TO BE SURE IT IS ALLOCAT
2227 0704 8000 A          RTS     0
2228 0705 C303 A          LD      AC0,3(AC3)
2229 0706 453E A          BOC     NZRO,$R1
2230 0707 500A A          LI      AC0,10        ; IF MODES 2 OR 8:
2231 0708 B813 A          SKAZ    AC0,MODE      ;    ERROR IF ADDR = 0
2232 0709 1901 A          JMP     .+2
2233 070A 193A A          JMP     $R1
2234 070B 94A1*A          JSR     MESG
2235 070C 1893 T          .WORD   ALOCM
2236 070D 8000 A          RTS     0
2237 070E C046 A $FN:     LD      AC0,IDENT+1   ; FN: 3RD CHAR IS FUNCTION LETTE
2238 070F F1B4*A          SKNE    AC0,='  '
2239 0710 19EE A          JMP     $AY2          ; IF FN( , IT IS NORMAL ARRAY
2240 0711 2C10 A          SHR     AC0,8,0
2241 0712 A508*A          OR      AC0,=02E00
2242 0713 D045 A          ST      AC0,IDENT     ; SET 1ST CHAR TO '.' FOR SYMBOL
2243 0714 5007 A          LI      AC0,7         ; USER FUNCTION
2244 0715 19E2 A          JMP     $SAVE
2245  ;
2246 0716 0045 A          .POOL   6
     0717 078A T
     0718 0028 A
     0719 46CE A
     071A 089D T
     071B 2E00 A
2247  ;
2248  ;        PROCESS KEYWORD CODE
2249  ;
2250 071C 5019 A KEY:     LI      AC0,25        ; DEFAULT CLASS IS 25
2251 071D D041 A          ST      AC0,CLASS
2252 071E CD44*A          LD      AC3,=VALTAB   ; CHECK IF IT IS IN TABLE
2253 071F C300 A $K1:     LD      AC0,(AC3)     ;    (DIFFERENT CLASSES)
2254 0720 410A A          BOC     ZRO,$K3
2255 0721 F042 A          SKNE    AC0,SYMB
2256 0722 1902 A          JMP     $K2
2257 0723 7B02 A          AISZ    AC3,2
2258 0724 19FA A          JMP     $K1
2259 0725 C301 A $K2:     LD      AC0,1(AC3)    ; SAVE PREC & CLASS
2260 0726 5D00 A          RCPY    AC0,AC1
2261 0727 2D10 A          SHR     AC1,8,0
2262 0728 D444 A          ST      AC1,PREC
2263 0729 A8D5*A          AND     AC0,=0FF
2264 072A D041 A          ST      AC0,CLASS
```

```
2265 072B C196*A $K3:    LD      ACO,=157     ; IF 'GO', CHECK GOTO OR GOSUB
2266 072C F042 A         SKNE    ACO,SYMB
2267 072D 1926 A         JMP     $KGO
2268 072E 94A2*A $K3A:   JSR     GETCH        ; CHECK IF ( IS NEXT
2269 072F 19FE A         JMP     $K3A
2270 0730 1903 A         JMP     $K4
2271 0731 1902 A         JMP     $K4
2272 0732 F1E5*A         SKNE    ACO,='('/256
2273 0733 1902 A         JMP     $K5
2274 0734 AC3F A $K4:    DSZ     COL          ; NOT PAREN - BACK UP
2275 0735 1902 A         JMP     $K6
2276 0736 C0D4*A $K5:    LD      ACO,=080     ; SET ( FLAG
2277 0737 D04E A         ST      ACO,ESIGN
2278 0738 C041 A $K6:    LD      ACO,CLASS    ; NOW CHECK CLASS OF KEYWORD
2279 0739 9CCE*A         SKG     ACO,=9       ; IF CLASS IS A FUNCTION, CHECK
2280 073A 9D29*A         SKG     ACO,=5
2281 073B 1901 A         JMP     .+2
2282 073C 1909 A         JMP     $CKP
2283 073D F127*A         SKNE    ACO,=23
2284 073E 1907 A         JMP     $CKP
2285 073F F126*A         SKNE    ACO,=19
2286 0740 1905 A         JMP     $CKP
2287 0741 C04E A         LD      ACO,ESIGN    ; IF PAREN FOLLOWS, DECREMENT CO
2288 0742 4102 A         BOC     ZRO,$R1      ;    SO IT WILL BE SCANNED NEXT
2289 0743 AC3F A         DSZ     COL
2290 0744 5C00 A         NOP
2291 0745 8001 A $R1:    RTS     1
2292 0746 C04E A $CKP:   LD      ACO,ESIGN    ; FUNCTION MUST BE FOLLOWED BY '
2293 0747 45FD A         BOC     NZRO,$R1
2294 0748 C042 A         LD      ACO,SYMB     ; CHECK 'RND' - SPECIAL VARIABLE
2295 0749 F11D*A         SKNE    ACO,=176
2296 074A 1904 A         JMP     $RND
2297 074B F11C*A         SKNE    ACO,=140     ; CHECK 'INPUT' - KEYWORD OR FUN
2298 074C 1905 A         JMP     $IN
2299 074D 5008 A         LI      ACO,8        ; SYNTAX ERROR (NO '(') THEN RTS
2300 074E 98C3*A         JMP     ERROR
2301 074F 5018 A $RND:   LI      ACO,24       ; RND - SPECIAL IDENTIFIER (WITH
2302 0750 D041 A         ST      ACO,CLASS
2303 0751 8001 A         RTS     1
2304 0752 5019 A $IN:    LI      ACO,25       ; INPUT COMMAND
2305 0753 19FC A         JMP     $RND+1
2306 0754 94A2*A $KGO:   JSR     GETCH        ; 'GO' - CHECK FOR 'TO' OR 'SUB'
2307 0755 19FE A         JMP     $KGO
2308 0756 5C00 A         NOP
2309 0757 5C00 A         NOP
2310 0758 F110*A         SKNE    ACO,=137     ; TO ?
2311 0759 1904 A         JMP     $KGTO
2312 075A F10F*A         SKNE    ACO,=158     ; SUB ?
2313 075B 1904 A         JMP     $KGSUB
2314 075C 5008 A         LI      ACO,8        ; ANYTHING ELSE IS SYNTAX ERROR
2315 075D 98C3*A         JMP     ERROR
2316 075E C10C*A $KGTO:  LD      ACO,=131     ; GO TO -> GOTO
2317 075F 1901 A         JMP     .+2
2318 0760 C10B*A $KGSUB: LD      ACO,=132     ; GO SUB -> GOSUB
```

```
2319 0761 D042 A          ST      AC0,SYMB      ; SAVE SYMB (CLASS ALREADY SET)
2320 0762 8001 A          RTS     1
2321 0763 17A3 T          .POOL   11
     0764 0005 A
     0765 0017 A
     0766 0013 A
     0767 00B0 A
     0768 008C A
     0769 0089 A
     076A 009E A
     076B 0083 A
     076C 0084 A
     076D 0000 A
2322  ;
2323  ;         CHECK SPECIAL CHARACTERS (PUNCTUATION)
2324  ;
2325 076E D042 A $SPEC:   ST      AC0,SYMB      ; SAVE CHARACTER/CODE
2326 076F B8D4*A          SKAZ    AC0,=080      ; CHECK IF KEYWORD CODE
2327 0770 19AB A          JMP     KEY           ;    YES
2328 0771 F162 A          SKNE    AC0,$PER      ; PERIOD: FLOATING POINT CONSTAN
2329 0772 196E A          JMP     CONST
2330 0773 F1F9*A          SKNE    AC0,='$'/256  ; DOLLAR: HEXADECIMAL CONSTANT
2331 0774 9963*A          JMP     GETHX
2332 0775 F163*A          SKNE    AC0,=022      ; QUOTATION MARK: START A STRING
2333 0776 191F A          JMP     GETSTR
2334 0777 CD62*A          LD      AC3,=PCTAB    ; LOOK UP IN TABLE
2335 0778 F300 A $CK1A:   SKNE    AC0,0(AC3)
2336 0779 1907 A          JMP     $FOUND
2337 077A FD60*A          SKNE    AC3,=PCLAST
2338 077B 1902 A          JMP     $NOTF         ; NOT IN TABLE
2339 077C 7B02 A          AISZ    AC3,2
2340 077D 19FA A          JMP     $CK1A         ; GO TO NEXT
2341 077E 511E A $NOTF:   LI      AC1,30        ; CHAR NOT FOUND
2342 077F D441 A          ST      AC1,CLASS     ; CLASS=30
2343 0780 8001 A          RTS     1
2344 0781 C301 A $FOUND:  LD      AC0,1(AC3)    ; LEFT BYTE - PRECEDENCE
2345 0782 2C10 A          SHR     AC0,8,0
2346 0783 F0D5*A          SKNE    AC0,=0FF      ;    -1 IF OFF
2347 0784 50FF A          LI      AC0,-1
2348 0785 D044 A          ST      AC0,PREC
2349 0786 C301 A          LD      AC0,1(AC3)    ; RIGHT BYTE - CLASS
2350 0787 A8D5*A          AND     AC0,=0FF
2351 0788 D041 A          ST      AC0,CLASS
2352 0789 8001 A          RTS     1
2353  ;
2354 078A 94A2*A GETDST:  JSR     GETCH         ; GET DATA/INPUT STRING
2355 078B 19FE A          JMP     GETDST        ; IGNORE LEADING BLANKS
2356 078C 5C00 A          NOP
2357 078D 5C00 A          NOP
2358 078E F14A*A          SKNE    AC0,=022      ; IF STARTS WITH QUOTE -
2359 078F 1906 A          JMP     GETSTR        ;    THEN SCAN AS QUOTED STRING
2360 0790 1538 A          JSR     CHKTERM       ; IF DELIMITER CHAR,
2361 0791 19DC A          JMP     $SPEC         ;    RETURN THE CHARACTER
2362 0792 512C A          LI      AC1,','/256   ; TERMINATOR IS COMMA (OR SOME O
```

```
2363 0793 AC3F A          DSZ     COL          ; BACKUP TO REREAD 1ST CHAR OF S
2364 0794 5C00 A          NOP
2365 0795 1901 A          JMP     .+2
2366 0796 5122 A GETSTR:  LI      AC1,022      ; QUOTE TERMINATOR (NORMAL)
2367 0797 D44E A          ST      AC1,ESIGN
2368 0798 500A A          LI      AC0,10       ; ALLOCATE TEMP STRING IF IN
2369 0799 B813 A          SKAZ    AC0,MODE     ;    COMMAND MODE OR PASS TWO
2370 079A 94A5*A          JSR     STRTMP       ; ALLOCATE & INITIALIZE TEMP STR
2371 079B 94A2*A $STR1:   JSR     GETCH        ; GET NEXT STRING CHAR
2372 079C 5C00 A          NOP
2373 079D 5C00 A          NOP
2374 079E 5C00 A          NOP
2375 079F F04E A          SKNE    AC0,ESIGN    ; CHECK IF TERMINATOR
2376 07A0 190F A          JMP     $STR3
2377 07A1 F0D0*A          SKNE    AC0,=0D      ; CR FOUND BEFORE QUOTE -
2378 07A2 1920 A          JMP     $SERR        ;    ERROR IF QUOTED STRING
2379 07A3 5122 A          LI      AC1,022      ; IF STRING IS UNQUOTED,
2380 07A4 F44E A          SKNE    AC1,ESIGN    ;    CHECK IF TERMINATING PUNCTU
2381 07A5 1902 A          JMP     $STR2
2382 07A6 1522 A          JSR     CHKTERM
2383 07A7 1908 A          JMP     $STR3        ; TERMINATING PUNCTUATION
2384 07A8 5D00 A $STR2:   RCPY    AC0,AC1      ; ADD CHAR TO STRING (IF NOT EDI
2385 07A9 500A A          LI      AC0,10
2386 07AA A813 A          AND     AC0,MODE
2387 07AB 41EF A          BOC     ZR0,$STR1
2388 07AC 5C40 A          RCPY    AC1,AC0
2389 07AD 9498*A          JSR     PUTS
2390 07AE 5C00 A          NOP
2391 07AF 19EB A          JMP     $STR1
2392 07B0 F128*A $STR3:   SKNE    AC0,=022     ; CHECK IF TERMINATOR WAS QUOTE
2393 07B1 1901 A          JMP     .+2
2394 07B2 1906 A          JMP     $STR4        ;    NO - STRING IS DONE
2395 07B3 94A2*A          JSR     GETCH        ; CHECK CHAR AFTER QUOTE
2396 07B4 1904 A          JMP     $STR4
2397 07B5 1903 A          JMP     $STR4
2398 07B6 1902 A          JMP     $STR4
2399 07B7 F121*A          SKNE    AC0,=022
2400 07B8 19EF A          JMP     $STR2        ; ANOTHER QUOTE: STILL PART OF S
2401 07B9 AC3F A $STR4:   DSZ     COL          ; END OF STRING: BACK UP INPUT C
2402 07BA 5C00 A          NOP
2403 07BB 5016 A          LI      AC0,22       ; CLASS=22
2404 07BC D041 A          ST      AC0,CLASS
2405 07BD C857 A          LD      AC2,SSTART
2406 07BE C058 A          LD      AC0,SCOL
2407 07BF 7AFF A          AISZ    AC2,-1
2408 07C0 D200 A          ST      AC0,(AC2)    ; SAVE LENGTH IN 0'TH WORD
2409 07C1 D843 A          ST      AC2,SYMA     ; SYMA = ADDRESS OF STRING
2410 07C2 8001 A          RTS     1
2411 07C3 5022 A $SERR:   LI      AC0,022      ; CR ENDED STRING - ERROR IF
2412 07C4 F04E A          SKNE    AC0,ESIGN    ;    STRING WAS QUOTED
2413 07C5 1901 A          JMP     .+2
2414 07C6 19F2 A          JMP     $STR4
2415 07C7 500C A          LI      AC0,12       ; STRING HAS NO CLOSING QUOTE
2416 07C8 98C3*A          JMP     ERROR
```

```
2417  ;
2418 07C9 F112*A CHKTERM: SKNE    AC0,='','/256 ; CHECK FOR UNQUOTED STRING TERM
2419 07CA 8000 A          RTS     0             ; ',', '&', '!', APOSTROPHE, OR
2420 07CB F111*A          SKNE    AC0,='&'/256
2421 07CC 8000 A.         RTS     0
2422 07CD F110*A          SKNE    AC0,=''''/256
2423 07CE 8000 A          RTS     0
2424 07CF F10F*A          SKNE    AC0,='!'/256
2425 07D0 8000 A          RTS     0
2426 07D1 F0D0*A          SKNE    AC0,=0D
2427 07D2 8000 A          RTS     0
2428 07D3 8001 A          RTS     1             ; NOT ONE OF THOSE
2429  ;
2430 07D4 002E A $PER:    .WORD   '.'/256       ; PERIOD - PART OF NUMBER
2431 07D5 0045 A $E:      .WORD   'E'/256       ; EXPONENT INDICATOR
2432 07D6 002B A $PLUS:   .WORD   '+'/256
2433 07D7 002D A $MINUS:  .WORD   '-'/256
2434 07D8 087B T          .POOL   9
     07D9 0022 A
     07DA 166E T
     07DB 168A T
     07DC 002C A
     07DD 0026 A
     07DE 0027 A
     07DF 0021 A
     07E0 0000 A
2435  ;
2436  ;        ASCII TO FLOATING POINT CONVERSION
2437  ;
2438  ;
2439 07E1 5101 A CONST:   LI      AC1,1         ; INITIALIZE VARIABLES NOT
2440 07E2 D44E A          ST      AC1,ESIGN     ;    INITIALIZED AT START OF SCA
2441 07E3                 DLD     AC1,FL1
2442 07E5                 DST     AC1,PLC
2443 07E7 F1EC A          SKNE    AC0,$PER
2444 07E8 1933 A          JMP     $NOTD
2445 07E9 1905 A          JMP     $D0           ; INITIALLY USE INTEGER ARITHMET
2446 07EA 94A2*A $NC0:    JSR     GETCH
2447 07EB 1915 A          JMP     $FLT2
2448 07EC 1902 A          JMP     $D0
2449 07ED 1913 A          JMP     $FLT2
2450 07EE 1912 A          JMP     $FLT2
2451 07EF A8D1*A $D0:     AND     AC0,=0F       ; ADD IN DIGIT TO VALUE
2452 07F0 D04A A          ST      AC0,DIGIT
2453 07F1 C047 A          LD      AC0,VALUE
2454 07F2 9DED*A          SKG     AC0,=3275
2455 07F3 1901 A          JMP     .+2
2456 07F4 1907 A          JMP     $FLT1         ; VALUE > 3275 -- CONTINUE IN FL
2457 07F5 2802 A          SHL     AC0,1,0       ; MULTIPLY VALUE BY 10
2458 07F6 5D00 A          RCPY    AC0,AC1
2459 07F7 2804 A          SHL     AC0,2,0
2460 07F8 6840 A          RADD    AC1,AC0
2461 07F9 E04A A          ADD     AC0,DIGIT
2462 07FA D047 A          ST      AC0,VALUE
```

```
2463 07FB 19EE A          JMP     $NC0
2464 07FC 94B6*A $FLT1:   JSR     INTFL       ; CONVERT VALUE TO FLOATING POIN
2465 07FD                 DST     AC0,VALUE   ;    THEN CONTINUE WITH DIGIT
2466 07FF C04A A          LD      AC0,DIGIT
2467 0800 190C A          JMP     $D1A
2468 0801 D04A A $FLT2:   ST      AC0,DIGIT   ; NON-DIGIT: CONVERT INTEGER TO
2469 0802 C047 A          LD      AC0,VALUE   ;    THEN CHECK CHARACTER
2470 0803 94B6*A          JSR     INTFL
2471 0804                 DST     AC0,VALUE
2472 0806 C04A A          LD      AC0,DIGIT
2473 0807 1914 A          JMP     $NOTD
2474 0808 94A2*A $NC:     JSR     GETCH       ; GET NEXT CHAR
2475 0809 1954 A          JMP     $DONE
2476 080A 1902 A          JMP     $D1A
2477 080B 1910 A          JMP     $NOTD       ; NOT A DIGIT
2478 080C 190F A          JMP     $NOTD
2479 080D A8D1*A $D1A:    AND     AC0,=0F     ; CONVERT ASCII TO BINARY
2480 080E 94B6*A          JSR     INTFL
2481 080F                 DST     AC0,DIGIT
2482 0811                 DLD     AC0,VALUE   ; VALUE = VALUE*10 + DIGIT
2483 0813                 DLD     AC2,F10
2484 0815 94BD*A          JSR     FLMULT
2485 0816                 DLD     AC2,DIGIT
2486 0818 94BC*A          JSR     FLADD
2487 0819                 DST     AC0,VALUE
2488 081B 19EC A          JMP     $NC
2489 081C F1B7 A $NOTD:   SKNE    AC0,$PER    ; IS THE CHAR A PERIOD?
2490 081D 1901 A          JMP     $FR1
2491 081E 191B A          JMP     $ND2
2492 081F 94A2*A $FR1:    JSR     GETCH       ; GET FRACTIONAL PART
2493 0820 193D A          JMP     $DONE
2494 0821 1902 A          JMP     .+3
2495 0822 1917 A          JMP     $ND2        ; NOT A DIGIT
2496 0823 1916 A          JMP     $ND2
2497 0824 A8D1*A          AND     AC0,=0F     ; CONVERT ASCII TO BINARY
2498 0825 94B6*A          JSR     INTFL
2499 0826                 DST     AC0,DIGIT
2500 0828                 DLD     AC0,PLC     ; PLC = PLC/10
2501 082A                 DLD     AC2,F10
2502 082C 94BE*A          JSR     FLDIV
2503 082D                 DST     AC0,PLC
2504 082F                 DLD     AC2,DIGIT   ; VALUE = VALUE +
2505 0831 94BD*A          JSR     FLMULT      ;          (DIGIT*PLC)
2506 0832                 DLD     AC2,VALUE
2507 0834 94BC*A          JSR     FLADD
2508 0835                 DST     AC0,VALUE
2509 0837 19E7 A          JMP     $FR1
2510 0838 5000 A F10:     .WORD   05000,0004  ; 10 (FLOATING POINT)
     0839 0004 A
2511 083A F19A A $ND2:    SKNE    AC0,$E      ; IS CHARACTER AN 'E'?
2512 083B 1901 A          JMP     .+2
2513 083C 1921 A          JMP     $DONE
2514 083D 94A2*A          JSR     GETCH       ; GET NEXT
2515 083E 193A A          JMP     $ERR2
```

```
2516 083F 190D A          JMP     $NC3
2517 0840 1938 A          JMP     $ERR2
2518 0841 F194 A          SKNE    AC0,$PLUS
2519 0842 1905 A          JMP     $NC2
2520 0843 F193 A.         SKNE    AC0,$MINUS
2521 0844 1901 A          JMP     .+2
2522 0845 1933 A          JMP     $ERR2
2523 0846 5100 A          LI      AC1,0
2524 0847 D44E A          ST      AC1,ESIGN       ; SAVE EXP SIGN: POS=1, NEG=0
2525 0848 94A2*A $NC2:    JSR     GETCH           ; GET 1ST EXP DIGIT
2526 0849 192F A          JMP     $ERR2
2527 084A 1902 A          JMP     $NC3
2528 084B 192D A          JMP     $ERR2
2529 084C 192C A          JMP     $ERR2
2530 084D A8D1*A $NC3:    AND     AC0,=0F
2531 084E D049 A          ST      AC0,EXPN        ; SAVE 1ST EXPN DIGIT
2532 084F 94A2*A          JSR     GETCH
2533 0850 190D A          JMP     $DONE
2534 0851 1902 A          JMP     .+3
2535 0852 190B A          JMP     $DONE
2536 0853 190A A          JMP     $DONE
2537 0854 A8D1*A          AND     AC0,=0F
2538 0855 D04A A          ST      AC0,DIGIT
2539 0856 C049 A          LD      AC0,EXPN        ; MULTIPLY EXPONENT BY 10
2540 0857 2802 A          SHL     AC0,1,0         ;    THEN ADD DIGIT (MAX=99)
2541 0858 5D00 A          RCPY    AC0,AC1
2542 0859 2904 A          SHL     AC1,2,0
2543 085A 6840 A          RADD    AC1,AC0
2544 085B E04A A          ADD     AC0,DIGIT
2545 085C D049 A          ST      AC0,EXPN
2546 085D 8C3F A $D2:     ISZ     COL             ; LAST CHAR ALREADY PROCESSED
2547 085E C049 A $DONE:   LD      AC0,EXPN        ; CHECK IF EXPONENT TO BE PROCES
2548 085F 4115 A          BOC     ZR0,$D5
2549 0860 AC4E A          DSZ     ESIGN           ; NEGATIVE EXPONENT IF ESIGN=0
2550 0861 1909 A          JMP     $D3N
2551 0862                 DLD     AC0,VALUE       ; EXPONENT > 0 --
2552 0864         $D3P:   DLD     AC2,F10         ;    DO REPEATED MULTIPLY BY 10
2553 0866 94BD*A          JSR     FLMULT
2554 0867 4C0B A          BOC     OVF,$D4
2555 0868 AC49 A          DSZ     EXPN
2556 0869 19FA A          JMP     $D3P
2557 086A 1908 A          JMP     $D4
2558 086B         $D3N:   DLD     AC0,VALUE       ; NEGATIVE EXPONENT
2559 086D         $D3N1:  DLD     AC2,F10         ;    DO REPEATED DIVIDE BY 10
2560 086F 94BE*A          JSR     FLDIV
2561 0870 4C02 A          BOC     OVF,$D4
2562 0871 AC49 A          DSZ     EXPN
2563 0872 19FA A          JMP     $D3N1
2564 0873         $D4:    DST     AC0,VALUE       ; STORE FINAL VALUE
2565 0875 5015 A $D5:     LI      AC0,21          ; CLASS=21
2566 0876 D041 A          ST      AC0,CLASS
2567 0877 AC3F A          DSZ     COL
2568 0878 8001 A          RTS     1
2569 0879 500A A $ERR2:   LI .    AC0,10          ; EXPRESSION ERROR IN CONSTANT
```

```
2570 087A 98C3*A          JMP     ERROR          ; PRINT ERROR MSG THEN RTS
2571  ;
2572  ;
2573  ;        HEXADECIMAL INPUT - $NNNN
2574  ;
2575  ;        INTEGER IS CONVERTED TO FLOATING POINT AND RETURNED
2576  ;        AS A CONSTANT.  THEREFORE, HEX INPUT MAY BE USED WHEREEVER
2577  ;        A NUMBER IS NEEDED (ALWAYS TREATED AS AN INTEGER).
2578  ;
2579                    .LOCAL
2580 087B 5000 A GETHX: LI      AC0,0          ; CLEAR VALUE TO ZERO
2581 087C 5100 A        LI      AC1,0
2582 087D 94A2*A        JSR     GETCH          ; FIRST FOLLOWING CHAR
2583 087E 191B A        JMP     $OVF           ;    MUST BE HEX DIGIT
2584 087F 190B A        JMP     $X2
2585 0880 1906 A        JMP     $X3
2586 0881 1918 A        JMP     $OVF
2587 0882 94A2*A $X1:   JSR     GETCH          ; GET NEXT CHAR
2588 0883 190E A        JMP     $X4            ;    SPACE
2589 0884 1906 A        JMP     $X2            ;    DIGIT
2590 0885 1901 A        JMP     $X3            ;    LETTER - CHECK IF A TO F
2591 0886 190B A        JMP     $X4            ;    PUNCTUATION - STOP SCAN
2592 0887 9D14*A $X3:   SKG     AC0,='F'/256
2593 0888 1901 A        JMP     .+2
2594 0889 1910 A        JMP     $OVF
2595 088A 7809 A        AISZ    AC0,9          ; CONVERT 'A'-'F'
2596 088B A8D1*A $X2:   AND     AC0,=0F        ; CONVERT TO 4 BITS
2597 088C 6D00 A        RXCH    AC0,AC1
2598 088D B8D9*A        SKAZ    AC0,=0F000     ; SHIFT VALUE LEFT 4 BITS, BUT F
2599 088E 190B A        JMP     $OVF           ;    CHECK IF ALREADY HAVE 16 BI
2600 088F 2808 A        SHL     AC0,4,0
2601 0890 6900 A        RADD    AC0,AC1        ; NOW ADD IN HEX DIGIT
2602 0891 19F0 A        JMP     $X1            ; GO ON TO NEXT DIGIT
2603 0892 AC3F A $X4:   DSZ     COL            ; RETURN POINTER TO TERMINATING
2604 0893 5015 A        LI      AC0,21         ; CLASS = 21 (FLOATING PT.)
2605 0894 D041 A        ST      AC0,CLASS
2606 0895 5C40 A        RCPY    AC1,AC0
2607 0896 94B6*A        JSR     INTFL
2608 0897               DST     AC0,VALUE
2609 0899 8001 A        RTS     1
2610 089A 500A A $OVF:  LI      AC0,10         ; VALUE ERROR
2611 089B 98C3*A        JMP     ERROR
2612  ;
2613 089C 0046 A        .POOL   1
```

```
2614                        .PAGE   'SYMBOL TABLE ROUTINE'
2615                        .LOCAL
2616  ;
2617  ;         LOOKUP  -   LOOK UP SYMBOL IN SYMBOL TABLE.
2618  ;             .       ADD TO TABLE IF NOT FOUND.
2619  ;
2620  ;         USUALLY CALLED AS PART OF 'SCAN' - RETURN IS RTS 1
2621  ;
2622 089D CC38 A LOOKUP: LD      AC3,SY1       ; LOAD HIGH ADDR.
2623 089E 5000 A         LI      AC0,0         ; PRESET SYMA TO 0 FOR LATER TES
2624 089F D043 A         ST      AC0,SYMA
2625 08A0 C052 A         LD      AC0,FNLINE    ; IF ID IS A SIMPLE IDENTIFIER,
2626 08A1 4106 A         BOC     ZRO,$L1       ;    MODE = 8 OR 2, AND FNLINE <
2627 08A2 C041 A         LD      AC0,CLASS     ;    THEN GO TO SPECIAL PROCESSI
2628 08A3 78F1 A         AISZ    AC0,-15       ;    TO CHECK FOR DUMMY PARAMETE
2629 08A4 1903 A         JMP     $L1
2630 08A5 500A A         LI      AC0,10
2631 08A6 A813 A         AND     AC0,MODE
2632 08A7 4524 A         BOC     NZRO,$LSPEC
2633 08A8 FC51 A $L1:    SKNE    AC3,SY2       ; COMPARE TO LOW-1
2634 08A9 190F A         JMP     $ADD
2635 08AA 7BFC A         AISZ    AC3,-4
2636 08AB C301 A         LD      AC0,1(AC3)    ; COMPARE CURRENT ENTRY
2637 08AC F042 A         SKNE    AC0,SYMB
2638 08AD 1901 A         JMP     .+2           ;    MATCH
2639 08AE 19F9 A         JMP     $L1           ;    NO MATCH - TRY NEXT
2640 08AF C041 A         LD      AC0,CLASS     ; IF SIMPLE ID, CHECK PARAM LINE
2641 08B0 78F1 A         AISZ    AC0,-15
2642 08B1 1904 A         JMP     $FND
2643 08B2 C302 A         LD      AC0,2(AC3)    ; COMPARE PARAMETER LINE
2644 08B3 F052 A         SKNE    AC0,FNLINE
2645 08B4 1901 A         JMP     $FND
2646 08B5 19F2 A         JMP     $L1
2647 08B6 7B01 A $FND:   AISZ    AC3,1         ; SYMBOL ADDRESS
2648 08B7 DC43 A         ST      AC3,SYMA
2649 08B8 8001 A $RTN:   RTS     1
2650  ;
2651 08B9 C451 A $ADD:   LD      AC1,SY2       ; MAKE SURE THERE IS ROOM TO
2652 08BA 79FC A         AISZ    AC1,-4        ;    EXPAND TABLE DOWNWARD
2653 08BB C055 A         LD      AC0,ANEXT
2654 08BC 94A4*A         JSR     SKL           ; SKIP IF (AC0) < (AC1)
2655 08BD 1920 A         JMP     $ERR
2656 08BE 5F40 A         RCPY    AC1,AC3       ; ADD NEW ITEM TO TABLE
2657 08BF DC51 A         ST      AC3,SY2       ; UPDATE SY2 POINTER
2658 08C0 C042 A         LD      AC0,SYMB
2659 08C1 D301 A         ST      AC0,1(AC3)    ; PUT SYMBOL IN TABLE
2660 08C2 5000 A         LI      AC0,0         ; INITIALIZE OTHER WORDS TO 0
2661 08C3 D302 A         ST      AC0,2(AC3)
2662 08C4 D303 A         ST      AC0,3(AC3)
2663 08C5 D304 A         ST      AC0,4(AC3)
2664 08C6 C041 A         LD      AC0,CLASS
2665 08C7 78F1 A         AISZ    AC0,-15
2666 08C8 19ED A         JMP     $FND
2667 08C9 C052 A         LD      AC0,FNLINE    ; SIMPLE ID - SET 2ND WORD
```

```
2668 08CA D302 A          ST      AC0,2(AC3)    ;     TO PARAMETER LINE VALUE
2669 08CB 19EA A          JMP     $FND          ;     (NORMALLY ZERO)
2670  ;
2671  ;        SPECIAL TESTING FOR DUMMY PARAMETERS
2672  ;
2673 08CC FC51 A $LSPEC:  SKNE    AC3,SY2       ; COMPARE TO LOW-1
2674 08CD 190D A          JMP     $NOT          ; NO MATCHING DUMMY PARAM
2675 08CE 7BFC A          AISZ    AC3,-4
2676 08CF C301 A          LD      AC0,1(AC3)    ; COMPARE CURRENT ENTRY
2677 08D0 F042 A          SKNE    AC0,SYMB
2678 08D1 1901 A          JMP     .+2           ;     MATCH
2679 08D2 19F9 A          JMP     $LSPEC        ;     NO MATCH - TRY NEXT
2680 08D3 C302 A          LD      AC0,2(AC3)    ; COMPARE PARAMETER LINE
2681 08D4 F052 A          SKNE    AC0,FNLINE
2682 08D5 19E0 A          JMP     $FND          ;     MATCH - DUMMY PARAMETER FOU
2683 08D6 45F5 A          BOC     NZRO,$LSPEC   ; WRONG DUMMY PARAMETER
2684 08D7 5CC0 A          RCPY    AC3,AC0       ; NOT A DUMMY PARAMETER -
2685 08D8 7801 A          AISZ    AC0,1         ;     SAVE IN CASE NO DUMMY FOUND
2686 08D9 D043 A          ST      AC0,SYMA
2687 08DA 19F1 A          JMP     $LSPEC
2688 08DB C043 A $NOT:    LD      AC0,SYMA      ; MAKE SURE SYMBOL FOUND
2689 08DC 45DB A          BOC     NZRO,$RTN     ;     OK
2690 08DD 94C9*A          JSR     SYSERR        ; SYMBOL NOT FOUND
2691  ;
2692 08DE 5000 A $ERR:    LI      AC0,0         ; OUT OF STORAGE DURING SYMBOL T
2693 08DF 98C5*A          JMP     RUNERR        ;     CREATION
2694  ;
```

```
2752                        .PAGE   'SYNTAX ANALYZER'
2753                        .LOCAL
2754  ;
2755  ;         SYNTAX ANALYZER  (TABLE DRIVER)
2756  ;
2757  ;         THE SYNTAX ANALYZER CHECKS SYNTAX ACCORDING TO
2758  ;         PROCESSING DATA STORED IN THE SYNTAX TABLE.  THE TABLE
2759  ;         TELLS THE ANALYZER WHAT TO CHECK FOR AND WHAT TO DO WHEN
2760  ;         SOMETHING EXPECTED IS FOUND.
2761  ;
2762  ;
2763  ;                 RTS 0 - ERROR
2764  ;                 RTS 1 - STOP/END
2765  ;                 RTS 2 - NORMAL RETURN
2766  ;
2767 08E0 C055 A LINIT:  LD      AC0,ANEXT     ; NEW STATEMENT INITIALIZATION
2768 08E1 D056 A         ST      AC0,TNEXT
2769 08E2 C01B A         LD      AC0,OUTDVX    ; OUTDVC = DEFAULT (TTY)
2770 08E3 D01C A         ST      AC0,OUTDVC
2771 08E4 8000 A         RTS
2772 08E5 6400 A SYNTAX: PULL    AC0           ; SAVE RETURN ADDR IN SYNRTN
2773 08E6 D050 A         ST      AC0,SYNRTN
2774 08E7 6000 A         PUSH    AC0
2775 08E8 5104 A         LI      AC1,4
2776 08E9 94C2*A         JSR     CLRST1        ; RESET SINGLE-LINE STACKS
2777 08EA 5000 A         LI      AC0,0
2778 08EB D03A A         ST      AC0,NTYPE
2779 08EC D03B A         ST      AC0,STYPE
2780 08ED 94A3*A         JSR     SCAN          ; GET 1ST ITEM
2781 08EE 8000 A         RTS     0             ;    (ERROR)
2782 08EF CD69*A         LD      AC3,=SYNTAB   ; START WITH FIRST ENTRY IN TABL
2783 08F0 C300 A NXTSYN: LD      AC0,(AC3)     ; LOAD NEXT TABLE ITEM
2784 08F1 5101 A         LI      AC1,1
2785 08F2 7B01 A         AISZ    AC3,1         ; ADVANCE POINTER FOR NEXT TIME
2786 08F3 4412 A         BOC     BIT1,$2       ; CHECK ITEM TYPE
2787 08F4 4306 A         BOC     BIT0,CHKTKN
2788  ;
2789  ;         CHECK TOKEN CLASS
2790  ;
2791 08F5 2C04 A         SHR     AC0,2,0       ; CHECK CLASS
2792 08F6 5500 A         RAND    AC0,AC1
2793 08F7 2C02 A         SHR     AC0,1,0
2794 08F8 F041 A         SKNE    AC0,CLASS     ; CHECK CLASS
2795 08F9 1909 A         JMP     $M            ;    MATCH
2796 08FA 1905 A         JMP     $N            ;    NO MATCH
2797  ;
2798  ;         CHECK TOKEN VALUE
2799  ;
2800 08FB 2C04 A CHKTKN: SHR     AC0,2,0       ; CHECK TOKEN VALUE
2801 08FC 5500 A         RAND    AC0,AC1
2802 08FD 2C02 A         SHR     AC0,1,0
2803 08FE F042 A         SKNE    AC0,SYMB
2804 08FF 1903 A         JMP     $M
2805 0900 79FF A $N:     AISZ    AC1,-1        ; NO MATCH:
```

```
2806 0901 7B02 A          AISZ    AC3,2          ;    SKIP NEXT IF SKIP BIT IS CL
2807 0902 19ED A          JMP     NXTSYN
2808 0903 7900 A $M:      AISZ    AC1,0          ; MATCH:
2809 0904 7B02 A          AISZ    AC3,2          ;    SKIP NEXT IF SKIP BIT IS SE
2810 0905 19EA A          JMP     NXTSYN
2811  ;
2812 0906 4313 A $2:      BOC     BIT0,CTRL
2813 0907 420E A          BOC     POS,$ERR
2814  ;
2815  ;        CHECK A SPECIAL CASE
2816  ;
2817 0908 2C04 A          SHR     AC0,2,0
2818 0909 5500 A          RAND    AC0,AC1
2819 090A 2C02 A          SHR     AC0,1,0
2820 090B A8D5*A          AND     AC0,=0FF       ; MASK ROUTINE NUMBER
2821 090C 41F3 A          BOC     ZR0,$N         ; NO ROUTINE - ASSUME NO MATCH
2822 090D 5E00 A          RCPY    AC0,AC2
2823 090E E94B*A          ADD     AC2,=TAB2      ; DETERMINE ROUTINE ADDRESS
2824 090F CA00 A          LD      AC2,(AC2)
2825 0910 7A00 A          AISZ    AC2,0
2826 0911 1901 A          JMP     .+2
2827 0912 19ED A          JMP     $N             ; NO ROUTINE - ASSUME NO MATCH
2828 0913 1600 A          JSR     (AC2)          ; CALL ROUTINE - MAY USE AC0,AC2
2829 0914 19EE A          JMP     $M             ;    MATCH
2830 0915 19EA A          JMP     $N             ;    NO MATCH
2831  ;
2832  ;        ERROR ENTRY - PRINT MESSAGE AND RETURN
2833  ;
2834 0916 2C04 A $ERR:    SHR     AC0,2,0        ; ERROR ENTRY
2835 0917 94C3*A SYNERR:  JSR     ERROR          ; ERROR DURING SYNTAX ANALYSIS
2836 0918 94C1*A          JSR     CLRSTK         ; CLEAR STACK AND RETURN
2837 0919 9850 A          JMP     @SYNRTN
2838  ;
2839  ;        CONTROL ENTRY PROCESSING
2840  ;
2841 091A 2C04 A CTRL:    SHR     AC0,2,0
2842 091B 5500 A          RAND    AC0,AC1        ; SET NTYPE ACCORDING TO CONTROL
2843 091C D43A A          ST      AC1,NTYPE
2844 091D 2C02 A          SHR     AC0,1,0        ; SET STYPE ACCORDING TO CONTROL
2845 091E 5101 A          LI      AC1,1
2846 091F 5500 A          RAND    AC0,AC1
2847 0920 D43B A          ST      AC1,STYPE
2848 0921 2C02 A CHKMOD:  SHR     AC0,1,0        ; IF (MODE FIELD) AND (MODE BITS
2849 0922 5D00 A          RCPY    AC0,AC1        ;    THEN EXECUTE THE ROUTINE
2850 0923 A813 A          AND     AC0,MODE
2851 0924 410B A          BOC     ZR0,NXTSTA
2852 0925 5C40 A          RCPY    AC1,AC0
2853 0926 2C08 A          SHR     AC0,4,0        ; GET ROUTINE ADDRESS
2854 0927 A8D3*A          AND     AC0,=07F
2855 0928 4107 A          BOC     ZR0,NXTSTA     ; (ZERO - SKIP)
2856 0929 5E00 A          RCPY    AC0,AC2
2857 092A E92F*A          ADD     AC2,=TAB2
2858 092B CA00 A          LD      AC2,(AC2)
2859 092C 9499*A          JSR     SAVE
```

```
2860 092D 7A00 A          AISZ    AC2,0
2861 092E 1600 A          JSR     (AC2)          ; CALL THE ROUTINE IF NON-ZERO A
2862 092F 949A*A          JSR     RESTOR
2863 0930 C3FF A NXTSTA:  LD      AC0,-1(AC3)    ; CHECK FOR END OF BRANCH
2864 0931 4204 A          BOC     POS,$CTL2
2865 0932 5302 A          LI      AC3,2          ; END OF BRANCH -
2866 0933 949C*A          JSR     PULL           ;    CHECK THAT OPERAND STACK IS
2867 0934 8002 A          RTS     2              ; GO ON TO NEXT LINE
2868 0935 94C9*A          JSR     SYSERR         ; OPERAND STACK NOT EMPTY AT END
2869 0936 C300 A $CTL2:   LD      AC0,(AC3)      ; LOAD 2ND CONTROL WORD
2870 0937 7B01 A          AISZ    AC3,1
2871 0938 4301 A          BOC     BIT0,.+2
2872 0939 1904 A          JMP     $POP
2873 093A 9499*A          JSR     SAVE           ; SCAN BIT SET - CALL SCAN
2874 093B 94A3*A          JSR     SCAN
2875 093C 989A*A          JMP     RESTOR         ;    ERROR - RESTORE REG & RTS
2876 093D 949A*A          JSR     RESTOR         ; NORMAL RETURN: RESTORE, CHECK
2877 093E 2C02 A $POP:    SHR     AC0,1,0        ; CHECK FOR POP OF STATE FROM ST
2878 093F 4301 A          BOC     BIT0,.+2
2879 0940 1905 A          JMP     $PUSH
2880 0941 5301 A          LI      AC3,1          ; PULL TABLE ADDRESS FROM STACK
2881 0942 949C*A          JSR     PULL
2882 0943 94C9*A          JSR     SYSERR         ; STACK EMPTY
2883 0944 5F00 A          RCPY    AC0,AC3
2884 0945 19AA A          JMP     NXTSYN
2885 0946 2C02 A $PUSH:   SHR     AC0,1,0
2886 0947 A8D3*A          AND     AC0,=07F       ; DETERMINE ADDRESS TO PUSH
2887 0948 4109 A          BOC     ZR0,$NEXT      ;    ZERO STATE - DON'T PUSH
2888 0949 5E00 A          RCPY    AC0,AC2
2889 094A E910*A          ADD     AC2,=TAB1
2890 094B CA00 A          LD      AC2,(AC2)
2891 094C 5C80 A          RCPY    AC2,AC0
2892 094D 9499*A          JSR     SAVE
2893 094E 5301 A          LI      AC3,1          ; PUSH STATE ONTO SYNTAX STACK
2894 094F 949B*A          JSR     PUSH
2895 0950 1906 A          JMP     $SFULL         ; STACK FULL
2896 0951 949A*A          JSR     RESTOR
2897 0952 CFFF A $NEXT:   LD      AC3,-1(AC3)    ; PROCESS NEXT STATE FIELD
2898 0953 2F12 A          SHR     AC3,9,0
2899 0954 ED06*A          ADD     AC3,=TAB1
2900 0955 CF00 A          LD      AC3,(AC3)
2901 0956 1999 A          JMP     NXTSYN
2902  ;
2903 0957 5012 A $SFULL:  LI      AC0,18         ; STACK FULL - EXPR NEST TOO DEE
2904 0958 19BE A          JMP     SYNERR
2905  ;
2906 0959 095C T          .POOL   3
     095A 15E6 T
     095B 1582 T
2907  ;
```

```
2908                        .PAGE   'SYNTAX TABLES'
2909    ;
2910    ;  TABLE ENTRY FORMATS ARE:
2911    ;
2912    ;         CHK1     VALUE,S          CHECK CLASS
2913    ;         CHK2     VALUE,S          CHECK SYMB (TOKEN)
2914    ;         CHK3     RTN,S            CHECK SPECIAL CASE
2915    ;
2916    ;         CTL1     E,RTN,MODE,D&N   CONTROL
2917    ;         CTL2     NEXT,PUSH,P&SC   CONTROL
2918    ;
2919    ;         ERR      ERROR NUMBER
2920    ;
2921    ;      WHERE:
2922    ;
2923    ;         S=0 --> SKIP 2 IF NO MATCH, NEXT IF MATCH
2924    ;         S=1 --> SKIP 2 IF MATCH, NEXT IF NO MATCH
2925    ;         E=1 --> END OF STATEMENT (GO TO NEXT LINE)
2926    ;         D=1 --> STYPE=1 (DATA STATEMENT SCANNING)
2927    ;         N=1 --> NTYPE=1 (STMT# EXPECTED - OTHERWISE FLT.PT.)
2928    ;         P=1 --> POP SYNTAX ANALYZER STATE ADDRESS
2929    ;         SC=1 -> SCAN FOR NEXT TOKEN (OTHERWISE REPARSE)
2930    ;         RTN  =  EXECUTION ROUTINE NUMBER (EXEC WHEN MODE MATCHES)
2931    ;         NEXT =  NEXT STATE NUMBER
2932    ;         PUSH =  STATE NUMBER TO PUSH (IF NON-ZERO)
2933    ;
2934    ;
2935    ;
2936    ;         *** SYNTAX TABLE ***
2937    ;
2938 095C          SYNTAB: CTL1    0,72,15,0     ; INITIALIZE FOR NEW STATEMENT
2939 095D                  CTL2    1,0,0         ;    AND GO TO STATE 1
2940    ;
2941    ;         START OF LINE
2942    ;
2943 095E          ST01:   CHK1    13,0          ; CLASS=EOL ?
2944 095F                  CTL1    1,0,0,0       ;    YES - DELETE LINE (OR COMME
2945 0960                  CTL2    0,0,0
2946 0961                  CHK1    27,0          ; CLASS=COMMAND(DIRECT EXEC) ?
2947 0962                  CTL1    0,0,0,0
2948 0963                  CTL2    7,0,0
2949 0964                  CHK2    128,0         ; SYMB=REM ?
2950 0965                  CTL1    1,0,0,0       ;    YES - END OF STMT
2951 0966                  CTL2    0,0,0
2952 0967                  CHK3    9,0           ; MODE=2 (DIRECT EXEC) ?
2953 0968                  CTL1    0,0,0,0       ;    YES - SKIP TO ALLOWABLE ITE
2954 0969                  CTL2    2,0,0
2955 096A                  CHK2    136,0         ; SYMB=FOR ?
2956 096B                  CTL1    0,0,0,0
2957 096C                  CTL2    61,0,1
2958 096D                  CHK2    139,0         ; SYMB=NEXT ?
2959 096E                  CTL1    0,0,0,0
2960 096F                  CTL2    66,0,1
2961 0970                  CHK2    144,0         ; SYMB=DATA ?
```

```
2962 0971                CTL1    0,0,0,2
2963 0972                CTL2    67,0,1
2964 0973                CHK2    147,0          ; SYMB=DEF ?
2965 0974                CTL1    0,0,0,0
2966 0975                CTL2    87,0,1
2967 0976                CHK2    129,0          ; SYMB=DIM ?
2968 0977                CTL1    0,0,0,0
2969 0978                CTL2    53,0,1
2970  ;
2971  ;       THE FOLLOWING THINGS ARE ALLOWED FOLLOWING IF...THEN
2972  ;
2973 0979       ST02:    CHK1    15,0           ; CLASS=SIMPLE-ID ?
2974 097A                CTL1    0,0,0,0
2975 097B                CTL2    19,0,0
2976 097C                CHK1    16,0           ; CLASS=ARRAY-ID ?
2977 097D                CTL1    0,0,0,0
2978 097E                CTL2    20,0,0
2979 097F                CHK1    17,0           ; CLASS=STRING-ID ?
2980 0980                CTL1    0,0,0,0
2981 0981                CTL2    21,0,0
2982 0982                CHK1    18,0           ; CLASS=STRING-ARRAY ?
2983 0983                CTL1    0,0,0,0
2984 0984                CTL2    21,0,0
2985 0985                CHK2    130,0          ; SYMB=LET ?
2986 0986                CTL1    0,0,0,0
2987 0987                CTL2    18,0,1
2988 0988                CHK2    134,0          ; SYMB=IF ?
2989 0989                CTL1    0,0,0,0
2990 098A                CTL2    33,59,1
2991 098B                CHK2    141,0          ; SYMB=PRINT ?
2992 098C                CTL1    0,0,0,0
2993 098D                CTL2    12,0,1
2994 098E                CHK2    145,0          ; SYMB=READ ?
2995 098F                CTL1    0,51,10,0
2996 0990                CTL2    74,0,1
2997 0991                CHK2    146,0          ; SYMB=RESTORE ?
2998 0992                CTL1    0,52,10,0
2999 0993                CTL2    4,0,1
3000 0994                CHK2    150,0          ; SYMB=RANDOMIZE ?
3001 0995                CTL1    0,53,10,0
3002 0996                CTL2    4,0,1
3003 0997                CHK2    153,0          ; SYMB=POKE ?
3004 0998                CTL1    0,29,10,0
3005 0999                CTL2    33,85,1
3006 099A                CHK2    154,0          ; SYMB=OUT ?
3007 099B                CTL1    0,29,10,0
3008 099C                CTL2    33,85,1
3009 099D                CHK2    200,0          ; SYMB=CALL ?
3010 099E                CTL1    0,0,0,0
3011 099F                CTL2    33,70,1
3012 09A0                CHK2    159,0          ; SYMB=WIDTH ?
3013 09A1                CTL1    0,0,0,0
3014 09A2                CTL2    33,79,1
3015 09A3                CHK2    149,0          ; SYMB=TRACE ?
```

```
3016 09A4                  CTL1    0,0,0,0
3017 09A5                  CTL2    6,0,1
3023 09A6                  CHK2    156,0          ; SYMB=SIZE ?
3024 09A7                  CTL1    0,69,10,0
3025 09A8                  CTL2    4,0,1
3026 09A9                  CHK2    199,0          ; SYMB=WAIT ?
3027 09AA                  CTL1    0,29,10,0
3028 09AB                  CTL2    33,93,1
3029 09AC                  CHK2    151,0          ; SYMB=TWAIT ?
3030 09AD                  CTL1    0,0,0,0
3031 09AE                  CTL2    33,78,1
3032 09AF                  CHK2    142,0          ; SYMB=STOP ?
3033 09B0                  CTL1    0,4,8,0
3034 09B1                  CTL2    3,0,1
3035 09B2                  CHK2    143,0          ; SYMB=END ?
3036 09B3                  CTL1    0,4,9,0
3037 09B4                  CTL2    3,0,1
3038 09B5                  CHK2    214,0          ; SYMB=EXEC ?
3039 09B6                  CTL1    0,5,10,2
3040 09B7                  CTL2    82,0,1
3041 09B8                  CHK1    28,0           ; CLASS=DISK-COMMAND ?
3042 09B9                  CTL1    0,5,10,2
3043 09BA                  CTL2    96,0,1
3044 09BB                  CHK3    9,0            ; MODE=2 (DIRECT EXEC) ?
3045 09BC                  ERR     4              ;    YES - LIST ENDED
3046 09BD 0000 A           .WORD   0              ;    NO - FOLLOWING ARE OK
3047 09BE                  CHK2    131,0          ; SYMB=GOTO ?
3048 09BF                  CTL1    0,0,0,1        ;    YES - SCAN FOR STMT-NO.
3049 09C0                  CTL2    5,0,1
3050 09C1                  CHK2    132,0          ; SYMB=GOSUB ?
3051 09C2                  CTL1    0,1,8,1        ;    YES - SAVE NEXTAD THEN SCAN
3052 09C3                  CTL2    5,0,1
3053 09C4                  CHK2    133,0          ; SYMB=RETURN ?
3054 09C5                  CTL1    0,2,8,0        ;    YES - PULL NEXTAD
3055 09C6                  CTL2    3,0,1
3056 09C7                  CHK2    148,0          ; SYMB=ON ?
3057 09C8                  CTL1    0,0,0,0
3058 09C9                  CTL2    33,80,1
3059 09CA                  CHK2    140,0          ; SYMB=INPUT ?
3060 09CB                  CTL1    0,48,8,0
3061 09CC                  CTL2    71,0,1
3062 09CD                  ERR     4
3063  ;
3064  ;        NEXT ENTRY SHOULD BE END-OF-LINE
3065  ;
3066 09CE        ST03:     CHK1    13,0           ; CLASS=EOL ?
3067 09CF                  CTL1    1,0,0,0        ;    YES - GO ON TO NEXT
3068 09D0                  CTL2    0,0,0
3069 09D1                  ERR     6              ;    NO - ERROR
3070  ;
3071  ;        CHECK FOR END-OF-LINE OR STATEMENT SEPARATOR
3072  ;
3073 09D2        ST04:     CHK1    13,0           ; CLASS=EOL ?
3074 09D3                  CTL1    1,0,0,0        ;    YES - END OF STMT
```

```
3075 09D4                CTL2    0,0,0
3076 09D5                CHK1    14,0            ; CLASS=SEP ?
3077 09D6                CTL1    0,0,0,0         ;    YES - PROCESS NEXT STMT IN
3078 09D7                CTL2    0,0,1
3079 09D8                ERR     6
3080  ;
3081  ;       PROCESS GOTO OR GOSUB NEXT ADDRESS
3082  ;
3083 09D9       ST05:    CHK1    20,0            ; CLASS=STMT-NO.  ?
3084 09DA                CTL1    0,3,8,0
3085 09DB                CTL2    3,0,1
3086 09DC                ERR     8               ;    NO
3087  ;
3088  ;       TRACE - ON OR OFF
3089  ;
3090 09DD       ST06:    CHK2    148,0           ; SYMB=ON ?
3091 09DE                CTL1    0,45,10,0
3092 09DF                CTL2    4,0,1
3093 09E0                CHK2    152,0           ; SYMB=OFF ?
3094 09E1                CTL1    0,46,10,0
3095 09E2                CTL2    4,0,1
3096 09E3                CTL1    0,45,10,0       ; NEITHER - TURN ON TRACE
3097 09E4                CTL2    4,0,0
3098  ;
3099  ;       DIRECT EXEC COMMAND
3100  ;
3101 09E5       ST07:    CHK3    9,0             ; MODE=2 ?
3102 09E6                CTL1    0,5,10,1
3103 09E7                CTL2    8,0,1
3104 09E8                ERR     4               ;    NO
3105 09E9       ST08:    CHK1    20,0            ; CLASS=STMT# ?
3106 09EA                CTL1    0,6,10,1        ;    YES - SAVE
3107 09EB                CTL2    9,0,1
3108 09EC                CTL1    0,0,0,0
3109 09ED                CTL2    11,0,0
3110 09EE       ST09:    CHK2    COMMA,0         ; SYMB=COMMA ?
3111 09EF                CTL1    0,0,0,1         ;    YES - GO ON
3112 09F0                CTL2    10,0,1
3113 09F1                CTL1    0,0,0,0
3114 09F2                CTL2    11,0,0
3115 09F3       ST10:    CHK1    20,0            ; CLASS=STMT# ?
3116 09F4                CTL1    0,7,10,0
3117 09F5                CTL2    11,0,1
3118 09F6                ERR     8
3119 09F7       ST11:    CHK1    13,0            ; CLASS=EOL ?
3120 09F8                CTL1    1,8,10,0        ;    EXECUTE COMMAND
3121 09F9                CTL2    0,0,0
3122 09FA                ERR     6               ; NOT EOL
3123  ;
3124  ;       PRINT STATEMENT
3125  ;
3126 09FB       ST12:    CHK1    13,0            ; CLASS=EOL ?
3127 09FC                CTL1    1,10,10,0       ;    YES - CRLF
3128 09FD                CTL2    0,0,0
```

```
3129 09FE                CHK1    14,0          ; CLASS=SEP ?
3130 09FF                CTL1    0,10,10,0
3131 0A00                CTL2    0,0,1
3132 0A01        ST13:   CHK1    12,0          ; CLASS=DELIMITER ?
3133 0A02                CTL1    0,11,10,0
3134 0A03                CTL2    14,0,1
3135 0A04                CHK3    16,0          ; CLASS=STRING ?
3136 0A05                CTL1    0,0,0,0
3137 0A06                CTL2    46,15,0
3138 0A07                CHK1    8,0           ; CLASS=PRINT-FCN ?
3139 0A08                CTL1    0,29,10,0
3140 0A09                CTL2    37,16,1
3141 0A0A                CTL1    0,0,0,0       ; EXPRESSION
3142 0A0B                CTL2    33,17,0
3143 0A0C        ST14:   CHK1    13,0          ; CLASS=EOL ? (AFTER DELIMITER)
3144 0A0D                CTL1    1,0,0,0
3145 0A0E                CTL2    0,0,0
3146 0A0F                CHK1    14,0          ; CLASS=SEP ?
3147 0A10                CTL1    0,0,0,0
3148 0A11                CTL2    0,0,1
3149 0A12                CTL1    0,0,0,0       ;    OTHER - REPARSE
3150 0A13                CTL2    13,0,0
3151 0A14        ST15:   CTL1    0,13,10,0     ; PRINT STRING
3152 0A15                CTL2    12,0,1
3153 0A16        ST16:   CTL1    0,14,10,0     ; EXECUTE PRINT FUNCTION
3154 0A17                CTL2    12,0,1
3155 0A18        ST17:   CTL1    0,15,10,0     ; PRINT EXPRESSION VALUE
3156 0A19                CTL2    12,0,0
3157  ;
3158  ;       LET STATEMENT
3159  ;
3160 0A1A        ST18:   CHK1    15,0          ; CLASS=IDENT ?
3161 0A1B        ST19:   CTL1    0,26,10,0
3162 0A1C                CTL2    22,0,1
3163 0A1D                CHK1    16,0          ; CLASS=ARRAY ?
3164 0A1E        ST20:   CTL1    0,21,10,0
3165 0A1F                CTL2    30,24,0
3166 0A20                CHK1    17,0          ; CLASS=STRING-ID ?
3167 0A21        ST21:   CTL1    0,0,0,0
3168 0A22                CTL2    46,25,0
3169 0A23                CHK1    18,0          ; CLASS=STRING-ARRAY ?
3170 0A24                CTL1    0,0,0,0
3171 0A25                CTL2    46,25,0
3172 0A26                ERR     8
3173 0A27        ST22:   CHK2    EQUAL,0       ; SYMB=EQUAL ?
3174 0A28                CTL1    0,0,0,0       ;    YES - CHECK EXPR
3175 0A29                CTL2    33,23,1
3176 0A2A                ERR     8
3177 0A2B        ST23:   CHK1    13,0          ; CLASS=EOL ?
3178 0A2C                CTL1    1,18,10,0
3179 0A2D                CTL2    0,0,0
3180 0A2E                CHK1    14,0          ; CLASS=SEP ?
3181 0A2F                CTL1    0,18,10,0
3182 0A30                CTL2    0,0,1
```

```
3183 0A31                ERR     6
3184 0A32        ST24:   CTL1    0,31,10,0       ; SAVE ARRAY ITEM ADDR
3185 0A33                CTL2    22,0,1
3186  ;
3187  ;       STRING ASSIGN PROCESSING
3188  ;
3189 0A34        ST25:   CTL1    0,0,0,0         ; SCAN PAST STRING
3190 0A35                CTL2    26,0,1
3191 0A36        ST26:   CHK2    EQUAL,0         ; SYMB=EQUAL ?
3192 0A37                CTL1    0,0,0,0
3193 0A38                CTL2    46,27,1
3194 0A39                ERR     8
3195 0A3A        ST27:   CTL1    0,19,10,0       ; STORE STRING
3196 0A3B                CTL2    28,0,1
3197 0A3C        ST28:   CHK2    02B,0           ; SYMB='+' ?
3198 0A3D                CTL1    0,0,0,0         ;    YES
3199 0A3E                CTL2    46,29,1
3200 0A3F                CHK1    13,0            ; CLASS=EOL ?
3201 0A40                CTL1    1,12,10,0
3202 0A41                CTL2    0,0,0
3203 0A42                CHK1    14,0            ; CLASS=SEP ?
3204 0A43                CTL1    0,12,10,0
3205 0A44                CTL2    0,0,1
3206 0A45                ERR     6
3207 0A46        ST29:   CTL1    0,20,10,0       ; CONCATENATE STRING
3208 0A47                CTL2    28,0,1
3209  ;
3210  ;       GET ARRAY SUBSCRIPT(S)
3211  ;
3212 0A48        ST30:   CTL1    0,25,10,0       ; LPAREN; GET EXPRESSION
3213 0A49                CTL2    33,31,1
3214 0A4A        ST31:   CHK2    COMMA,0         ; SYMB=COMMA ?
3215 0A4B                CTL1    0,22,10,0
3216 0A4C                CTL2    33,32,1
3217 0A4D                CHK1    11,0            ; CLASS=RPAREN ?
3218 0A4E                CTL1    0,23,10,0       ;    YES - 1 SUBSCRIPT
3219 0A4F                CTL2    0,0,2
3220 0A50                ERR     8
3221 0A51        ST32:   CHK1    11,0            ; CLASS=RPAREN ?
3222 0A52                CTL1    0,24,10,0       ;    YES - 2 SUBSCRIPTS
3223 0A53                CTL2    0,0,2
3224 0A54                ERR     8
3225  ;
3226  ;       EXPRESSION
3227  ;
3228 0A55        ST33:   CHK1    1,0             ; CLASS=UNARY-OP ?
3229 0A56                CTL1    0,28,10,0
3230 0A57                CTL2    34,0,1
3231 0A58                CHK1    3,0             ; CLASS=NOT ?
3232 0A59                CTL1    0,28,10,0
3233 0A5A                CTL2    34,0,1
3234 0A5B        ST34:   CHK1    10,0            ; CLASS=LPAREN ?
3235 0A5C                CTL1    0,25,10,0
3236 0A5D                CTL2    33,39,1
```

```
3237 0A5E                CHK1    21,0          ; CLASS=CONST ?
3238 0A5F                CTL1    0,27,10,0
3239 0A60                CTL2    39,0,1
3240 0A61                CHK1    15,0          ; CLASS=IDENT ?
3241 0A62                CTL1    0,26,10,0
3242 0A63                CTL2    39,0,1
3243 0A64                CHK1    16,0          ; CLASS=ARRAY ?
3244 0A65                CTL1    0,21,10,0
3245 0A66                CTL2    30,35,0
3246 0A67                CHK1    6,0           ; CLASS=B.I.FCN ?
3247 0A68                CTL1    0,29,10,0
3248 0A69                CTL2    37,36,1
3249 0A6A                CHK1    24,0          ; CLASS=SPECIAL-IDENT ?
3250 0A6B                CTL1    0,71,10,0
3251 0A6C                CTL2    39,0,1
3252 0A6D                CHK3    16,0          ; SPECIAL CLASS=STRING ?
3253 0A6E                CTL1    0,0,0,0
3254 0A6F                CTL2    46,41,0
3255 0A70                CHK1    9,0           ; CLASS=STR->NUM FCN ?
3256 0A71                CTL1    0,29,10,0
3257 0A72                CTL2    46,44,1
3258 0A73                CHK1    7,0           ; CLASS=USER-FCN ?
3259 0A74                CTL1    0,30,10,0     ;    SAVE FUNCTION INFO
3260 0A75                CTL2    37,91,1       ;    GET ARGUMENT
3261 0A76                ERR     8
3262 0A77        ST35:   CTL1    0,31,10,0     ; SAVE ARRAY DATA VALUE
3263 0A78                CTL2    39,0,1
3264 0A79        ST36:   CTL1    0,32,10,0     ; EXECUTE BUILT-IN FUNCTION
3265 0A7A                CTL2    39,0,1
3266  ;
3267  ;       GET FUNCTION ARGUMENT
3268  ;
3269 0A7B        ST37:   CTL1    0,0,0,0       ; CHECK EXPRESSION
3270 0A7C                CTL2    33,38,0
3271 0A7D        ST38:   CHK1    11,0          ; CLASS=RPAREN ?
3272 0A7E                CTL1    0,0,0,0       ; YES - POP STATE
3273 0A7F                CTL2    0,0,2
3274 0A80                ERR     8
3275  ;
3276  ;       CHECK OPERATOR
3277  ;
3278 0A81        ST39:   CHK1    1,0           ; CLASS=OP1 ?
3279 0A82                CTL1    0,33,10,0
3280 0A83                CTL2    33,0,1
3281 0A84                CHK1    2,0           ; CLASS=OP2 ?
3282 0A85                CTL1    0,33,10,0
3283 0A86                CTL2    33,0,1
3284 0A87                CHK1    4,0           ; CLASS=OPLOG ?
3285 0A88                CTL1    0,33,10,0
3286 0A89                CTL2    33,0,1
3287 0A8A                CHK1    5,0           ; CLASS=OPREL ?
3288 0A8B                CTL1    0,33,10,0
3289 0A8C                CTL2    33,0,1
3290 0A8D                CHK1    11,0          ; CLASS=RPAREN ?
```

```
3291 0A8E                CTL1    0,0,0,0
3292 0A8F                CTL2    40,0,0
3293 0A90                CHK3    34,1            ; STACKTOP=ST39 ?
3294 0A91                CTL1    0,0,0,0         ;    NO - POP
3295 0A92                CTL2    0,0,2
3296 0A93                ERR     8               ;    YES - MISSING RIGHT PAREN
3297 0A94       ST40:    CHK3    34,0            ; STACKTOP=ST39 ?
3298 0A95                CTL1    0,33,10,0       ;    YES - EXEC 33 & POP (SCAN)
3299 0A96                CTL2    0,0,3
3300 0A97                CTL1    0,0,0,0         ;    NO - POP (REPARSE)
3301 0A98                CTL2    0,0,2
3302  ;
3303  ;       STRING COMPARISON PROCESSING
3304  ;
3305 0A99       ST41:    CTL1    0,0,0,0         ; SCAN PAST STRING
3306 0A9A                CTL2    42,0,1
3307 0A9B       ST42:    CHK1    5,0             ; CLASS=RELATION ?
3308 0A9C                CTL1    0,60,-10,0
3309 0A9D                CTL2    46,43,1
3310 0A9E                ERR     8
3311 0A9F       ST43:    CTL1    0,61,10,0       ; COMPARE STRINGS
3312 0AA0                CTL2    39,0,1
3313  ;
3314  ;       STRING OPERAND FUNCTION
3315  ;
3316 0AA1       ST44:    CTL1    0,0,0,0         ; SCAN PAST STRING
3317 0AA2                CTL2    45,0,1
3318 0AA3       ST45:    CHK1    11,0            ; CLASS=RPAREN ?
3319 0AA4                CTL1    0,62,10,0
3320 0AA5                CTL2    39,0,1
3321 0AA6                ERR     8
3322  ;
3323  ;       CHECK STRING OPERAND
3324  ;
3325 0AA7       ST46:    CHK2    180,0           ; SYMB=MID$ ?
3326 0AA8                CTL1    0,29,10,0
3327 0AA9                CTL2    47,50,1
3328 0AAA                CHK1    19,0            ; CLASS=SUBSTRING FUNCTION ?
3329 0AAB                CTL1    0,29,10,0
3330 0AAC                CTL2    47,97,1
3331 0AAD       ST47:    CHK1    17,0            ; CLASS=STRING-ID ?
3332 0AAE                CTL1    0,58,10,0
3333 0AAF                CTL2    0,0,2
3334 0AB0                CHK1    18,0            ; CLASS=STRING-ARRAY ?
3335 0AB1                CTL1    0,76,10,0
3336 0AB2                CTL2    30,48,0
3337 0AB3                CHK1    22,0            ; CLASS=STRING-CONST ?
3338 0AB4                CTL1    0,59,10,0
3339 0AB5                CTL2    0,0,2
3340 0AB6                CHK1    23,0            ; CLASS=STRING-FCN ?
3341 0AB7                CTL1    0,29,10,0
3342 0AB8                CTL2    37,49,1
3343 0AB9                ERR     20
3344 0ABA       ST48:    CTL1    0,73,10,0       ; UPDATE STRING POINTER ON STACK
```

```
3345 0ABB                CTL2    0,0,2
3346 0ABC        ST49:   CTL1    0,74,10,0       ; EXECUTE NUM->STR FUNCTION
3347 0ABD                CTL2    0,0,2
3348 0ABE        ST50:   CTL1    0,0,0,0         ; SCAN PAST STRING
3349 0ABF                CTL2    51,0,1
3350 0AC0        ST51:   CHK2    COMMA,0         ; CHECK ADDITIONAL OPERANDS
3351 0AC1                CTL1    0,23,10,0
3352 0AC2                CTL2    30,52,0         ; GET 1 OR 2 NUMBERS
3353 0AC3                ERR     8
3354 0AC4        ST52:   CTL1    0,75,10,0       ; EXECUTE SUBSTRING FUNCTION
3355 0AC5                CTL2    0,0,2
3356 0AC6        ST97:   CTL1    0,0,0,0         ; LEFT$,RIGHT$ - ONE NUMERIC ARG
3357 0AC7                CTL2    98,0,1
3358 0AC8        ST98:   CHK2    COMMA,0
3359 0AC9                CTL1    0,25,10,0
3360 0ACA                CTL2    33,99,1
3361 0ACB                ERR     8
3362 0ACC        ST99:   CHK1    11,0            ; CLASS=RPAREN ?
3363 0ACD                CTL1    0,23,10,0
3364 0ACE                CTL2    52,0,0
3365 0ACF                ERR     8
3366  ;
3367  ;       DIM STATEMENT
3368  ;
3369 0AD0        ST53:   CHK1    16,0            ; CLASS=ARRAY ?
3370 0AD1                CTL1    0,35,4,0
3371 0AD2                CTL2    54,0,1
3372 0AD3                CHK1    18,0            ; CLASS=STRING-ARRAY ?
3373 0AD4                CTL1    0,35,4,0
3374 0AD5                CTL2    54,0,1
3375 0AD6                ERR     20
3376 0AD7        ST54:   CHK1    21,0            ; CLASS=CONST ?
3377 0AD8                CTL1    0,36,4,0
3378 0AD9                CTL2    55,0,1
3379 0ADA                ERR     8
3380 0ADB        ST55:   CHK2    COMMA,0         ; SYMB=COMMA ?
3381 0ADC                CTL1    0,0,0,0
3382 0ADD                CTL2    56,0,1
3383 0ADE                CHK1    11,0            ; CLASS=RPAREN ?
3384 0ADF                CTL1    0,0,0,0
3385 0AE0                CTL2    58,0,1
3386 0AE1                ERR     8
3387 0AE2        ST56:   CHK1    21,0            ; CLASS=CONST ?
3388 0AE3                CTL1    0,37,4,0        ;    YES - 2ND DIMENSION
3389 0AE4                CTL2    57,0,1
3390 0AE5                ERR     8
3391 0AE6        ST57:   CHK1    11,0            ; CLASS=RPAREN ?
3392 0AE7                CTL1    0,0,0,0
3393 0AE8                CTL2    58,0,1
3394 0AE9                ERR     8
3395 0AEA        ST58:   CHK2    COMMA,0         ; SYMB=COMMA ?
3396 0AEB                CTL1    0,0,0,0         ;    YES - PROCESS NEXT ARRAY
3397 0AEC                CTL2    53,0,1
3398 0AED                CTL1    0,0,0,0         ;    NO - CHECK EOL
```

```
3399 0AEE                  CTL2    4,0,0
3400  ;
3401  ;        IF STATEMENT
3402  ;
3403 0AEF        ST59:     CHK2    135,0          ; SYMB=THEN ?
3404 0AF0                  CTL1    0,38,10,1
3405 0AF1                  CTL2    60,0,1
3406 0AF2                  CHK3    9,0            ; MODE=DIRECT EXEC?
3407 0AF3                  ERR     4              ; YES - NO GOTO
3408 0AF4 0000 A           .WORD   0
3409 0AF5                  CHK2    131,0          ; SYMB=GOTO ?
3410 0AF6                  CTL1    0,38,10,1
3411 0AF7                  CTL2    5,0,1
3412 0AF8                  ERR     8
3413 0AF9          ST60:   CHK1    20,1           ; CLASS=STMT# ?
3414 0AFA                  CTL1    0,0,0,0        ; NO - CHECK STATEMENT
3415 0AFB                  CTL2    2,0,0
3416 0AFC                  CHK3    9,0            ; STMT# - CHECK IF COMMAND MODE
3417 0AFD                  ERR     4              ; YES - NO GOTO
3418 0AFE 0000 A           .WORD   0
3419 0AFF                  CTL1    0,3,8,0
3420 0B00                  CTL2    3,0,1
3421  ;
3422  ;        FOR - TO - STEP STATEMENT
3423  ;
3424 0B01          ST61:   CHK1    15,0           ; CLASS=IDENT ?
3425 0B02                  CTL1    0,39,12,0
3426 0B03                  CTL2    62,0,1
3427 0B04                  ERR     8
3428 0B05          ST62:   CHK2    EQUAL,0        ; SYMB=EQUAL ?
3429 0B06                  CTL1    0,0,0,0
3430 0B07                  CTL2    33,63,1
3431 0B08                  ERR     8
3432 0B09          ST63:   CHK2    137,0          ; SYMB=TO ?
3433 0B0A                  CTL1    0,40,8,0
3434 0B0B                  CTL2    33,64,1
3435 0B0C                  ERR     8
3436 0B0D          ST64:   CHK2    138,0          ; SYMB=STEP ?
3437 0B0E                  CTL1    0,41,8,0       ;    SAVE END VALUE
3438 0B0F                  CTL2    33,65,1
3439 0B10                  CHK1    13,0           ; CLASS=EOL ? (NO STEP)
3440 0B11                  CTL1    1,42,8,0       ;    SAVE END VALUE, SET STEP=1
3441 0B12                  CTL2    0,0,0
3442 0B13                  ERR     8
3443 0B14          ST65:   CHK1    13,0           ; CLASS=EOL ? (AFTER STEP)
3444 0B15                  CTL1    1,43,8,0
3445 0B16                  CTL2    0,0,0
3446 0B17                  ERR     6
3447  ;
3448  ;        NEXT STATEMENT
3449  ;
3450 0B18          ST66:   CHK1    15,0           ; CLASS=IDENT ?
3451 0B19                  CTL1    0,44,12,0
3452 0B1A                  CTL2    3,0,1
```

```
3453 0B1B                ERR     8
3454  ;
3455  ;        DATA STATEMENT
3456  ;
3457 0B1C      ST67:     CHK1    1,0             ; CLASS=OP1 (UNARY) ?
3458 0B1D                CTL1    0,0,0,2
3459 0B1E                CTL2    68,0,1
3460 0B1F      ST68:     CHK1    21,0            ; CLASS=CONST ?
3461 0B20                CTL1    0,0,0,0
3462 0B21                CTL2    69,0,1
3463 0B22                CHK1    22,0            ; CLASS=STRING-CONST ?
3464 0B23                CTL1    0,0,0,0
3465 0B24                CTL2    69,0,1
3466 0B25                ERR     20
3467 0B26      ST69:     CHK2    COMMA,0         ; SYMB=COMMA ?
3468 0B27                CTL1    0,0,0,2
3469 0B28                CTL2    67,0,1
3470 0B29                CTL1    0,0,0,0         ;    NO - CHECK EOL
3471 0B2A                CTL2    3,0,0
3472  ;
3473  ;        INPUT STATEMENT - CHECK FOR PROMPT STRING
3474  ;
3475 0B2B      ST71:     CHK1    22,1            ; CLASS=STRING-CONST ?
3476 0B2C                CTL1    0,0,0,0         ;    NO
3477 0B2D                CTL2    74,0,0
3478 0B2E                CTL1    0,59,10,0       ;    YES
3479 0B2F                CTL2    72,0,0
3480 0B30      ST72:     CTL1    0,13,10,0       ; PRINT STRING
3481 0B31                CTL2    73,0,1
3482 0B32      ST73:     CHK1    12,0            ; DELIMITER ?
3483 0B33                CTL1    0,11,10,0
3484 0B34                CTL2    74,0,1
3485 0B35                ERR     8
3486  ;
3487  ;        INPUT/READ STATEMENTS
3488  ;
3489 0B36      ST74:     CHK1    15,0            ; CLASS=IDENT ?
3490 0B37                CTL1    0,49,10,0
3491 0B38                CTL2    76,0,1
3492 0B39                CHK1    16,0            ; CLASS=ARRAY ?
3493 0B3A                CTL1    0,21,10,0
3494 0B3B                CTL2    30,75,0
3495 0B3C                CHK1    17,0            ; CLASS=STRING-ID ?
3496 0B3D                CTL1    0,0,0,0
3497 0B3E                CTL2    46,77,0
3498 0B3F                CHK1    18,0            ; CLASS=STRING-ARRAY ?
3499 0B40                CTL1    0,0,0,0
3500 0B41                CTL2    46,77,0
3501 0B42                ERR     20
3502 0B43      ST75:     CTL1    0,50,10,0       ; READ ARRAY ELEMENT
3503 0B44                CTL2    76,0,1
3504 0B45      ST76:     CHK2    COMMA,0         ; SYMB=COMMA ?
3505 0B46                CTL1    0,0,0,0
3506 0B47                CTL2    74,0,1
```

```
3507 0B48                CTL1    0,0,0,0      ;     NO - CHECK EOL/SEP
3508 0B49                CTL2    4,0,0
3509 0B4A       ST77:    CTL1    0,70,10,0    ; STRING INPUT/READ
3510 0B4B                CTL2    76,0,1
3511  ;
3512  ;        TWAIT STATEMENT
3513  ;
3514 0B4C       ST78:    CTL1    0,54,10,0    ; EXECUTE TIMED WAIT
3515 0B4D                CTL2    4,0,0        ; CHECK EOL/SEP
3516  ;
3517  ;        WIDTH STATEMENT
3518  ;
3519 0B4E       ST79:    CTL1    0,77,10,0    ; SAVE TERMINAL WIDTH
3520 0B4F                CTL2    4,0,0        ; CHECK EOL/SEP
3521  ;
3522  ;        CALL STATEMENT
3523  ;
3524 0B50       ST70:    CTL1    0,80,10,0    ; CALL SUBROUTINE
3525 0B51                CTL2    4,0,0        ; CHECK EOL/SEP
3526  ;
3527  ;        ON STATEMENT
3528  ;
3529 0B52       ST80:    CTL1    0,55,8,0     ; SAVE ON VALUE
3530 0B53                CTL2    81,0,0
3531 0B54       ST81:    CHK2    131,0        ; SYMB=GOTO ?
3532 0B55                CTL1    0,0,0,1
3533 0B56                CTL2    83,0,1
3534 0B57                CHK2    132,0        ; SYMB=GOSUB ?
3535 0B58                CTL1    0,1,8,1
3536 0B59                CTL2    83,0,1
3537 0B5A                CHK2    135,0        ; SYMB=THEN ?
3538 0B5B                CTL1    0,0,0,1
3539 0B5C                CTL2    83,0,1
3540 0B5D                ERR     8
3541 0B5E       ST83:    CHK1    20,0         ; CLASS=STMT# ?
3542 0B5F                CTL1    0,56,8,1
3543 0B60                CTL2    84,0,1
3544 0B61                ERR     8
3545 0B62       ST84:    CHK2    COMMA,0      ; SYMB=COMMA ?
3546 0B63                CTL1    0,0,0,1
3547 0B64                CTL2    83,0,1
3548 0B65                CHK1    13,0         ; CLASS=EOL ?
3549 0B66                CTL1    1,57,8,0
3550 0B67                CTL2    0,0,0
3551 0B68                ERR     6
3552  ;
3553  ;        POKE, OUT STATEMENTS
3554  ;
3555 0B69       ST85:    CHK2    COMMA,0      ; SYMB=COMMA ?
3556 0B6A                CTL1    0,24,10,0
3557 0B6B                CTL2    33,86,1
3558 0B6C                ERR     8
3559 0B6D       ST86:    CTL1    0,66,10,0    ; EXECUTE AFTER 2ND EXPRESSION
3560 0B6E                CTL2    4,0,0
```

```
3561  ;
3562  ;       DEF STATEMENT
3563  ;
3564 0B6F       ST87:   CHK3    17,0          ; MODE=8 (PASS2) ?
3565 0B70               CTL1    1,0,0,0       ;    YES - TREAT AS REM
3566 0B71               CTL2    0,0,0
3567 0B72               CHK1    7,0           ; CLASS=USER-FCN ?
3568 0B73               CTL1    0,63,4,0
3569 0B74               CTL2    88,0,1
3570 0B75               ERR     8
3571 0B76       ST88:   CHK1    15,0          ; CLASS=IDENT ?
3572 0B77               CTL1    0,64,4,0
3573 0B78               CTL2    89,0,1
3574 0B79               ERR     8
3575 0B7A       ST89:   CHK1    11,0          ; CLASS=RPAREN ?
3576 0B7B               CTL1    0,0,0,0
3577 0B7C               CTL2    90,0,1
3578 0B7D               ERR     8
3579 0B7E       ST90:   CHK2    EQUAL,0       ; SYMB=EQUAL ?
3580 0B7F               CTL1    0,65,4,0      ;    YES - CHECK EXPRESSION
3581 0B80               CTL2    33,3,1
3582  ;
3583  ;       USER FUNCTION PROCESSING (FNX)
3584  ;
3585 0B81       ST91:   CHK3    17,0          ; CHECK MODE=8 (PASS2)
3586 0B82               CTL1    0,67,10,0
3587 0B83               CTL2    33,92,1       ;    YES - CONTINUE ON TO EXPRES
3588 0B84               CTL1    0,0,0,0
3589 0B85               CTL2    39,0,1        ;    NO - CONTINUE SCAN
3590 0B86       ST92:   CTL1    0,68,10,0     ; END FUNCTION PROCESSING
3591 0B87               CTL2    39,0,1
3592  ;
3593  ;       WAIT STATEMENT
3594  ;
3595 0B88       ST93:   CHK2    COMMA,0       ; SYMB=COMMA ?
3596 0B89               CTL1    0,24,10,0     ;    YES - GET 2ND EXPRESSION
3597 0B8A               CTL2    33,94,1
3598 0B8B               ERR     8
3599 0B8C       ST94:   CHK2    COMMA,0       ; SYMB=COMMA ?
3600 0B8D               CTL1    0,24,10,0     ;    YES - GET 3RD EXPRESSION
3601 0B8E               CTL2    33,95,1
3602 0B8F               CTL1    0,79,10,0     ; EXECUTE AFTER 2 ARGUMENTS
3603 0B90               CTL2    4,0,0
3604 0B91       ST95:   CTL1    0,78,10,0     ; EXECUTE AFTER 3 ARGUMENTS
3605 0B92               CTL2    4,0,0
3606  ;
3607  ;       DISK COMMANDS (LOAD,MERGE,SAVE)
3608  ;
3609 0B93       ST96:   CHK3    9,0           ; MUST BE MODE=2 ONLY
3610 0B94               CTL1    0,0,0,0
3611 0B95               CTL2    82,0,0
3612 0B96               ERR     4
3613  ;
3614  ;       DISK COMMANDS, EXEC STATEMENT
```

```
3615  ;
3616 0B97        ST82:   CHK1    22,0          ; CLASS=STRING-CONST ?
3617 0B98                CTL1    0,81,10,0     ;    PROCESS FILENAME
3618 0B99                CTL2    11,0,1        ;    EXECUTE COMMAND
3619 0B9A                ERR     8
3620  ;
3624  ;
```

```
3625                        .PAGE   'EXECUTION ROUTINES'
3626                        .LOCAL
3627  ;
3628  ;         GOTO/GOSUB/RETURN ROUTINES
3629  ;
3630 0B9B C040 A XQT1:    LD      AC0,NEXTAD    ; GOSUB - PUSH NEXTAD
3631 0B9C 5305 A          LI      AC3,5
3632 0B9D 949B*A          JSR     PUSH
3633 0B9E 1901 A          JMP     .+2
3634 0B9F 8000 A          RTS     0
3635 0BA0 5012 A          LI      AC0,18        ; GOSUBS NESTED TOO DEEP
3636 0BA1 98C5*A          JMP     RUNERR
3637  ;
3638 0BA2 5305 A XQT2:    LI      AC3,5         ; RETURN - PULL NEXTAD
3639 0BA3 949C*A          JSR     PULL
3640 0BA4 1902 A          JMP     .+3
3641 0BA5 D040 A          ST      AC0,NEXTAD
3642 0BA6 8000 A          RTS     0
3643 0BA7 5010 A          LI      AC0,16        ; RETURN FINDS NO ADDRESS
3644 0BA8 98C5*A          JMP     RUNERR
3645  ;
3646 0BA9 C443 A XQT3:    LD      AC1,SYMA      ; GOTO/GOSUB - FIND STMT#
3647 0BAA 94AF*A          JSR     EFIND
3648 0BAB 1902 A          JMP     .+3
3649 0BAC D840 A          ST      AC2,NEXTAD
3650 0BAD 8000 A          RTS     0
3651 0BAE 500E A NOGO:    LI      AC0,14        ; STATEMENT NOT FOUND
3652 0BAF 98C5*A          JMP     RUNERR
3653  ;
3654  ;         STOP/END ROUTINE
3655  ;
3656 0BB0 5008 A XQT4:    LI      AC0,8         ; CHECK EDIT OR PASS2 MODE
3657 0BB1 F013 A          SKNE    AC0,MODE
3658 0BB2 1907 A          JMP     $1
3659 0BB3 5002 A          LI      AC0,2         ; EDIT MODE (END STMT ONLY):
3660 0BB4 F01A A          SKNE    AC0,UNIT
3661 0BB5 8000 A          RTS     0             ; IGNORE IF DISK INPUT
3662 0BB6 5000 A          LI      AC0,0         ; RESET INPUT UNIT
3663 0BB7 D01A A          ST      AC0,UNIT
3664 0BB8 D016 A          ST      AC0,TXTMOD    ; RETURN TO COMMAND MODE
3665 0BB9 8000 A          RTS     0
3666 0BBA C042 A $1:      LD      AC0,SYMB      ; PASS 2: CHECK IF STOP OR END
3667 0BBB F129*A          SKNE    AC0,=143
3668 0BBC 9895*A          JMP     BEGIN         ; END: GO TO BEGIN ('READY')
3669 0BBD 6400 A          PULL    AC0           ; STOP: PULL STACK FROM JSR TO H
3670 0BBE 8001 A          RTS     1             ; RETURN FROM  JSR SYNTAX
3671  ;
3672  ;         COMMAND PROCESSING ROUTINES
3673  ;
3674 0BBF C042 A XQT5:    LD      AC0,SYMB      ; SAVE COMMAND ADDRESS
3675 0BC0 D064 A          ST      AC0,TMPADR
3676 0BC1 5001 A          LI      AC0,1         ; INITIALIZE COMMAND OPERANDS
3677 0BC2 D062 A          ST      AC0,LIST1
3678 0BC3 C122*A          LD      AC0,=09999
```

```
3679 0BC4 D063 A          ST      AC0,LIST2
3680 0BC5 5000 A          LI      AC0,0          ; OP3 = NUMBER OF OPERANDS
3681 0BC6 D060 A          ST      AC0,OP3
3682 0BC7 8000 A          RTS     0
3683  ;
3684 0BC8 C043 A XQT6:    LD      AC0,SYMA       ; SAVE 1ST COMMAND OPERAND
3685 0BC9 D062 A          ST      AC0,LIST1
3686 0BCA 5001 A          LI      AC0,1
3687 0BCB D060 A          ST      AC0,OP3
3688 0BCC 8000 A          RTS     0
3689  ;
3690 0BCD C043 A XQT7:    LD      AC0,SYMA       ; SAVE 2ND COMMAND OPERAND
3691 0BCE D063 A          ST      AC0,LIST2
3692 0BCF 5002 A          LI      AC0,2
3693 0BD0 D060 A          ST      AC0,OP3
3694 0BD1 8000 A          RTS     0
3695  ;
3696 0BD2 C864 A XQT8:    LD      AC2,TMPADR     ; FETCH COMMAND ROUTINE ADDRESS
3697 0BD3 E913*A          ADD     AC2,=CMDTAB-201
3698 0BD4 9A00 A          JMP     @(AC2)         ; THEN EXECUTE IT
3699  ;
3700  ;        SPECIAL CHECKS
3701  ;
3702 0BD5 5002 A XQT9:    LI      AC0,2          ; CHECK IF MODE=2 (DIRECT EXEC)
3703 0BD6 1901 A          JMP     .+2
3704 0BD7 5008 A XQT17:   LI      AC0,8          ; CHECK IF MODE=8 (EXEC - PASS 2
3705 0BD8 F013 A          SKNE    AC0,MODE
3706 0BD9 8000 A          RTS     0
3707 0BDA 8001 A          RTS     1
3708  ;
3709 0BDB C041 A XQT16:   LD      AC0,CLASS      ; CHECK IF ITEM IS A STRING
3710 0BDC 9D0B*A          SKG     AC0,=19        ; ALLOWABLE CLASS VALUES ARE:
3711 0BDD 9D0B*A          SKG     AC0,=16        ;    17 - SIMPLE STRING
3712 0BDE 1901 A          JMP     .+2            ;    18 - STRING ARRAY
3713 0BDF 8000 A          RTS     0              ;    19 - SUBSTRING
3714 0BE0 F109*A          SKNE    AC0,=22        ;    22 - STRING CONSTANT
3715 0BE1 8000 A          RTS     0              ;    23 - STRING FUNCTION
3716 0BE2 F108*A          SKNE    AC0,=23
3717 0BE3 8000 A          RTS     0
3718 0BE4 8001 A          RTS     1
3719 0BE5 008F A          .POOL   7
     0BE6 9999 A
     0BE7 1795 T
     0BE8 0013 A
     0BE9 0010 A
     0BEA 0016 A
     0BEB 0017 A
3720  ;
3721  ;        PRINT ROUTINES
3722  ;
3723 0BEC 503B A XQT11:   LI      AC0,';'/256    ; PRINT DELIMITER -
3724 0BED F042 A          SKNE    AC0,SYMB       ;    RETURN IF SEMI-COLON
3725 0BEE 8000 A          RTS     0
3726 0BEF 500F A          LI      AC0,15         ; CALCULATE TAB FOR NEXT ZONE
```

```
3727 0BF0 9C1D A ZONE:   SKG     AC0,OUTCOL
3728 0BF1 1906 A         JMP     NZ1
3729 0BF2 F01E A         SKNE    AC0,LINLEN    ; AT END OF LINE-
3730 0BF3 989F*A         JMP     CRLF          ;    GO TO NEXT LINE
3731 0BF4 D05C A         ST      AC0,OP1
3732 0BF5 9C1E A         SKG     AC0,LINLEN    ; TAB TO COLUMN IF NOT
3733 0BF6 1903 A         JMP     NZ2           ;    PAST END OF LINE
3734 0BF7 989F*A         JMP     CRLF
3735 0BF8 780E A NZ1:    AISZ    AC0,14        ; ZONES ARE 14 COLUMNS WIDE
3736 0BF9 19F6 A         JMP     ZONE
3737 0BFA 5D00 A NZ2:    RCPY    AC0,AC1       ; BE SURE NEW ZONE IS WIDE ENOUG
3738 0BFB 7001 A         CAI     AC0,1
3739 0BFC E01E A         ADD     AC0,LINLEN
3740 0BFD 9D4D*A         SKG     AC0,=12       ; CHECK # SPACES - 1
3741 0BFE 989F*A         JMP     CRLF          ;    SMALL ZONE - GO TO NEXT LIN
3742 0BFF 5C40 A         RCPY    AC1,AC0
3743 0C00 192A A         JMP     TAB           ;    TAB TO ZONE
3744  ;
3745 0C01 94AB*A XQT13:  JSR     STROP         ; PRINT A STRING
3746 0C02 C600 A         LD      AC1,(AC2)
3747 0C03 7900 A         AISZ    AC1,0         ; CHECK FOR STRING LENGTH=0
3748 0C04 1901 A         JMP     .+2
3749 0C05 8000 A         RTS     0
3750 0C06 C01D A         LD      AC0,OUTCOL    ; IF STRING WILL GO PAST LINE EN
3751 0C07 6840 A         RADD    AC1,AC0       ;    THEN START A NEW LINE
3752 0C08 9C1E A         SKG     AC0,LINLEN
3753 0C09 1901 A         JMP     .+2
3754 0C0A 949F*A         JSR     CRLF
3755 0C0B 7A01 A         AISZ    AC2,1         ; SKIP OVER STRING LENGTH
3756 0C0C C200 A $PLOOP: LD      AC0,(AC2)
3757 0C0D 2410 A         ROR     AC0,8,0
3758 0C0E 949E*A         JSR     OUT1
3759 0C0F 79FF A         AISZ    AC1,-1
3760 0C10 1901 A         JMP     .+2
3761 0C11 8000 A         RTS     0
3762 0C12 2410 A         ROR     AC0,8,0
3763 0C13 949E*A         JSR     OUT1
3764 0C14 79FF A         AISZ    AC1,-1
3765 0C15 1901 A         JMP     .+2
3766 0C16 8000 A         RTS     0
3767 0C17 7A01 A         AISZ    AC2,1
3768 0C18 19F3 A         JMP     $PLOOP
3769  ;
3770  ;        PRINT FUNCTIONS:  TAB, SPC, LIN
3771  ;
3772 0C19 94A8*A XQT14:  JSR     CALCUL        ; GET OPERAND
3773 0C1A 94B5*A         JSR     ROUND
3774 0C1B 5C00 A         NOP
3775 0C1C D05C A         ST      AC0,OP1
3776 0C1D 94AA*A         JSR     GETFCN        ; PULL FUNCTION FLAG
3777 0C1E 5D00 A         RCPY    AC0,AC1
3778 0C1F C05C A         LD      AC0,OP1
3779 0C20 F52B*A         SKNE    AC1,=189
3780 0C21 1913 A         JMP     SPCFCN
```

```
3781 0C22 F52A*A          SKNE    AC1,=190
3782 0C23 191C A          JMP     LINFCN
3783 0C24 9CDD*A TABFCN:  SKG     AC0,=1          ; TAB: CRLF IF (X <= 1)
3784 0C25 989F*A          JMP     CRLF
3785 0C26 9C1E A.$T1:     SKG     AC0,LINLEN      ; REDUCE ARGUMENT TO RANGE
3786 0C27 1903 A          JMP     TAB             ;    1 <= X <= LINLEN
3787 0C28 3780 A          SFLG    CARRY
3788 0C29 901E A          SUBB    AC0,LINLEN
3789 0C2A 19FB A          JMP     $T1
3790 0C2B 5D00 A TAB:     RCPY    AC0,AC1         ; (AC0) IS TAB COLUMN
3791 0C2C F01D A          SKNE    AC0,OUTCOL      ; THERE NOW ?
3792 0C2D 8000 A          RTS     0               ;    YES - QUIT
3793 0C2E 9C1D A          SKG     AC0,OUTCOL      ; BEYOND ?
3794 0C2F 949F*A          JSR     CRLF            ;    YES - TAB ON NEXT LINE
3795 0C30 5020 A          LI      AC0,BLANK       ; SEND BLANKS UNTIL REACH TABSTO
3796 0C31 949E*A $T2:     JSR     OUT1
3797 0C32 F41D A          SKNE    AC1,OUTCOL
3798 0C33 8000 A          RTS     0
3799 0C34 19FC A          JMP     $T2
3800 0C35 4109 A SPCFCN:  BOC     ZR0,$RTN        ; SEND SPACES
3801 0C36 4B08 A          BOC     NEG,$RTN
3802 0C37 E01D A          ADD     AC0,OUTCOL
3803 0C38 9C1E A          SKG     AC0,LINLEN
3804 0C39 1901 A          JMP     .+2
3805 0C3A 989F*A          JMP     CRLF            ; JUST GO TO NEXT LINE IF TOO MA
3806 0C3B 5020 A          LI      AC0,BLANK
3807 0C3C 949E*A $SEND:   JSR     OUT1            ; SEND (OP1) CHARACTERS
3808 0C3D AC5C A          DSZ     OP1
3809 0C3E 19FD A          JMP     $SEND
3810 0C3F 8000 A $RTN:    RTS     0
3811 0C40 41FE A LINFCN:  BOC     ZR0,$RTN        ; SEND LINE FEEDS
3812 0C41 4BFD A          BOC     NEG,$RTN
3813 0C42 500A A          LI      AC0,0A
3814 0C43 19F8 A          JMP     $SEND
3815  ;
3816 0C44 94A8*A XQT15:   JSR     CALCUL          ; PRINT EXPRESSION VALUE
3817 0C45 5E00 A          RCPY    AC0,AC2
3818 0C46 C01F A          LD      AC0,LZONE       ; CHECK IF NEW LINE NEEDED
3819 0C47 9C1D A          SKG     AC0,OUTCOL
3820 0C48 949F*A          JSR     CRLF
3821 0C49 5C80 A          RCPY    AC2,AC0
3822 0C4A 98BA*A          JMP     FLDEC           ; PRINT THE STRING
3823 0C4B 000C A          .POOL   3
     0C4C 00BD A
     0C4D 00BE A
3824  ;
3825  ;         LET STATEMENT
3826  ;
3827 0C4E 94A8*A XQT18:   JSR     CALCUL          ; GET VALUE AND STORE IT
3828 0C4F D065 A          ST      AC0,TEMP
3829 0C50 D466 A          ST      AC1,TEMP+1
3830 0C51 94A9*A          JSR     GETVAL          ; PULL DESTINATION ADDRESS
3831 0C52 7A00 A          AISZ    AC2,0           ; CHECK IT
3832 0C53 1901 A          JMP     .+2
```

```
3833 0C54 94C8*A          JSR     OPERR
3834 0C55 949C*A          JSR     PULL          ; MAKE SURE STACK IS NOW EMPTY
3835 0C56 1901 A          JMP     .+2
3836 0C57 94C8*A          JSR     OPERR
3837 0C58 C065 A          LD      AC0,TEMP
3838 0C59 C466 A          LD      AC1,TEMP+1
3839 0C5A D200 A          ST      AC0,0(AC2)
3840 0C5B D601 A          ST      AC1,1(AC2)
3841 0C5C 8000 A          RTS     0
3842  ;
3843  ;        STRING ASSIGNMENT PROCESSING
3844  ;
3845                      .LOCAL
3846 0C5D 94AB*A XQT19:   JSR     STROP         ; GET STRING TO STORE
3847 0C5E C8A7*A          LD      AC2,=STR3     ; TEMPORARILY COPY TO STR3
3848 0C5F 5F00 A          RCPY    AC0,AC3       ; AC3 = SOURCE
3849 0C60 992C*A          JMP     COPYS         ; COPY STRING
3850  ;
3851 0C61 94AB*A XQT20:   JSR     STROP         ; GET STRING TO CONCATENATE
3852 0C62 C600 A          LD      AC1,0(AC2)
3853 0C63 D465 A          ST      AC1,TEMP      ; TEMP = # CHAR TO ADD
3854 0C64 5100 A          LI      AC1,0         ; AC1 = CHAR NO. OF ADDED STRING
3855 0C65 CCA7*A          LD      AC3,=STR3
3856 0C66 7A01 A          AISZ    AC2,1         ; UPDATE POINTERS TO SKIP LENGTH
3857 0C67 7B01 A          AISZ    AC3,1
3858 0C68 DC57 A          ST      AC3,SSTART    ; SSTART,SCOL = DESTINATION INFO
3859 0C69 C3FF A          LD      AC0,-1(AC3)
3860 0C6A D058 A          ST      AC0,SCOL
3861 0C6B C05A A          LD      AC0,NSTRC
3862 0C6C D059 A          ST      AC0,MAXSTR
3863 0C6D F465 A $CAT:    SKNE    AC1,TEMP      ; CHECK IF DONE
3864 0C6E 1904 A          JMP     $RTN
3865 0C6F 9497*A          JSR     GETS          ; GET CHAR (INCREMENTS AC1)
3866 0C70 9498*A          JSR     PUTS          ; STORE CHAR
3867 0C71 5C00 A          NOP
3868 0C72 19FA A          JMP     $CAT
3869 0C73 C857 A $RTN:    LD      AC2,SSTART    ; UPDATE STRING LENGTH
3870 0C74 C058 A          LD      AC0,SCOL
3871 0C75 D2FF A          ST      AC0,-1(AC2)
3872 0C76 8000 A          RTS     0
3873  ;
3874 0C77 94AB*A XQT12:   JSR     STROP         ; GET STRING DESTINATION
3875 0C78 5302 A          LI      AC3,2
3876 0C79 949C*A          JSR     PULL          ; CHECK TO BE SURE STACK IS EMPT
3877 0C7A 1901 A          JMP     .+2
3878 0C7B 1564 A          JSR     OPERR
3879 0C7C CCA7*A          LD      AC3,=STR3
3880 0C7D 990F*A          JMP     COPYS
3881  ;
3882  ;        ARRAY SUBSCRIPT PROCESSING
3883  ;
3884 0C7E 5102 A XQT21:   LI      AC1,2         ; SAVE ARRAY ID ON OPERAND STACK
3885 0C7F 1901 A          JMP     .+2
3886 0C80 5106 A XQT76:   LI      AC1,6         ; SAVE STRING ARRAY ID ON OPERAN
```

```
3887 0C81 5302 A          LI      AC3,2
3888 0C82 C043 A          LD      AC0,SYMA
3889 0C83 949B*A          JSR     PUSH
3890 0C84 192A A          JMP     EXPERR
3891 0C85 5C40 A          RCPY    AC1,AC0       ; TYPE IS 2 (ARRAY) OR 6 (STRING
3892 0C86 1930 A          JMP     EPUSH2
3893  ;
3894 0C87 94AC*A XQT22:   JSR     RPAREN        ; COMMA IN SUBSCRIPT: TREAT AS
3895                                            ;    RPAREN THEN LPAREN
3896 0C88 5303 A LPAREN:  LI      AC3,3         ; LEFT PAREN - PUT ONTO OPERATOR
3897 0C89 5000 A          LI      AC0,0         ;    STACK (ZERO PRECEDENCE)
3898 0C8A 949B*A          JSR     PUSH
3899 0C8B 1923 A          JMP     EXPERR
3900 0C8C 8000 A          RTS     0
3901 0C8D 0E7D T          .POOL   1
3902  ;
3903  ;        EXPRESSION PROCESSING
3904  ;
3905                      .LOCAL
3906 0C8E 5302 A XQT26:   LI      AC3,2         ; PUT IDENTIFIER ONTO OPERAND ST
3907 0C8F C043 A          LD      AC0,SYMA      ; CONVERT TO DATA ADDRESS
3908 0C90 7802 A          AISZ    AC0,2
3909 0C91 949B*A          JSR     PUSH
3910 0C92 191C A          JMP     EXPERR
3911 0C93 5001 A          LI      AC0,1         ; FLAG=1 (ADDRESS)
3912 0C94 1922 A          JMP     EPUSH2
3913  ;
3914 0C95 5302 A XQT27:   LI      AC3,2         ; PUT CONSTANT ONTO OPERAND STAC
3915 0C96 C047 A          LD      AC0,VALUE
3916 0C97 949B*A          JSR     PUSH
3917 0C98 1916 A          JMP     EXPERR
3918 0C99 C048 A          LD      AC0,VALUE+1
3919 0C9A 949B*A          JSR     PUSH
3920 0C9B 1913 A          JMP     EXPERR
3921 0C9C 5000 A          LI      AC0,0         ; FLAG=0 (CONSTANT)
3922 0C9D 1919 A          JMP     EPUSH2
3923  ;
3924 0C9E C042 A XQT60:   LD      AC0,SYMB      ; PUSH OPERATOR UNCONDITIONALLY
3925 0C9F C444 A          LD      AC1,PREC
3926 0CA0 1909 A          JMP     OPPUSH
3927 0CA1 502B A XQT28:   LI      AC0,'+'/256   ; UNARY OP: IF IT IS +
3928 0CA2 F042 A          SKNE    AC0,SYMB      ;    THEN IGNORE IT
3929 0CA3 8000 A          RTS     0
3930 0CA4 C444 A          LD      AC1,PREC
3931 0CA5 502D A          LI      AC0,'-'/256   ; IF OP IS -, THEN ADJUST
3932 0CA6 F042 A          SKNE    AC0,SYMB      ;    ITS PRECEDENCE
3933 0CA7 510C A          LI      AC1,12
3934 0CA8 C042 A          LD      AC0,SYMB      ; IDENTIFY OP AS UNARY (SET BIT
3935 0CA9 A57A*A          OR      AC0,=0100
3936 0CAA 5303 A OPPUSH:  LI      AC3,3         ; PUSH ONTO OPERATOR STACK
3937 0CAB 949B*A          JSR     PUSH
3938 0CAC 1902 A          JMP     EXPERR
3939 0CAD 5C40 A          RCPY    AC1,AC0       ; PUSH PRECEDENCE ONTO STACK
3940 0CAE 1909 A          JMP     EPUSH3
```

```
3941 0CAF 5012 A EXPERR: LI      AC0,18        ; EXPRESSION STACK OVERFLOW
3942 0CB0 98C6*A         JMP     SYNERR
3943  ;
3944 0CB1 15D6 A XQT29:  JSR     LPAREN
3945 0CB2 C042 A         LD      AC0,SYMB      ; PUSH BUILT-IN-FUNCTION ADDR
3946 0CB3 5302 A         LI      AC3,2         ;    ONTO OPERAND STACK
3947 0CB4 949B*A         JSR     PUSH
3948 0CB5 19F9 A         JMP     EXPERR
3949 0CB6 5003 A         LI      AC0,3
3950 0CB7 5302 A EPUSH2: LI      AC3,2         ; PUSH (AC0) ONTO OPERAND STACK
3951 0CB8 949B*A EPUSH3: JSR     PUSH
3952 0CB9 19F5 A         JMP     EXPERR
3953 0CBA 8000 A         RTS     0
3954  ;
3955 0CBB 15CC A XQT30:  JSR     LPAREN
3956 0CBC C043 A         LD      AC0,SYMA      ; PUSH USER-FUNCTION POINTER
3957 0CBD 5302 A         LI      AC3,2         ;    ONTO OPERAND STACK
3958 0CBE 949B*A         JSR     PUSH
3959 0CBF 19EF A         JMP     EXPERR
3960 0CC0 5E00 A         RCPY    AC0,AC2       ; CHECK TO BE SURE FUNCTION IS D
3961 0CC1 C601 A         LD      AC1,1(AC2)
3962 0CC2 7900 A         AISZ    AC1,0
3963 0CC3 94AF*A         JSR     EFIND
3964 0CC4 1902 A         JMP     FNERR         ; ERROR IF NOT FOUND OR STMT=0
3965 0CC5 5004 A         LI      AC0,4
3966 0CC6 19F0 A         JMP     EPUSH2
3967  ;
3968 0CC7 5022 A FNERR:  LI      AC0,34        ; USER FUNCTION NOT DEFINED
3969 0CC8 98C6*A         JMP     SYNERR
3970  ;
3971 0CC9 94A9*A GETSUB: JSR     GETVAL        ; GET ARRAY SUBSCRIPT VALUE(S)
3972 0CCA 94B5*A         JSR     ROUND
3973 0CCB 5C00 A         NOP
3974 0CCC D05C A         ST      AC0,OP1
3975 0CCD 5000 A         LI      AC0,0
3976 0CCE D05E A         ST      AC0,OP2
3977 0CCF 5302 A         LI      AC3,2
3978 0CD0 949C*A         JSR     PULL
3979 0CD1 150E A         JSR     OPERR
3980 0CD2 B952*A         SKAZ    AC0,=0FFFE    ; CHECK TYPE OF NEXT ITEM
3981 0CD3 190B A         JMP     $RTN          ;    NOT A NUMBER
3982 0CD4 949B*A         JSR     PUSH          ; PUT TYPE FLAG BACK FOR GETVAL
3983 0CD5 94C9*A         JSR     SYSERR
3984 0CD6 94A9*A         JSR     GETVAL
3985 0CD7 94B5*A         JSR     ROUND
3986 0CD8 5C00 A         NOP
3987 0CD9 C45C A         LD      AC1,OP1       ; STORE DOUBLE SUBSCRIPTS
3988 0CDA D05C A         ST      AC0,OP1
3989 0CDB D45E A         ST      AC1,OP2
3990 0CDC 5302 A         LI      AC3,2
3991 0CDD 949C*A         JSR     PULL          ; PULL ARRAY BASE FLAG
3992 0CDE 1501 A         JSR     OPERR
3993 0CDF 8000 A $RTN:   RTS     0
3994 0CE0 94A1*A OPERR:  JSR     MESG          ; MISSING OR IMPROPER OPERAND ON
```

```
3995 0CE1 18B8 T          .WORD   OPERM
3996 0CE2 9943*A          JMP     SYSER2
3997  ;
3998 0CE3 15E5 A GETARY:  JSR     GETSUB      ; CALCULATE ARRAY ITEM ADDRESS
3999 0CE4 78FE A          AISZ    AC0,-2      ; CHECK TO BE SURE ITEM IS ARRAY
4000 0CE5 15FA A          JSR     OPERR
4001 0CE6 5002 A          LI      AC0,2       ; 2 WORDS PER ENTRY
4002 0CE7 150B A          JSR     CLCSUB      ; CALCULATE ADDRESS
4003 0CE8 5001 A          LI      AC0,1       ; FLAG=1 (ADDRESS)
4004 0CE9 1906 A          JMP     $PUT
4005  ;
4006 0CEA 15DE A XQT73:   JSR     GETSUB      ; CALCULATE STRING ARRAY ADDRESS
4007 0CEB 78FA A          AISZ    AC0,-6      ; CHECK TO BE SURE ITEM IS STRIN
4008 0CEC 15F3 A          JSR     OPERR
4009 0CED C05B A          LD      AC0,NSTRW   ; NO. WORDS PER STRING ITEM
4010 0CEE 1504 A          JSR     CLCSUB      ; CALCULATE ADDRESS
4011 0CEF 5005 A          LI      AC0,5       ; FLAG=5 (STRING)
4012 0CF0 949B*A $PUT:    JSR     PUSH        ; PUSH TYPE FLAG
4013 0CF1 94C9*A          JSR     SYSERR
4014 0CF2 8000 A          RTS     0
4015  ;
4016 0CF3 D04A A CLCSUB:  ST      AC0,DIGIT   ; SAVE # WORDS PER ENTRY
4017 0CF4 949C*A          JSR     PULL        ; PULL BASE ADDRESS
4018 0CF5 15EA A          JSR     OPERR
4019 0CF6 5E00 A          RCPY    AC0,AC2
4020 0CF7 C05C A          LD      AC0,OP1     ; CHECK VALIDITY OF SUBSCRIPTS
4021 0CF8 4104 A          BOC     ZR0,$V2
4022 0CF9 4B16 A          BOC     NEG,SUBERR  ; SUBSCRIPTS MUST BE >= 0
4023 0CFA 9E01 A          SKG     AC0,1(AC2)  ;    AND <= DIMENSION
4024 0CFB 1901 A          JMP     .+2
4025 0CFC 1913 A          JMP     SUBERR
4026 0CFD C05E A $V2:     LD      AC0,OP2
4027 0CFE 4104 A          BOC     ZR0,$CE
4028 0CFF 4B10 A          BOC     NEG,SUBERR
4029 0D00 9E02 A          SKG     AC0,2(AC2)
4030 0D01 1901 A          JMP     .+2
4031 0D02 190D A          JMP     SUBERR
4032 0D03 C05C A $CE:     LD      AC0,OP1     ; CALCULATE ARRAY ELEMENT ADDRES
4033 0D04 C602 A          LD      AC1,2(AC2)  ; ADDR = (((SUB1 * (DIM2+1))
4034 0D05 7901 A          AISZ    AC1,1
4035 0D06 94B8*A          JSR     MULT
4036 0D07 E45E A          ADD     AC1,OP2     ;         + SUB2)
4037 0D08 C04A A          LD      AC0,DIGIT   ;         * WORDS-PER-ENTRY )
4038 0D09 94B8*A          JSR     MULT
4039 0D0A E603 A          ADD     AC1,3(AC2)  ;         + BASE
4040 0D0B 5C40 A          RCPY    AC1,AC0
4041 0D0C 5302 A          LI      AC3,2       ; PUT NEW ADDRESS ONTO OPERAND S
4042 0D0D 949B*A          JSR     PUSH
4043 0D0E 1572 A          JSR     SYSERR
4044 0D0F 8000 A          RTS     0
4045 0D10 5016 A SUBERR:  LI      AC0,22      ; SUBSCRIPT OUT OF RANGE
4046 0D11 98C6*A          JMP     SYNERR
4047  ;
4048 0D12 15B6 A XQT75:   JSR     GETSUB      ; SUBSTRING: FIX UP ITEM ON STAC
```

```
4049 0D13 78FB A          AISZ    AC0,-5        ; CHECK TO BE SURE STACK HAS STR
4050 0D14 15CB A          JSR     OPERR
4051 0D15 C05C A          LD      AC0,OP1       ; CHECK 2 NUMERIC OPERANDS
4052 0D16 B8DA*A          SKAZ    AC0,=0FF00
4053 0D17 19F8 A          JMP     SUBERR        ;    ERROR IF <0 OR >255
4054 0D18 2810 A          SHL     AC0,8,0
4055 0D19 5D00 A          RCPY    AC0,AC1
4056 0D1A C05E A          LD      AC0,OP2       ; CHECK 2ND NUMBER
4057 0D1B B8DA*A          SKAZ    AC0,=0FF00
4058 0D1C 19F3 A          JMP     SUBERR
4059 0D1D 5840 A          RXOR    AC1,AC0       ; MERGE 2 VALUES INTO BYTES OF A
4060 0D1E 949B*A          JSR     PUSH          ; PUSH ONTO STACK
4061 0D1F 1561 A          JSR     SYSERR
4062 0D20 5007 A          LI      AC0,7         ; SUBSTRING ITEM IS TYPE 7
4063 0D21 949B*A          JSR     PUSH
4064 0D22 155E A          JSR     SYSERR
4065 0D23 8000 A          RTS     0
4066  ;
4067 0D24 0100 A          .POOL   3
     0D25 FFFE A
     0D26 0D83 T
4068  ;
4069 0D27 9563*A XQT71:   JSR     RND           ; RND - SPECIAL IDENTIFIER
4070 0D28 1910 A          JMP     RESULT        ;    (WITHOUT ARGUMENT)
4071  ;
4072 0D29 1567 A XQT32:   JSR     CALCUL        ; EXECUTE BUILT-IN-FUNCTION
4073 0D2A D05C A          ST      AC0,OP1       ; SAVE OPERAND
4074 0D2B D45D A          ST      AC1,OP1+1
4075 0D2C 94AA*A          JSR     GETFCN        ; PULL B. I. F.  NUMBER
4076 0D2D 5E00 A          RCPY    AC0,AC2
4077 0D2E CD5D*A          LD      AC3,=TAB4     ; DETERMINE FUNCTION TO CALL
4078 0D2F FB00 A $F1:     SKNE    AC2,0(AC3)    ; COMPARE TO TABLE ENTRY
4079 0D30 1904 A          JMP     $F2
4080 0D31 FD5B*A          SKNE    AC3,=TAB4L
4081 0D32 15AD A          JSR     OPERR         ; NOT FOUND
4082 0D33 7B02 A          AISZ    AC3,2
4083 0D34 19FA A          JMP     $F1
4084 0D35 CB01 A $F2:     LD      AC2,1(AC3)    ; LOAD ROUTINE ADDR
4085 0D36 C05C A          LD      AC0,OP1       ; OPERAND IN AC0,AC1
4086 0D37 C45D A          LD      AC1,OP1+1
4087 0D38 1600 A          JSR     (AC2)         ; CALL FUNCTION ROUTINE
4088 0D39 5302 A RESULT:  LI      AC3,2         ; PUSH RESULT (AC0,AC1) ONTO
4089 0D3A 949B*A          JSR     PUSH          ;    OPERAND STACK
4090 0D3B 98C7*A          JMP     EXPERR
4091 0D3C 5C40 A          RCPY    AC1,AC0
4092 0D3D 949B*A          JSR     PUSH
4093 0D3E 98C7*A          JMP     EXPERR
4094 0D3F 5000 A          LI      AC0,0
4095 0D40 949B*A          JSR     PUSH
4096 0D41 98C7*A          JMP     EXPERR
4097 0D42 8000 A          RTS     0
4098  ;
4099 0D43 50FF A RPAREN:  LI      AC0,-1        ; PROCESS RIGHT PARENTHESIS
4100 0D44 1901 A          JMP     .+2
```

```
4101 0D45 C044 A XQT33:  LD      AC0,PREC      ; PROCESS BINARY OPERATOR
4102 0D46 D061 A         ST      AC0,G         ;    G = PRECEDENCE OF INCOMING
4103 0D47 5303 A OPER:   LI      AC3,3         ; PULL OPERATOR ON TOP OF STACK
4104 0D48 949C*A         JSR     PULL          ;    F = PRECEDENCE OF OPERATOR
4105 0D49 1915 A         JMP     $EMP          ; STACK EMPTY
4106 0D4A C461 A         LD      AC1,G         ; *** FOR DEBUG ****************
4107 0D4B 5C00 A         NOP                   ; *** FOR DEBUG ****************
4108 0D4C 4201 A         BOC     POS,.+2       ; *** MAKE SURE OPERATOR FROM ST
4109 0D4D 1592 A         JSR     OPERR         ; ***    HAS POSITIVE PRECEDENCE
4110 0D4E 410E A         BOC     ZRO,$LP       ; LEFT PAREN
4111 0D4F F061 A         SKNE    AC0,G
4112 0D50 1915 A         JMP     $OP2
4113 0D51 9C61 A         SKG     AC0,G
4114 0D52 1901 A         JMP     .+2
4115 0D53 1912 A         JMP     $OP2
4116 0D54 949B*A         JSR     PUSH          ; F<G - RESTORE F TO STACK
4117 0D55 152B A         JSR     SYSERR
4118 0D56 C042 A $OP1:   LD      AC0,SYMB      ; PUSH OPERATOR ONTO STACK
4119 0D57 949B*A         JSR     PUSH
4120 0D58 98C7*A         JMP     EXPERR
4121 0D59 C061 A         LD      AC0,G
4122 0D5A 949B*A         JSR     PUSH
4123 0D5B 98C7*A         JMP     EXPERR
4124 0D5C 8000 A         RTS     0
4125 0D5D 949B*A $LP:    JSR     PUSH          ; PUSH LEFT PAREN BACK
4126 0D5E 1522 A         JSR     SYSERR
4127 0D5F 50FF A $EMP:   LI      AC0,-1        ; LPAREN OR STACK EMPTY:
4128 0D60 F061 A         SKNE    AC0,G         ;    IF RPAREN, PULL LPAREN,
4129 0D61 1901 A         JMP     .+2           ;    OTHERWISE PUSH OPERATOR
4130 0D62 19F3 A         JMP     $OP1
4131 0D63 949C*A         JSR     PULL          ; PULL LPAREN IF ANY
4132 0D64 5C00 A         NOP                   ;    (OR STACK EMPTY)
4133 0D65 8000 A         RTS     0
4134 0D66 949C*A $OP2:   JSR     PULL          ; F>=G - OPERATE!
4135 0D67 1519 A         JSR     SYSERR        ;    (BINARY OR UNARY OP)
4136 0D68 D060 A         ST      AC0,OP3       ; SAVE OPERATOR
4137 0D69 B9BA*A         SKAZ    AC0,=0100     ; CHECK FOR UNARY OP
4138 0D6A 1903 A         JMP     $OP3
4139 0D6B 1527 A         JSR     GETVAL
4140 0D6C D05E A         ST      AC0,OP2
4141 0D6D D45F A         ST      AC1,OP2+1
4142 0D6E 1524 A $OP3:   JSR     GETVAL
4143 0D6F D05C A         ST      AC0,OP1
4144 0D70 D45D A         ST      AC1,OP1+1
4145 0D71 CD1C*A         LD      AC3,=TAB3
4146 0D72 1501 A         JSR     XOPER         ; CALL OPERATOR ROUTINE
4147 0D73 19D3 A         JMP     OPER          ; THEN TEST NEXT OPERATOR FROM S
4148   :
4149 0D74 C860 A XOPER:  LD      AC2,OP3       ; DETERMINE ROUTINE TO CALL
4150 0D75 FB00 A $F3:    SKNE    AC2,0(AC3)    ; COMPARE TO TABLE ENTRY
4151 0D76 1904 A         JMP     $F4
4152 0D77 FD17*A         SKNE    AC3,=TAB3L
4153 0D78 1508 A         JSR     SYSERR        ; NOT FOUND
4154 0D79 7B02 A         AISZ    AC3,2
```

```
4155 0D7A 19FA A          JMP     $F3
4156 0D7B CF01 A $F4:     LD      AC3,1(AC3)    ; LOAD ROUTINE ADDR (OP1 STILL I
4157 0D7C DC60 A          ST      AC3,OP3
4158 0D7D C85E A          LD      AC2,OP2       ; LOAD OP2 INTO AC2,AC3
4159 0D7E CC5F A          LD      AC3,OP2+1
4160 0D7F 9460 A          JSR     @OP3          ; GO TO THE ROUTINE
4161 0D80 19B8 A          JMP     RESULT        ; PUT RESULT ONTO STACK
4162  ;
4163 0D81 94A1*A SYSERR:  JSR     MESG          ; SYSTEM ERROR
4164 0D82 18AC T          .WORD   SYSERM
4165 0D83 94A1*A SYSER2:  JSR     MESG          ; PRINT ERROR LOCATION
4166 0D84 18B1 T          .WORD   ATLOC
4167 0D85 1E00 A          XCHRS   AC2
4168 0D86 949D*A          JSR     PUTW
4169 0D87 1E00 A          XCHRS   AC2
4170 0D88 949F*A          JSR     CRLF
4171 0D89 0000 A          HALT
4172 0D8A 9895*A          JMP     BEGIN
4173 0D8B 12D9 T          .POOL   6
     0D8C 183E T
     0D8D 185C T
     0D8E 1818 T
     0D8F 183C T
     0D90 0000 A
4174  ;
4175  ;
4176  ;         GET EXPRESSION VALUE FROM STACK
4177  ;
4178  ;         CALCUL - PERFORM ANY OPERATIONS REMAINING
4179  ;         GETVAL - GET ONLY LAST OPERAND
4180  ;
4181 0D91 15B1 A CALCUL:  JSR     RPAREN        ; MAKE SURE EXPRESSION IS COMPLE
4182 0D92 5C00 A          NOP                   ; *** FOR DEBUG ****************
4183 0D93 5302 A GETVAL:  LI      AC3,2         ; GET VALUE FROM OPERAND STACK
4184 0D94 949C*A          JSR     PULL
4185 0D95 94C8*A          JSR     OPERR
4186 0D96 4108 A          BOC     ZR0,$0
4187 0D97 B98D*A          SKAZ    AC0,=0FFFE    ; SHOULD BE ONLY CONSTANT OR SIM
4188 0D98 94C8*A          JSR     OPERR         ;    AT THIS POINT
4189 0D99 949C*A          JSR     PULL
4190 0D9A 94C8*A          JSR     OPERR
4191 0D9B 5E00 A          RCPY    AC0,AC2       ; COPY ADDRESS INTO AC2
4192 0D9C C200 A          LD      AC0,0(AC2)    ; LOAD VALUE INTO AC0,AC1
4193 0D9D C601 A          LD      AC1,1(AC2)
4194 0D9E 8000 A          RTS     0
4195 0D9F 5200 A $0:      LI      AC2,0         ; CONSTANT: SET AC2=0 (NO ADDRES
4196 0DA0 949C*A          JSR     PULL          ; PULL CONSTANT FROM STACK
4197 0DA1 94C8*A          JSR     OPERR
4198 0DA2 5D00 A          RCPY    AC0,AC1
4199 0DA3 949C*A          JSR     PULL
4200 0DA4 94C8*A          JSR     OPERR
4201 0DA5 8000 A          RTS     0
4202  ;
4203  ;
```

```
4204  ;        CHECK STACKTOP=ST39 (OPERATOR)
4205  ;
4206                          .LOCAL
4207 0DA6 9499*A XQT34:   JSR     SAVE          ; SAVE REG
4208 0DA7 5301 A          LI      AC3,1
4209 0DA8 949C*A          JSR     PULL          ; PULL ADDR FROM SYNTAX STACK
4210 0DA9 1904 A          JMP     $NO           ;    NOTHING ON STACK
4211 0DAA 949B*A          JSR     PUSH          ; PUT IT BACK
4212 0DAB 15D5 A          JSR     SYSERR        ; SOMETHING'S WRONG?
4213 0DAC F1E3*A          SKNE    AC0,=ST39
4214 0DAD 989A*A          JMP     RESTOR        ; MATCH: RESTORE & RTS 0
4215 0DAE 949A*A $NO:     JSR     RESTOR        ; NO MATCH
4216 0DAF 8001 A          RTS     1
4217  ;
4218  ;        DIM ROUTINES
4219  ;
4220 0DB0 C843 A XQT35:   LD      AC2,SYMA      ; SAVE SYMBOL TABLE ADDR
4221 0DB1 D864 A          ST      AC2,TMPADR    ;    OF ARRAY NAME
4222 0DB2 C201 A          LD      AC0,1(AC2)    ; CHECK IF ALREADY DEFINED
4223 0DB3 4501 A          BOC     NZR0,.+2
4224 0DB4 8000 A          RTS     0
4225 0DB5 5018 A          LI      AC0,24        ; ARRAY PREVIOUSLY DIMENSIONED
4226 0DB6 98C6*A          JMP     SYNERR
4227  ;
4228 0DB7 1507 A XQT36:   JSR     $DIM          ; DIMENSION #1 OF ARRAY
4229 0DB8 C864 A          LD      AC2,TMPADR
4230 0DB9 D201 A          ST      AC0,1(AC2)
4231 0DBA 8000 A          RTS     0
4232 0DBB 1503 A XQT37:   JSR     $DIM          ; DIMENSION #2 OF ARRAY
4233 0DBC C864 A          LD      AC2,TMPADR
4234 0DBD D202 A          ST      AC0,2(AC2)
4235 0DBE 8000 A          RTS     0
4236 0DBF C047 A $DIM:    LD      AC0,VALUE     ; CHECK CONSTANT DIMENSION
4237 0DC0 C448 A          LD      AC1,VALUE+1
4238 0DC1 94B5*A          JSR     ROUND
4239 0DC2 5C00 A          NOP
4240 0DC3 4101 A          BOC     ZR0,.+2
4241 0DC4 8000 A          RTS     0
4242 0DC5 5016 A          LI      AC0,22        ; ZERO SUBSCRIPT NOT ALLOWED
4243 0DC6 98C6*A          JMP     SYNERR
4244  ;
4245  ;        IF STATEMENT ROUTINE
4246  ;
4247 0DC7 15C9 A XQT38:   JSR     CALCUL        ; GET VALUE FROM STACK
4248 0DC8 4101 A          BOC     ZR0,.+2
4249 0DC9 8000 A          RTS     0             ; IF VALUE <> 0, CONTINUE PROCES
4250 0DCA 6400 A          PULL    AC0           ; IF VALUE = 0, GO ON TO
4251 0DCB 8002 A          RTS     2             ;    THE NEXT LINE
4252  ;
4253  ;        FOR/NEXT ROUTINES
4254  ;
4255 0DCC 5008 A XQT39:   LI      AC0,8         ; FOR STATEMENT:
4256 0DCD F013 A          SKNE    AC0,MODE      ; CHECK IF PASS1 OR PASS2
4257 0DCE 190E A          JMP     $P2
```

```
4258 0DCF 5306 A          LI      AC3,6
4259 0DD0 C043 A          LD      AC0,SYMA      ; PUT ID ADDRESS ON STACK
4260 0DD1 949B*A          JSR     PUSH
4261 0DD2 1908 A          JMP     $OVER
4262 0DD3 50FF A FORPTR:  LI      AC0,-1        ; GET SYMBOL TABLE ENTRY
4263 0DD4 D052 A          ST      AC0,FNLINE    ; - 'DUMMY PARAM' TO HOLD ADDR
4264 0DD5 9575*A          JSR     LOOKUP
4265 0DD6 9895*A          JMP     BEGIN
4266 0DD7 C843 A          LD      AC2,SYMA
4267 0DD8 5000 A          LI      AC0,0
4268 0DD9 D052 A          ST      AC0,FNLINE
4269 0DDA 8000 A          RTS     0
4270 0DDB 5012 A $OVER:   LI      AC0,18        ; FOR-LOOP NEST IS TOO DEEP
4271 0DDC 98C6*A          JMP     SYNERR
4272 0DDD C043 A $P2:     LD      AC0,SYMA      ; PASS 2
4273 0DDE D064 A          ST      AC0,TMPADR    ; TMPADR = FOR-VARIABLE PTR
4274 0DDF 15F3 A FORSAV:  JSR     FORPTR        ; CHECK SAVE AREA ADDR
4275 0DE0 CA03 A          LD      AC2,3(AC2)
4276 0DE1 7A00 A          AISZ    AC2,0         ; IS IT ALLOCATED YET?
4277 0DE2 1904 A          JMP     $FSV          ;    YES
4278 0DE3 5005 A          LI      AC0,5         ;    NO - GET 5 WORDS
4279 0DE4 9567*A          JSR     ALOC
4280 0DE5 CC43 A          LD      AC3,SYMA
4281 0DE6 DB03 A          ST      AC2,3(AC3)    ; SAVE AREA ADDRESS
4282 0DE7 D865 A $FSV:    ST      AC2,TEMP      ; TEMP = AREA ADDR
4283 0DE8 8000 A          RTS     0
4284  ;
4285 0DE9 15A7 A XQT40:   JSR     CALCUL        ; SAVE INITIAL VALUE
4286 0DEA C864 A          LD      AC2,TMPADR
4287 0DEB D202 A          ST      AC0,2(AC2)
4288 0DEC D603 A          ST      AC1,3(AC2)
4289 0DED 8000 A          RTS     0
4290 0DEE 15A2 A XQT41:   JSR     CALCUL        ; SAVE END VALUE
4291 0DEF C865 A          LD      AC2,TEMP
4292 0DF0 D200 A          ST      AC0,0(AC2)
4293 0DF1 D601 A          ST      AC1,1(AC2)
4294 0DF2 8000 A $RTS:    RTS     0
4295 0DF3 15FA A XQT42:   JSR     XQT41         ; NO STEP: SAVE END VALUE
4296 0DF4 C0DC A          LD      AC0,FL1       ; STEP=1
4297 0DF5 C4DD A          LD      AC1,FL1+1
4298 0DF6 1901 A          JMP     .+2
4299 0DF7 1599 A XQT43:   JSR     CALCUL        ; GET STEP VALUE
4300 0DF8 CC65 A          LD      AC3,TEMP
4301 0DF9 D302 A          ST      AC0,2(AC3)
4302 0DFA D703 A          ST      AC1,3(AC3)
4303 0DFB C040 A          LD      AC0,NEXTAD    ; COPY ADDRESS OF NEXT LINE
4304 0DFC D304 A          ST      AC0,4(AC3)
4305 0DFD C864 A          LD      AC2,TMPADR
4306 0DFE C202 A          LD      AC0,2(AC2)    ; LOAD INITIAL VALUE
4307 0DFF C603 A          LD      AC1,3(AC2)
4308 0E00 CB00 A          LD      AC2,0(AC3)    ; LOAD END VALUE
4309 0E01 CF01 A          LD      AC3,1(AC3)
4310 0E02 94BB*A          JSR     FLCMP         ; COMPARE START & END VALUES
4311 0E03 5D00 A          RCPY    AC0,AC1       ;    SAVE IN AC1 TEMPORARILY
```

```
4312 0E04 C865 A          LD      AC2,TEMP      ; CHECK SIGN OF STEP
4313 0E05 C202 A          LD      AC0,2(AC2)
4314 0E06 4201 A          BOC     POS,.+2
4315 0E07 7101 A          CAI     AC1,1         ; NEG STEP: INVERT THE TEST
4316 0E08 5C40 A          RCPY    AC1,AC0
4317 0E09 41E8 A          BOC     ZR0,$RTS      ; START <= END -- DO THE LOOP
4318 0E0A 4BE7 A          BOC     NEG,$RTS
4319 0E0B C840 A SKIPFOR: LD      AC2,NEXTAD    ; START > END -- SKIP OVER FOR-L
4320 0E0C D83E A          ST      AC2,LSTART
4321 0E0D 94B0*A          JSR     FINDCR
4322 0E0E DC40 A          ST      AC3,NEXTAD
4323 0E0F 5002 A          LI      AC0,2
4324 0E10 D03F A          ST      AC0,COL
4325 0E11 94A3*A          JSR     SCAN          ; SCAN 1ST ITEM IN LINE
4326 0E12 94C9*A          JSR     SYSERR
4327 0E13 C042 A          LD      AC0,SYMB      ; IS IT 'NEXT'?
4328 0E14 F138*A          SKNE    AC0,=139
4329 0E15 1901 A          JMP     .+2           ;    YES
4330 0E16 19F4 A          JMP     SKIPFOR       ;    NO
4331 0E17 94A3*A          JSR     SCAN          ; SCAN IDENTIFIER
4332 0E18 94C9*A          JSR     SYSERR
4333 0E19 C043 A          LD      AC0,SYMA      ; COMPARE ADDRESS TO FOR-VARIABL
4334 0E1A F064 A          SKNE    AC0,TMPADR
4335 0E1B 1901 A          JMP     .+2           ;    SAME
4336 0E1C 19EE A          JMP     SKIPFOR       ;    NOT SAME
4337 0E1D 6400 A          PULL    AC0           ; EXIT SYNTAX ANALYZER -
4338 0E1E 8002 A          RTS     2             ;    GO ON TO LINE AFTER 'NEXT'
4339  ;
4340 0E1F 5008 A XQT44:   LI      AC0,8         ; NEXT STATEMENT:
4341 0E20 F013 A          SKNE    AC0,MODE      ; CHECK PASS1 OR PASS2
4342 0E21 1907 A          JMP     $2
4343 0E22 5306 A          LI      AC3,6
4344 0E23 949C*A          JSR     PULL          ; PULL ID ADDRESS
4345 0E24 1902 A          JMP     $NERR
4346 0E25 F043 A          SKNE    AC0,SYMA      ; COMPARE TO NEXT IDENT
4347 0E26 8000 A          RTS     0
4348 0E27 501A A $NERR:   LI      AC0,26        ; IDENTIFIER DOES NOT MATCH
4349 0E28 98C6*A          JMP     SYNERR
4350 0E29 C043 A $2:      LD      AC0,SYMA      ; NEXT - PASS2: SAVE ID PTR
4351 0E2A D064 A          ST      AC0,TMPADR
4352 0E2B 15B3 A          JSR     FORSAV        ; GET ADDR OF SAVE AREA
4353 0E2C C204 A          LD      AC0,4(AC2)    ; CHECK FOR LEGAL NEXT ADDR
4354 0E2D 41F9 A          BOC     ZR0,$NERR     ; NONE - NO FOR WAS EXECUTED
4355 0E2E CC64 A          LD      AC3,TMPADR
4356 0E2F C202 A          LD      AC0,2(AC2)    ; ADD STEP TO VARIABLE
4357 0E30 C603 A          LD      AC1,3(AC2)
4358 0E31 CB02 A          LD      AC2,2(AC3)
4359 0E32 CF03 A          LD      AC3,3(AC3)
4360 0E33 94BC*A          JSR     FLADD
4361 0E34 CC64 A          LD      AC3,TMPADR
4362 0E35 D302 A          ST      AC0,2(AC3)    ; SAVE NEW VALUE
4363 0E36 D703 A          ST      AC1,3(AC3)
4364 0E37 CC65 A          LD      AC3,TEMP      ; COMPARE TO END VALUE
4365 0E38 CB00 A          LD      AC2,0(AC3)
```

```
4366 0E39 CF01 A          LD      AC3,1(AC3)
4367 0E3A 94BB*A          JSR     FLCMP
4368 0E3B 5D00 A          RCPY    AC0,AC1
4369 0E3C C865 A          LD      AC2,TEMP        ; CHECK SIGN OF STEP
4370 0E3D C202 A          LD      AC0,2(AC2)
4371 0E3E 4201 A          BOC     POS,.+2
4372 0E3F 7101 A          CAI     AC1,1
4373 0E40 5C40 A          RCPY    AC1,AC0
4374 0E41 4105 A          BOC     ZRO,$NXLIN
4375 0E42 4B04 A          BOC     NEG,$NXLIN
4376 0E43 C865 A          LD      AC2,TEMP        ; LOOP OVER - CLEAR NEXT ADDR,
4377 0E44 5000 A          LI      AC0,0           ;    SO THAT A JUMP INTO MIDDLE
4378 0E45 D204 A          ST      AC0,4(AC2)      ;    OF LOOP WILL BE AN ERROR
4379 0E46 8000 A          RTS     0               ; GO ON TO FOLLOWING STMT
4380 0E47 C865 A $NXLIN:  LD      AC2,TEMP        ; GET ADDRESS OF START OF LOOP
4381 0E48 C204 A          LD      AC0,4(AC2)
4382 0E49 D040 A          ST      AC0,NEXTAD      ; SAVE NEXTAD THEN CONTINUE ON T
4383 0E4A 8000 A          RTS     0
4384 0E4B 089D T          .POOL   4
     0E4C 025C T
     0E4D 008B A
     0E4E 0000 A
4385  ;
4386  ;        SAVE/RESTORE SCANNER POINTERS
4387  ;
4388 0E4F C03E A SSCAN:   LD      AC0,LSTART      ; SAVE SCANNER POINTERS
4389 0E50 D03C A          ST      AC0,SVSCAN
4390 0E51 C03F A          LD      AC0,COL
4391 0E52 D03D A          ST      AC0,SVSCAN+1
4392 0E53 8000 A          RTS     0
4393 0E54 C03C A RSCAN:   LD      AC0,SVSCAN      ; RESTORE SCANNER POINTERS
4394 0E55 D03E A          ST      AC0,LSTART
4395 0E56 C03D A          LD      AC0,SVSCAN+1
4396 0E57 D03F A          ST      AC0,COL
4397 0E58 8000 A          RTS     0
4398  ;
4399  ;        INPUT/READ ROUTINES
4400  ;
4401                      .LOCAL
4402 0E59 50FF A XQT48:   LI      AC0,-1          ; INPUT:
4403 0E5A D066 A          ST      AC0,TEMP+1      ; TEMP+1 = -1 (FLAG FOR NEW PROM
4404 0E5B 5000 A          LI      AC0,0           ; OUTDVC = 0 (TTY)
4405 0E5C D01C A          ST      AC0,OUTDVC
4406 0E5D 5001 A          LI      AC0,1           ; TEMP   = 1  (FLAG FOR INPUT)
4407 0E5E 1901 A          JMP     .+2
4408 0E5F 5000 A XQT51:   LI      AC0,0           ; READ:  TEMP=0 (FLAG FOR READ)
4409 0E60 D065 A          ST      AC0,TEMP
4410 0E61 8000 A          RTS     0
4411  ;
4412 0E62 C843 A XQT49:   LD      AC2,SYMA        ; INPUT/READ - SIMPLE IDENTIFIER
4413 0E63 7A02 A          AISZ    AC2,2           ; ADDRESS IN AC2
4414 0E64 1902 A          JMP     $IN
4415 0E65 95E8*A XQT50:   JSR     GETARY          ; GET ARRAY ADDR INTO TMPADR
4416 0E66 94A9*A          JSR     GETVAL
```

```
4417 0E67 D864 A $IN:    ST    AC2,TMPADR
4418 0E68 5000 A         LI    AC0,0          ; SET FLAG FOR NUMERIC INPUT
4419 0E69 D069 A         ST    AC0,TYPFLG
4420 0E6A 1904 A         JMP   $IN2
4421 0E6B 5001 A·XQT70:  LI    AC0,1          ; SET FLAG FOR STRING INPUT
4422 0E6C D069 A         ST    AC0,TYPFLG
4423 0E6D 94AB*A         JSR   STROP          ; COPY STRING ADDRESS
4424 0E6E D864 A         ST    AC2,TMPADR
4425 0E6F C065 A $IN2:   LD    AC0,TEMP       ; TEST WHETHER INPUT OR READ
4426 0E70 4514 A         BOC   NZRO,INPUT
4427 0F71 C069 A         LD    AC0,TYPFLG
4428 0E72 4507 A         BOC   NZRO,$STRD
4429 0E73 1543 A         JSR   NDATA          ; READ NUMERIC DATA
4430 0E74 C864 A $CPYN:  LD    AC2,TMPADR     ; COPY NUMBER TO RESULT
4431 0E75 C047 A         LD    AC0,VALUE
4432 0E76 C448 A         LD    AC1,VALUE+1
4433 0E77 D200 A         ST    AC0,0(AC2)
4434 0E78 D601 A         ST    AC1,1(AC2)
4435 0E79 8000 A         RTS   0
4436 0E7A 1545 A $STRD:  JSR   SDATA          ; READ STRING DATA
4437 0E7B C864 A $CPYS:  LD    AC2,TMPADR
4438 0E7C CC43 A         LD    AC3,SYMA
4439 0E7D C45B A COPYS:  LD    AC1,NSTRW      ; COPY STRING
4440 0E7E C300 A         LD    AC0,(AC3)      ;    AC3 POINTS TO SOURCE STRING
4441 0E7F D200 A         ST    AC0,(AC2)      ;    AC2 POINTS TO DESTINATION
4442 0E80 7B01 A         AISZ  AC3,1
4443 0E81 7A01 A         AISZ  AC2,1
4444 0E82 79FF A         AISZ  AC1,-1
4445 0E83 19FA A         JMP   COPYS+1
4446 0E84 8000 A         RTS   0
4447 0E85 15C9 A INPUT:  JSR   SSCAN          ; INPUT FROM KEYBOARD
4448 0E86 C0B1*A         LD    AC0,=INBUF     ; SET SCANNER TO PROCESS INPUT
4449 0E87 D03E A         ST    AC0,LSTART
4450 0E88 C066 A         LD    AC0,TEMP+1     ; CHECK COLUMN FLAG TO DETERMINE
4451 0E89 4206 A         BOC   POS,$2         ;    WHETHER TO PROMPT FOR NEW L
4452 0E8A 5000 A $1:     LI    AC0,0          ; READ NEW LINE
4453 0E8B D066 A         ST    AC0,TEMP+1
4454 0E8C 94AD*A         JSR   RDLINE
4455 0E8D 19FE A         JMP   .-1
4456 0E8E 1923 A         JMP   $STOP
4457 0E8F 94AE*A         JSR   PACK
4458 0E90 C069 A $2:     LD    AC0,TYPFLG     ; SET STYPE TO 0 IF NUMERIC
4459 0E91 D03B A         ST    AC0,STYPE      ;    INPUT, 1 IF STRING INPUT
4460 0E92 C066 A         LD    AC0,TEMP+1     ; SCAN FOR INPUT ITEM
4461 0E93 D03F A         ST    AC0,COL
4462 0E94 4504 A         BOC   NZRO,$2A       ; CHECK IF 1ST CALL ON LINE
4463 0E95 155E A         JSR   DSCAN1         ; SCAN DATA
4464 0E96 190D A         JMP   $INERR
4465 0E97 1917 A         JMP   $MORE
4466 0E98 1903 A         JMP   $2B
4467 0E99 155C A $2A:    JSR   DSCAN2         ; SCAN DATA
4468 0E9A 1909 A         JMP   $INERR
4469 0E9B 1913 A         JMP   $MORE
4470 0E9C C03F A $2B:    LD    AC0,COL        ; SAVE PTR FOR NEXT
```

```
4471 0E9D D066 A          ST      AC0,TEMP+1
4472 0E9E 15B5 A          JSR     RSCAN           ; RESTORE SCANNER POINTERS
4473 0E9F C069 A          LD      AC0,TYPFLG      ; VERIFY TYPE
4474 0EA0 450A A          BOC     NZRO,$SIN
4475 0EA1 5015 A          LI      AC0,21          ; SHOULD BE NUMERIC
4476 0EA2 F041 A          SKNE    AC0,CLASS
4477 0EA3 19D0 A          JMP     $CPYN
4478 0EA4 94A1*A $INERR:  JSR     MESG
4479 0EA5 18A0 T          .WORD   INERR
4480 0EA6 C03C A          LD      AC0,SVSCAN      ; SET POINTER FOR RESCAN OF
4481 0EA7 D040 A          ST      AC0,NEXTAD      ;    INPUT COMMAND
4482 0EA8 6400 A          PULL    AC0
4483 0EA9 C850 A          LD      AC2,SYNRTN      ; EXIT SYNTAX ANALYZER TO
4484 0EAA 1A02 A          JMP     2(AC2)          ;    RESCAN INPUT
4485 0EAB 5016 A $SIN:    LI      AC0,22          ; SHOULD BE STRING
4486 0EAC F041 A          SKNE    AC0,CLASS
4487 0EAD 19CD A          JMP     $CPYS
4488 0EAE 19F5 A          JMP     $INERR
4489 0EAF 94A1*A $MORE:   JSR     MESG            ; READ MORE INPUT
4490 0EB0 18AB T          .WORD   MOREIN
4491 0EB1 19D8 A          JMP     $1
4492 0EB2 C03C A $STOP:   LD      AC0,SVSCAN      ; EXECUTION STOPPED - SAVE
4493 0EB3 D040 A          ST      AC0,NEXTAD      ;    RESTART ADDRESS
4494 0EB4 6400 A          PULL    AC0
4495 0EB5 C850 A          LD      AC2,SYNRTN      ; GO TO STOP RETURN LOCATION
4496 0EB6 1A01 A          JMP     1(AC2)
4497  ;
4498  ;
4499  ;        READ NEXT NUMERIC DATA
4500  ;
4501 0EB7 5000 A NDATA:   LI      AC0,0
4502 0EB8 D069 A          ST      AC0,TYPFLG
4503 0EB9 150D A          JSR     GTDATA
4504 0EBA 5015 A          LI      AC0,21          ; CHECK IF NUMERIC
4505 0EBB F041 A          SKNE    AC0,CLASS
4506 0EBC 8000 A          RTS
4507 0EBD 1596 A $TERR:   JSR     RSCAN           ; TYPE ERROR
4508 0EBE 5014 A          LI      AC0,20
4509 0EBF 98C6*A          JMP     SYNERR
4510  ;
4511  ;        READ NEXT STRING DATA
4512  ;
4513 0EC0 5001 A SDATA:   LI      AC0,1
4514 0EC1 D069 A          ST      AC0,TYPFLG
4515 0EC2 1504 A          JSR     GTDATA          ; GET NEXT DATA
4516 0EC3 5016 A          LI      AC0,22          ; CHECK IF STRING
4517 0EC4 F041 A          SKNE    AC0,CLASS
4518 0EC5 8000 A          RTS
4519 0EC6 19F6 A          JMP     $TERR
4520  ;
4521  ;        GET NEXT DATA ITEM
4522  ;
4523 0EC7 9499*A GTDATA:  JSR     SAVE
4524 0EC8 1586 A          JSR     SSCAN
```

```
4525 0EC9 C053 A          LD      AC0,DSTART
4526 0ECA D03E A          ST      AC0,LSTART
4527 0ECB C054 A          LD      AC0,DPTR
4528 0ECC D03F A          ST      AC0,COL
4529 0ECD 4512 A          BOC     NZRO,$G4
4530 0ECE C83E A $G1:     LD      AC2,LSTART    ; FIND NEXT DATA STMT
4531 0ECF F835 A          SKNE    AC2,EEND
4532 0ED0 191B A          JMP     $OUT          ;    NO MORE DATA
4533 0ED1 5002 A          LI      AC0,2
4534 0ED2 D03F A          ST      AC0,COL
4535 0ED3 5000 A          LI      AC0,0
4536 0ED4 D03B A          ST      AC0,STYPE
4537 0ED5 94A3*A          JSR     SCAN          ; CHECK IF 1ST ITEM ON LINE IS '
4538 0ED6 94C9*A          JSR     SYSERR
4539 0ED7 C042 A          LD      AC0,SYMB
4540 0ED8 F116*A          SKNE    AC0,=144
4541 0ED9 1904 A          JMP     $G3
4542 0EDA C83E A $G2:     LD      AC2,LSTART    ; NOT DATA STMT - GO ON TO NEXT
4543 0EDB 94B0*A          JSR     FINDCR
4544 0EDC DC3E A          ST      AC3,LSTART
4545 0EDD 19F0 A          JMP     $G1
4546 0EDE CD11*A $G3:     LD      AC3,=DSCAN1   ; AC3 = SCAN ROUTINE ADDRESS
4547 0EDF 1901 A          JMP     .+2
4548 0EE0 CD10*A $G4:     LD      AC3,=DSCAN2
4549 0EE1 C069 A          LD      AC0,TYPFLG
4550 0EE2 D03B A          ST      AC0,STYPE
4551 0EE3 1700 A          JSR     (AC3)         ; SCAN DATA STATEMENT FOR NEXT I
4552 0EE4 19D8 A          JMP     $TERR
4553 0EE5 19F4 A          JMP     $G2
4554 0EE6 C03E A          LD      AC0,LSTART    ; UPDATE POINTERS
4555 0EE7 D053 A          ST      AC0,DSTART
4556 0EE8 C03F A          LD      AC0,COL
4557 0EE9 D054 A          ST      AC0,DPTR
4558 0EEA 9507*A          JSR     RSCAN
4559 0EEB 989A*A          JMP     RESTOR
4560  ;
4561 0EEC 9505*A $OUT:    JSR     RSCAN         ; OUT OF DATA
4562 0EED 501E A          LI      AC0,30
4563 0EEE 98C5*A          JMP     RUNERR
4564  ;
4565 0EEF 0090 A          .POOL   5
     0EF0 0EF4 T
     0EF1 0EF6 T
     0EF2 0E54 T
     0EF3 0000 A
4566  ;
4567  ;
4568  ;        SCAN DATA LINE FOR NEXT ITEM
4569  ;
4570  ;        DSCAN1 - 1ST CALL ON LINE: NO COMMA 1ST
4571  ;        DSCAN2 - SHOULD HAVE COMMA OR CR FIRST
4572  ;
4573  ;                RTS 0 - ERROR
```

```
4575  ;                RTS 2 - NORMAL RETURN (STRING OR NUMERIC)
4576  ;
4577 0EF4 5000 A DSCAN1: LI      ACO,0          ; SET COMMA FLAG
4578 0EF5 1901 A         JMP     .+2
4579 0EF6 5001 A.DSCAN2: LI      ACO,1
4580 0EF7 D067 A         ST      ACO,DTEMP
4581 0EF8 5000 A         LI      ACO,0          ; INITIALIZE SIGN FLAG
4582 0EF9 D068 A         ST      ACO,DSIGN
4583 0EFA D03A A         ST      ACO,NTYPE
4584 0EFB 94A3*A         JSR     SCAN           ; SCAN ITEM
4585 0EFC 8000 A         RTS     0              ;    ERROR
4586 0EFD C067 A         LD      ACO,DTEMP
4587 0EFE 4109 A         BOC     ZRO,$DS2
4588 0EFF 502C A         LI      ACO,',/256
4589 0F00 F042 A         SKNE    ACO,SYMB
4590 0F01 1904 A         JMP     $DS1
4591 0F02 500D A         LI      ACO,13
4592 0F03 F041 A         SKNE    ACO,CLASS
4593 0F04 8001 A         RTS     1
4594 0F05 8000 A         RTS     0              ; SYNTAX ERROR
4595 0F06 94A3*A $DS1:   JSR     SCAN           ; SCAN FOR ITEM AFTER COMMA
4596 0F07 8000 A         RTS     0
4597 0F08 500D A $DS2:   LI      ACO,13         ; CHECK FOR CR
4598 0F09 F041 A         SKNE    ACO,CLASS
4599 0F0A 8001 A         RTS     1
4600 0F0B 5001 A         LI      ACO,1          ; CHECK FOR +/-
4601 0F0C F041 A         SKNE    ACO,CLASS
4602 0F0D 1901 A         JMP     .+2
4603 0F0E 1906 A         JMP     $DS3
4604 0F0F 502D A         LI      ACO,'-'/256    ; +/- : CHECK SIGN
4605 0F10 F042 A         SKNE    ACO,SYMB
4606 0F11 D068 A         ST      ACO,DSIGN      ; DSIGN NON-ZERO IF MINUS
4607 0F12 94A3*A         JSR     SCAN           ; SCAN PAST SIGN
4608 0F13 8000 A         RTS
4609 0F14 1903 A         JMP     $DS4
4610 0F15 5016 A $DS3:   LI      ACO,22         ; CHECK IF STRING OR NUMERIC
4611 0F16 F041 A         SKNE    ACO,CLASS
4612 0F17 190B A         JMP     $DS5           ;    STRING - DONE
4613 0F18 5015 A $DS4:   LI      ACO,21
4614 0F19 F041 A         SKNE    ACO,CLASS
4615 0F1A 1901 A         JMP     .+2            ;    NUMERIC
4616 0F1B 8000 A         RTS     0              ;    NEITHER - ERROR
4617 0F1C C068 A         LD      ACO,DSIGN      ; CHECK SIGN OF NUMBER
4618 0F1D 4105 A         BOC     ZRO,$DS5       ;    0 - POSITIVE
4619 0F1E C047 A         LD      ACO,VALUE      ; NEGATE IF - PRECEDED
4620 0F1F C448 A         LD      AC1,VALUE+1
4621 0F20 94BF*A         JSR     NEGATE
4622 0F21 D047 A         ST      ACO,VALUE
4623 0F22 D448 A         ST      AC1,VALUE+1
4624 0F23 5000 A $DS5:   LI      ACO,0          ; RESET STYPE FOR NORMAL SCAN
4625 0F24 D03B A         ST      ACO,STYPE
4626 0F25 8002 A         RTS     2              ;    THEN RETURN
4627  ;
4628  ;        RANDOMIZE STATEMENT ROUTINE
```

```
 4629  ;
 4630 0F26 5300 A XQT53:  LI      AC3,0          ; ADD UP ALL OF MEMORY
 4631 0F27 E300 A         ADD     AC0,(AC3)      ;    FROM 0 TO LAST
 4632 0F28 7B01 A         AISZ    AC3,1
 4633 0F29 FC38 A         SKNE    AC3,LAST
 4634 0F2A 1901 A         JMP     .+2
 4635 0F2B 19FB A         JMP     XQT53+1
 4636 0F2C D086 A         ST      AC0,RNDNUM     ; STORE INTO RANDOM NUMBER
 4637 0F2D 8000 A         RTS     0
 4638  ;
 4639  ;        RESTORE STATEMENT ROUTINE
 4640  ;
 4641 0F2E C033 A RSDATA: LD      AC0,ESTART     ; RESTORE DATA POINTERS
 4642 0F2F D053 A         ST      AC0,DSTART
 4643 0F30 5000 A         LI      AC0,0
 4644 0F31 D054 A         ST      AC0,DPTR
 4645 0F32 8000 A         RTS     0
 4646  ;
 4647  ;        TIMED WAIT ROUTINE
 4648  ;
 4649                     .LOCAL
 4650 0F33 94A8*A XQT54:  JSR     CALCUL         ; GET DELAY TIME IN MILLISEC
 4651 0F34 94B5*A         JSR     ROUND
 4652 0F35 5C00 A         NOP
 4653 0F36 4106 A         BOC     ZR0,$RTN       ; RETURN IF TIME<=0
 4654 0F37 4B05 A         BOC     NEG,$RTN
 4655 0F38 515A A $W1:    LI      AC1,90         ; DELAY 1 MILLI-SECOND
 4656 0F39 79FF A         AISZ    AC1,-1
 4657 0F3A 19FE A         JMP     .-1
 4658 0F3B 78FF A         AISZ    AC0,-1         ; DECREMENT MILLISEC COUNTER
 4659 0F3C 19FB A         JMP     $W1            ; CONTINUE LOOP IF NOT DONE
 4660 0F3D 8000 A $RTN:   RTS     0
 4661  ;
 4662  ;        ON STATEMENT ROUTINES
 4663  ;
 4664 0F3E 94A8*A XQT55:  JSR     CALCUL         ; GET EXPRESSION INTEGER VALUE
 4665 0F3F 94B5*A         JSR     ROUND
 4666 0F40 5C00 A         NOP
 4667 0F41 4111 A         BOC     ZR0,XQT57      ; VALUE <= 0 -- ERROR
 4668 0F42 4B10 A         BOC     NEG,XQT57
 4669 0F43 D065 A         ST      AC0,TEMP
 4670 0F44 5000 A         LI      AC0,0          ; INITIALIZE COUNT
 4671 0F45 D066 A         ST      AC0,TEMP+1
 4672 0F46 8000 A         RTS     0
 4673  ;
 4674 0F47 C066 A XQT56:  LD      AC0,TEMP+1     ; INCREMENT STMT# COUNT
 4675 0F48 7801 A         AISZ    AC0,1
 4676 0F49 D066 A         ST      AC0,TEMP+1
 4677 0F4A F065 A         SKNE    AC0,TEMP       ; COMPARE COUNT TO VALUE
 4678 0F4B 1901 A         JMP     .+2            ;    EQUAL - PROCESS GOTO
 4679 0F4C 8000 A         RTS     0              ;    NOT EQUAL - GO ON TO NEXT S
 4680 0F4D C443 A         LD      AC1,SYMA       ; FIND STMT#
 4681 0F4E 94AF*A         JSR     EFIND
 4682 0F4F 99A3*A         JMP     NOGO
```

```
4683 0F50 D840 A          ST      AC2,NEXTAD
4684 0F51 6400 A          PULL    AC0
4685 0F52 8002 A          RTS     2
4686  ;
4687 0F53 5016 A XQT57:   LI      AC0,22        ; ON ARGUMENT OUT OF RANGE
4688 0F54 98C5*A          JMP     RUNERR
4689  ;
4690  ;        STRING PROCESSING ROUTINES
4691  ;
4692 0F55 C843 A XQT58:   LD      AC2,SYMA      ; STRING ID
4693 0F56 C203 A          LD      AC0,3(AC2)
4694 0F57 1901 A          JMP     $STK
4695 0F58 C043 A XQT59:   LD      AC0,SYMA      ; STRING CONSTANT
4696 0F59 5302 A $STK:    LI      AC3,2
4697 0F5A 949B*A          JSR     PUSH          ; PUSH ONTO OPERAND STACK
4698 0F5B 98C7*A          JMP     EXPERR
4699 0F5C 5005 A          LI      AC0,5         ; STRING ADDR IS TYPE 5
4700 0F5D 949B*A          JSR     PUSH
4701 0F5E 98C7*A          JMP     EXPERR
4702 0F5F 8000 A          RTS     0
4703 0F60 C857 A STKSTR:  LD      AC2,SSTART    ; TEMPORARY STRING: COPY
4704 0F61 C058 A          LD      AC0,SCOL      ;    LENGTH THEN PUT ON STACK
4705 0F62 7AFF A          AISZ    AC2,-1
4706 0F63 D200 A          ST      AC0,(AC2)
4707 0F64 5C80 A          RCPY    AC2,AC0
4708 0F65 19F3 A          JMP     $STK
4709  ;
4710  ;        STROP: GET STRING OPERAND
4711  ;
4712  ;        STRING ADDRESS IS RETURNED IN BOTH AC0 AND AC2
4713  ;
4714 0F66 5302 A STROP:   LI      AC3,2         ; PULL FROM OPERAND STACK
4715 0F67 949C*A          JSR     PULL
4716 0F68 94C9*A          JSR     SYSERR
4717 0F69 F147*A          SKNE    AC0,=7        ; CHECK IF SUBSTRING
4718 0F6A 1906 A          JMP     $7
4719 0F6B 78FB A          AISZ    AC0,-5        ; SHOULD BE STRING
4720 0F6C 94C8*A          JSR     OPERR
4721 0F6D 949C*A          JSR     PULL          ; PULL STRING ADDRESS
4722 0F6E 94C9*A          JSR     SYSERR
4723 0F6F 5E00 A          RCPY    AC0,AC2
4724 0F70 8000 A          RTS     0
4725 0F71 94A5*A $7:      JSR     STRTMP        ; ALLOCATE TEMP AREA
4726 0F72 949C*A          JSR     PULL          ; PULL SUBSTRING INFO
4727 0F73 94C9*A          JSR     SYSERR
4728 0F74 5D00 A          RCPY    AC0,AC1
4729 0F75 949C*A          JSR     PULL          ; PULL STRING ADDR
4730 0F76 94C9*A          JSR     SYSERR
4731 0F77 5E00 A          RCPY    AC0,AC2
4732 0F78 7A01 A          AISZ    AC2,1         ; AC2 = STRING START
4733 0F79 155C A          JSR     GETFCN        ; PULL SUBSTRING FUNCTION #
4734 0F7A CEFF A          LD      AC3,-1(AC2)   ; MAXSTR = STRING LENGTH
4735 0F7B DC59 A          ST      AC3,MAXSTR
4736 0F7C F135*A          SKNE    AC0,=191      ; CHECK SUBSTRING TYPE
```

```
4737 0F7D 1914 A          JMP     LEFFCN
4738 0F7E F134*A          SKNE    AC0,=192
4739 0F7F 1918 A          JMP     RIGFCN
4740 0F80 5C40 A MIDFCN:  RCPY    AC1,AC0       ; MID$ (STRING,FIRST,LENGTH)
4741 0F81 2D10 A          SHR     AC1,8,0       ; AC1 = START COLUMN (0 UP)
4742 0F82 4501 A          BOC     NZRO,.+2
4743 0F83 9930*A          JMP     SUBERR        ; ERROR IF START WITH CHAR 0
4744 0F84 79FF A          AISZ    AC1,-1
4745 0F85 5C00 A          NOP
4746 0F86 A8D5*A          AND     AC0,=0FF
4747 0F87 4501 A          BOC     NZRO,.+2
4748 0F88 507F A          LI      AC0,127       ; IF NO LENGTH SPEC, USE 127
4749 0F89 D059 A          ST      AC0,MAXSTR    ; SAVE LENGTH (TEMP)
4750 0F8A C2FF A          LD      AC0,-1(AC2)   ; CHECK SUBSTR LEGALITY
4751 0F8B 7001 A          CAI     AC0,1
4752 0F8C 6840 A          RADD    AC1,AC0
4753 0F8D 421D A          BOC     POS,$FIX      ; START > LENGTH -- RETURN NULL
4754 0F8E 7001 A          CAI     AC0,1         ; AC0 NOW HAS LENGTH OF REST OF
4755 0F8F 9C59 A          SKG     AC0,MAXSTR    ; LENGTH TO COPY MUST BE <= LENG
4756 0F90 D059 A          ST      AC0,MAXSTR    ;    OF REMAINDER OF STRING
4757 0F91 1910 A          JMP     $CPY
4758 0F92 5C40 A LEFFCN:  RCPY    AC1,AC0       ; LEFT$
4759 0F93 5100 A          LI      AC1,0         ; START WITH 1ST CHARACTER
4760 0F94 2C10 A          SHR     AC0,8,0       ; CHECK LENGTH <= LENGTH OF STRI
4761 0F95 9C59 A          SKG     AC0,MAXSTR
4762 0F96 D059 A          ST      AC0,MAXSTR    ; SAVE LENGTH IF <=
4763 0F97 190A A          JMP     $CPY
4764 0F98 2D10 A RIGFCN:  SHR     AC1,8,0       ; RIGHT$ - LAST CHAR = LAST OF S
4765 0F99 5C40 A          RCPY    AC1,AC0
4766 0F9A 9C59 A          SKG     AC0,MAXSTR    ; FIRST = - (MIN(LEN,MAX) - MAX)
4767 0F9B 1901 A          JMP     .+2
4768 0F9C C059 A          LD      AC0,MAXSTR
4769 0F9D 3780 A          SFLG    CARRY
4770 0F9E 9059 A          SUBB    AC0,MAXSTR
4771 0F9F 7001 A          CAI     AC0,1
4772 0FA0 6D00 A          RXCH    AC0,AC1
4773 0FA1 D059 A          ST      AC0,MAXSTR    ; SAVE SUBSTRING LENGTH
4774 0FA2 C059 A $CPY:    LD      AC0,MAXSTR    ; NULL STRING IF LENGTH=0
4775 0FA3 4107 A          BOC     ZRO,$FIX
4776 0FA4 9497*A $CPY2:   JSR     GETS          ; COPY AS MANY CHAR AS NEEDED
4777 0FA5 9498*A          JSR     PUTS          ;    TO TEMP STRING
4778 0FA6 5C00 A          NOP
4779 0FA7 C058 A          LD      AC0,SCOL
4780 0FA8 F059 A          SKNE    AC0,MAXSTR
4781 0FA9 1901 A          JMP     $FIX
4782 0FAA 19F9 A          JMP     $CPY2
4783 0FAB C857 A $FIX:    LD      AC2,SSTART    ; FIX UP TEMP STRING; RETURN ADD
4784 0FAC C058 A          LD      AC0,SCOL
4785 0FAD 7AFF A          AISZ    AC2,-1
4786 0FAE D200 A          ST      AC0,(AC2)
4787 0FAF 5C80 A          RCPY    AC2,AC0
4788 0FB0 8000 A          RTS     0
4789 0FB1 0007 A          .POOL   5
     0FB2 00BF A
```

```
     0FB3 00C0 A
     0FB4 0D10 T
     0FB5 0000 A
4790  ;
4791  ;        STRING COMPARISON
4792  ;
4793 0FB6 15AF A XQT61:   JSR     STROP         ; PULL 2ND ADDRESS
4794 0FB7 D05E A          ST      AC0,OP2
4795 0FB8 5303 A          LI      AC3,3         ; PULL RELATION
4796 0FB9 949C*A          JSR     PULL
4797 0FBA 94C9*A          JSR     SYSERR
4798 0FBB 949C*A          JSR     PULL
4799 0FBC 94C9*A          JSR     SYSERR
4800 0FBD D060 A          ST      AC0,OP3
4801 0FBE 15A7 A          JSR     STROP         ; PULL 1ST ADDRESS INTO AC2
4802 0FBF 7A01 A          AISZ    AC2,1
4803 0FC0 CC5E A          LD      AC3,OP2       ; RELOAD 2ND ADDRESS
4804 0FC1 7B01 A          AISZ    AC3,1
4805 0FC2 C45B A          LD      AC1,NSTRW     ; COMPARE STRINGS
4806 0FC3 79FF A          AISZ    AC1,-1
4807 0FC4 D45F A          ST      AC1,OP2+1
4808 0FC5 C200 A $CLOOP:  LD      AC0,(AC2)
4809 0FC6 C700 A          LD      AC1,(AC3)
4810 0FC7 94A4*A          JSR     SKL
4811 0FC8 1902 A          JMP     $GE
4812 0FC9 50FF A          LI      AC0,-1        ; FIRST < SECOND
4813 0FCA 1909 A          JMP     $SCMP
4814 0FCB 5840 A $GE:     RXOR    AC1,AC0       ; CHECK IF EQUAL
4815 0FCC 4506 A          BOC     NZRO,$GT
4816 0FCD 7B01 A          AISZ    AC3,1
4817 0FCE 7A01 A          AISZ    AC2,1
4818 0FCF AC5F A          DSZ     OP2+1         ; DECREMENT WORD COUNT
4819 0FD0 19F4 A          JMP     $CLOOP        ;    NOT DONE
4820 0FD1 5000 A          LI      AC0,0         ; FIRST = SECOND
4821 0FD2 1901 A          JMP     $SCMP
4822 0FD3 5001 A $GT:     LI      AC0,1         ; FIRST > SECOND
4823 0FD4 CDE0*A $SCMP:   LD      AC3,=TAB3A    ; COMPARE RESULT IN AC0 -
4824 0FD5 9933*A          JMP     XOPER         ;    EXEC OPERATOR & PUT
4825                                            ;    RESULT ONTO STACK
4826  ;
4827 0FD6 5302 A GETFCN:  LI      AC3,2         ; PULL FUNCTION POINTER
4828 0FD7 949C*A          JSR     PULL
4829 0FD8 94C9*A          JSR     SYSERR
4830 0FD9 78FD A          AISZ    AC0,-3        ; SHOULD BE TYPE 3 (FUNCTION)
4831 0FDA 94C8*A          JSR     OPERR
4832 0FDB 949C*A          JSR     PULL          ; PULL FUNCTION ID INTO AC0
4833 0FDC 94C9*A          JSR     SYSERR
4834 0FDD 8000 A          RTS     0
4835  ;
4836  ;        STRING->NUMERIC FUNCTIONS: LEN, ASC, VAL
4837  ;
4838 0FDE 5303 A XQT62:   LI      AC3,3         ; PULL LPAREN OFF OF OPERATOR ST
4839 0FDF 949C*A          JSR     PULL
4840 0FE0 94C8*A          JSR     OPERR
```

```
4841 0FE1 4101 A          BOC     ZR0,.+2
4842 0FE2 94C8*A          JSR     OPERR
4843 0FE3 1582 A          JSR     STROP          ; GET STRING OPERAND
4844 0FE4 D05C A          ST      AC0,OP1
4845 0FE5 15F0 A          JSR     GETFCN         ; GET FUNCTION NUMBER
4846 0FE6 C85C A          LD      AC2,OP1        ; RELOAD STRING POINTER
4847 0FE7 F122*A          SKNE    AC0,=183       ; TEST IF LEN (165)
4848 0FE8 1905 A          JMP     ASCFCN         ;      ASC (183), OR VAL (184)
4849 0FE9 F121*A          SKNE    AC0,=184
4850 0FEA 190A A          JMP     VALFCN
4851 0FEB C200 A LENFCN:  LD      AC0,0(AC2)     ; LOAD LENGTH FROM WORD 0 OF STR
4852 0FEC 94B6*A $VAL:    JSR     INTFL
4853 0FED 991E*A          JMP     RESULT
4854 0FEE C600 A ASCFCN:  LD      AC1,0(AC2)
4855 0FEF C201 A          LD      AC0,1(AC2)     ; LOAD 1ST 2 CHAR
4856 0FF0 7900 A          AISZ    AC1,0          ; IF LENGTH=0, RETURN 0
4857 0FF1 1901 A          JMP     .+2
4858 0FF2 5000 A          LI      AC0,0
4859 0FF3 2C10 A          SHR     AC0,8,0        ; RETURN FIRST CHARACTER
4860 0FF4 19F7 A          JMP     $VAL
4861 0FF5 9517*A VALFCN:  JSR     SSCAN          ; VAL: SCAN STRING AND RETURN NU
4862 0FF6 D83E A          ST      AC2,LSTART     ;    VALUE --  0 IF NOT NUMERIC
4863 0FF7 5002 A          LI      AC0,2
4864 0FF8 D03F A          ST      AC0,COL
4865 0FF9 5000 A          LI      AC0,0
4866 0FFA D03A A          ST      AC0,NTYPE
4867 0FFB D03B A          ST      AC0,STYPE
4868 0FFC 9511*A          JSR     DSCAN1
4869 0FFD 1904 A          JMP     $0             ; ... ERROR
4870 0FFE 1903 A          JMP     $0             ; ... END OF LINE
4871 0FFF 5015 A          LI      AC0,21         ; CHECK IF TYPE IS NUMBER
4872 1000 F041 A          SKNE    AC0,CLASS
4873 1001 1903 A          JMP     $NUM
4874 1002 5000 A $0:      LI      AC0,0          ; ERROR OF SOME SORT: RETURN 0
4875 1003 D047 A          ST      AC0,VALUE
4876 1004 D048 A          ST      AC0,VALUE+1
4877 1005 9509*A $NUM:    JSR     RSCAN          ; RESTORE SCANNER POINTERS
4878 1006 C047 A          LD      AC0,VALUE      ; LOAD VALUE
4879 1007 C448 A          LD      AC1,VALUE+1
4880 1008 9903*A          JMP     RESULT
4881 1009 0D74 T          .POOL   7
     100A 00B7 A
     100B 00B8 A
     100C 0D39 T
     100D 0E4F T
     100E 0EF4 T
     100F 0E54 T
4882  ;
4883  ;        DEF STATEMENT ROUTINES
4884  ;
4885 1010 C843 A XQT63:   LD      AC2,SYMA       ; TEMP = SYMB POINTER
4886 1011 D865 A          ST      AC2,TEMP
4887 1012 C201 A          LD      AC0,1(AC2)     ; CHECK IF PREVIOUSLY DEFINED
4888 1013 450E A          BOC     NZR0,$DUP
```

```
4889 1014 C014 A          LD    AC0,CURLIN   ; PUT LINE NO. INTO SYMBOL TABLE
4890 1015 D201 A          ST    AC0,1(AC2)
4891 1016 D052 A          ST    AC0,FNLINE
4892 1017 8000 A          RTS   0
4893 1018 C865 A XQT64:   LD    AC2,TEMP     ; COPY PARAMETER ADDRESS
4894 1019 C043 A          LD    AC0,SYMA
4895 101A D203 A          ST    AC0,3(AC2)
4896 101B 5000 A          LI    AC0,0
4897 101C D052 A          ST    AC0,FNLINE
4898 101D 8000 A          RTS   0
4899 101E C865 A XQT65:   LD    AC2,TEMP     ; SAVE EXPRESSION START COLUMN
4900 101F C03F A          LD    AC0,COL
4901 1020 D202 A          ST    AC0,2(AC2)
4902 1021 8000 A          RTS   0
4903 1022 5020 A $DUP:    LI    AC0,32       ; DUPLICATE FUNCTION DEF
4904 1023 98C6*A          JMP   SYNERR
4905  ;
4906  ;        USER FUNCTION PROCESSING
4907  ;
4908 1024 5304 A XQT67:   LI    AC3,4        ; START FUNCTION PROCESSING
4909 1025 C03E A          LD    AC0,LSTART   ; SAVE LSTART, COL, FNLINE
4910 1026 949B*A          JSR   PUSH
4911 1027 1922 A          JMP   FNERR2
4912 1028 C03F A          LD    AC0,COL
4913 1029 949B*A          JSR   PUSH
4914 102A 191F A          JMP   FNERR2
4915 102B C052 A          LD    AC0,FNLINE
4916 102C 949B*A          JSR   PUSH
4917 102D 191C A          JMP   FNERR2
4918 102E 94A8*A          JSR   CALCUL       ; PULL PARAMETER
4919 102F D047 A          ST    AC0,VALUE
4920 1030 D443 A          ST    AC1,VALUE+1
4921 1031 5302 A          LI    AC3,2        ; PULL USER FUNCTION POINTER
4922 1032 949C*A          JSR   PULL
4923 1033 94C9*A          JSR   SYSERR
4924 1034 78FC A          AISZ  AC0,-4
4925 1035 94C8*A          JSR   OPERR
4926 1036 949C*A          JSR   PULL
4927 1037 94C9*A          JSR   SYSERR
4928 1038 5E00 A          RCPY  AC0,AC2      ; CHECK TO BE SURE FUNCTION EXIS
4929 1039 C201 A          LD    AC0,1(AC2)
4930 103A 410C A          BOC   ZR0,$FNER
4931 103B C203 A          LD    AC0,3(AC2)
4932 103C 410A A          BOC   ZR0,$FNER
4933 103D 5F00 A          RCPY  AC0,AC3      ; STORE PARAMETER VALUE
4934 103E C047 A          LD    AC0,VALUE
4935 103F C448 A          LD    AC1,VALUE+1
4936 1040 D302 A          ST    AC0,2(AC3)
4937 1041 D703 A          ST    AC1,3(AC3)
4938 1042 C202 A          LD    AC0,2(AC2)   ; SET STARTING COLUMN OF EXPRESS
4939 1043 D03F A          ST    AC0,COL
4940 1044 C601 A          LD    AC1,1(AC2)   ; SET STATEMENT NO.
4941 1045 D452 A          ST    AC1,FNLINE
4942 1046 94AF*A          JSR   EFIND        ; FIND THE LINE
```

```
4943 1047 9965*A $FNER:  JMP     FNERR
4944 1048 D83E A         ST      AC2,LSTART    ; SET SCAN LINE POINTER
4945 1049 8000 A         RTS     0
4946  ;
4947 104A 5012 A FNERR2: LI      AC0,18        ; FUNCTIONS NESTED TOO DEEP
4948 104B 98C5*A         JMP     RUNERR
4949  ;
4950 104C 5304 A XQT68:  LI      AC3,4         ; END FUNCTON PROCESSING
4951 104D 949C*A         JSR     PULL          ; PULL FNLINE, COL, LSTART
4952 104E 94C8*A         JSR     OPERR
4953 104F D052 A         ST      AC0,FNLINE
4954 1050 949C*A         JSR     PULL
4955 1051 94C8*A         JSR     OPERR
4956 1052 D03F A         ST      AC0,COL
4957 1053 949C*A         JSR     PULL
4958 1054 94C8*A         JSR     OPERR
4959 1055 D03E A         ST      AC0,LSTART
4960 1056 8000 A         RTS     0
4961  ;
4962  ;       POKE,OUT COMMANDS - WRITE INTO MEMORY OR PERIPHERALS
4963  ;       WAIT COMMAND - WAIT FOR PERIPHERAL RESPONSE
4964  ;
4965 1057 94A8*A XQT78:  JSR     CALCUL        ; WAIT: GET 3RD PARAMETER (XOR)
4966 1058 94B5*A         JSR     ROUND
4967 1059 5C00 A         NOP
4968 105A 1901 A         JMP     .+2
4969 105B 5000 A XQT79:  LI      AC0,0         ; WAIT, 2 ARGUMENTS
4970 105C D060 A         ST      AC0,OP3
4971 105D 94A8*A XQT66:  JSR     CALCUL        ; GET 2ND PARAMETER (DATA)
4972 105E 94B5*A         JSR     ROUND
4973 105F 5C00 A         NOP
4974 1060 D05E A         ST      AC0,OP2
4975 1061 94A9*A         JSR     GETVAL        ; GET 1ST PARAMETER (ADDRESS)
4976 1062 94B5*A         JSR     ROUND
4977 1063 5C00 A         NOP
4978 1064 D05C A         ST      AC0,OP1
4979 1065 94AA*A         JSR     GETFCN        ; PULL FUNCTION NUMBER
4980 1066 CC5C A         LD      AC3,OP1
4981 1067 C45E A         LD      AC1,OP2
4982 1068 F145*A         SKNE    AC0,=154
4983 1069 1919 A         JMP     $OUT
4984 106A F144*A         SKNE    AC0,=199
4985 106B 191C A         JMP     $WAIT
4986 106C C011 A         LD      AC0,PROT      ; POKE: CHECK FOR PROTECTION
4987 106D 410F A         BOC     ZR0,$POKE
4988 106E 5CC0 A         RCPY    AC3,AC0
4989 106F 4111 A         BOC     ZR0,$44       ; POKE ERROR IF LOC 0
4990 1070 F0CD*A         SKNE    AC0,=2        ; POKE ERROR IF LOC 2
4991 1071 190F A         JMP     $44
4992 1072 9C33 A         SKG     AC0,ESTART    ; POKE ERROR IF WRITING INTO BAS
4993 1073 9D3C*A         SKG     AC0,=FIRST-1
4994 1074 1901 A         JMP     .+2
4995 1075 190B A         JMP     $44
4996 1076 9CD1*A         SKG     AC0,=0F       ; POKE ERROR IF LOC 0A-0F
```

```
4997 1077 9CCE*A          SKG     AC0,=9
4998 1078 1901 A          JMP     .+2
4999 1079 1907 A          JMP     $44
5000 107A C439 A          LD      AC1,TOPRAM    ; POKE ERROR IF DISK RAM AREA
5001 107B 94A4*A          JSR     SKL
5002 107C 1904 A          JMP     $44
5003 107D C45E A $POKE:   LD      AC1,OP2       ; POKE - WRITE INTO MEMORY
5004 107E D700 A          ST      AC1,(AC3)
5005 107F F700 A          SKNE    AC1,(AC3)     ; VERIFY
5006 1080 8000 A          RTS     0
5007 1081 502C A $44:     LI      AC0,44        ; POKE ERROR (WARNING ONLY)
5008 1082 98C4*A          JMP     WARNING
5009 1083 5CC0 A $OUT:    RCPY    AC3,AC0       ; OUT - WRITE TO PERIPHERAL
5010 1084 A4D7*A          OR      AC0,=08000
5011 1085 5F00 A          RCPY    AC0,AC3
5012 1086 D700 A          ST      AC1,(AC3)
5013 1087 8000 A          RTS     0
5014 1088 C300 A $WAIT:   LD      AC0,(AC3)     ; GET PERIPHERAL DATA
5015 1089 C460 A          LD      AC1,OP3       ; XOR WITH 3RD ARG
5016 108A 5840 A          RXOR    AC1,AC0
5017 108B A85E A          AND     AC0,OP2       ; AND WITH 2ND ARG
5018 108C 41FB A          BOC     ZR0,$WAIT     ; WAIT UNTIL NON-ZERO
5019 108D 8000 A          RTS     0             ; WAIT IS DONE
5020  ;
5021  ;        NUMERIC->STRING FUNCTIONS: CHR$, STR$, HEX$
5022  ;
5023 108E 94A5*A XQT74:   JSR     STRTMP        ; ALLOCATE TEMP STRING FOR RESUL
5024 108F 94A8*A          JSR     CALCUL        ; GET OPERAND
5025 1090 D071 A          ST      AC0,X
5026 1091 D472 A          ST      AC1,X+1
5027 1092 94AA*A          JSR     GETFCN        ; PULL FUNCTION NUMBER
5028 1093 5E00 A          RCPY    AC0,AC2
5029 1094 C071 A          LD      AC0,X         ; LOAD ARGUMENT
5030 1095 C472 A          LD      AC1,X+1
5031 1096 F91A*A          SKNE    AC2,=181
5032 1097 190F A          JMP     CHRFCN
5033 1098 CC1C A          LD      AC3,OUTDVC    ; STR$,HEX$: SAVE OUTDVC
5034 1099 6300 A          PUSH    AC3
5035 109A 5304 A          LI      AC3,4         ; OUTDVC = STRING
5036 109B DC1C A          ST      AC3,OUTDVC
5037 109C F915*A          SKNE    AC2,=194
5038 109D 1904 A          JMP     HEXFCN
5039 109E 94BA*A STRFCN:  JSR     FLDEC         ; STR$: CONVERT ARGUMENT TO STRI
5040 109F 6400 A $STR2:   PULL    AC0           ; RESTORE OUTPUT DEVICE
5041 10A0 D01C A          ST      AC0,OUTDVC
5042 10A1 98A6*A          JMP     STKSTR        ; STACK THE STRING
5043 10A2 94B5*A HEXFCN:  JSR     ROUND         ; HEX$: CONVERT INTEGER
5044 10A3 5C00 A          NOP                   ;    TO HEX STRING ('XXXX')
5045 10A4 5E00 A          RCPY    AC0,AC2
5046 10A5 953C*A          JSR     PUTW4
5047 10A6 19F8 A          JMP     $STR2         ; CLEAN UP AFTERWARD
5048 10A7 94B5*A CHRFCN:  JSR     ROUND         ; CHR$ - CONVERT RIGHT BYTE OF I
5049 10A8 5C00 A          NOP                   ;    TO AN ASCII CHARACTER
5050 10A9 A8D5*A          AND     AC0,=0FF
```

```
5051 10AA 9498*A          JSR     PUTS
5052 10AB 5C00 A          NOP
5053 10AC 98A6*A          JMP     STKSTR        ; PUT TEMP STRING ON STACK
5054  ;
5055 10AD 0CC7 T          .POOL   6
     10AE 009A A
     10AF 00C7 A
     10B0 00FF T
     10B1 00B5 A
     10B2 00C2 A
5056  ;
5057  ;        SIZE COMMAND - DUMP ADDRESS RANGES
5058  ;
5059 10B3 949F*A SIZE:    JSR     CRLF          ; PRINT ADDRESS RANGES
5060 10B4 94A1*A          JSR     MESG
5061 10B5 1874 T          .WORD   MSZ1
5062 10B6 C833 A          LD      AC2,ESTART
5063 10B7 949D*A          JSR     PUTW
5064 10B8 C835 A          LD      AC2,EEND
5065 10B9 949D*A          JSR     PUTW
5066 10BA 94A1*A          JSR     MESG
5067 10BB 1877 T          .WORD   MSZ3
5068 10BC C835 A          LD      AC2,EEND
5069 10BD 949D*A          JSR     PUTW
5070 10BE C855 A          LD      AC2,ANEXT
5071 10BF 949D*A          JSR     PUTW
5072 10C0 94A1*A          JSR     MESG
5073 10C1 187B T          .WORD   MSZ4
5074 10C2 C851 A          LD      AC2,SY2
5075 10C3 949D*A          JSR     PUTW
5076 10C4 C838 A          LD      AC2,SY1
5077 10C5 949D*A          JSR     PUTW
5078 10C6 989F*A          JMP     CRLF          ; CR/LF , RTS
5079  ;
5080  ;        TRACE - ON OR OFF
5081  ;
5082 10C7 5001 A TRCON:   LI      AC0,1
5083 10C8 1901 A          JMP     .+2
5084 10C9 5000 A TRCOFF:  LI      AC0,0
5085 10CA D015 A          ST      AC0,TRMODE
5086 10CB 8000 A          RTS     0
5087  ;
5088  ;        SET TERMINAL WIDTH
5089  ;
5090 10CC 94A8*A XQT77:   JSR     CALCUL
5091 10CD 94B5*A          JSR     ROUND
5092 10CE 5C00 A          NOP
5093 10CF 9D13*A TWIDTH:  SKG     AC0,=14       ; MUST BE BETWEEN 15 AND 132
5094 10D0 500F A          LI      AC0,15
5095 10D1 9D1B*A          SKG     AC0,=132
5096 10D2 1901 A          JMP     .+2
5097 10D3 C119*A          LD      AC0,=132
5098 10D4 D01E A          ST      AC0,LINLEN
5099 10D5 C01E A          LD      AC0,LINLEN    ; CALCULATE LAST (PACKED) PRINT
```

```
5100 10D6 78F2 A          AISZ    AC0,-14       ; (LAST 14 COLUMNS)
5101 10D7 D01F A          ST      AC0,LZONE
5102 10D8 C0B1*A          LD      AC0,=INBUF    ; CALCULATE END OF INPUT BUFFER
5103 10D9 E01E A          ADD     AC0,LINLEN
5104 10DA D020 A          ST      AC0,ENDBUF
5105 10DB 8000 A          RTS     0
5106  ;
5107  ;        CALL STATEMENT (NO PARAMETERS)
5108  ;
5109 10DC 94A8*A XQTS0:   JSR     CALCUL
5110 10DD 94B5*A          JSR     ROUND
5111 10DE 5C00 A          NOP
5112 10DF 5F00 A          RCPY    AC0,AC3       ; COPY ADDRESS
5113 10E0 1700 A          JSR     (AC3)         ; CALL THE ROUTINE
5114 10E1 8000 A          RTS     0
5115  ;
5116 10E2 03F1 T          .POOL   2
     10E3 000E A
5117  ;
```

```
5118                        .PAGE   'MATH PACKAGE ROUTINES'
5119                        .LOCAL
5120   ;////////////////////////////////////////////////////////////////////////////
5121   ;
5122   ;        FLOATING POINT NUMBER FORMAT:
5123   ;           ( 32 BITS / 2 MEMORY WORDS / 2 REGISTERS )
5124   ;
5125   ;                ------------------------------------------
5126   ;         1:     I       TWO'S COMPLEMENT FRACTION          I
5127   ;                ----------------------I-------------------
5128   ;         2:     I  FRACTION (CONT'D)  I  2'S COMP EXPONENT  I
5129   ;                ------------------------------------------
5130   ;                 15                    8 7                 0
5131   ;
5132   ;////////////////////////////////////////////////////////////////////////////
5133   ;
5134   ;        ALL ROUTINES ARE CALLED AS FOLLOWS:
5135   ;
5136   ;        ON ENTRY,  FIRST OPERAND  IS IN (AC0,AC1);
5137   ;                   SECOND OPERAND IS IN (AC2,AC3).
5138   ;        ON EXIT,   THE RESULT IS LEFT IN (AC0,AC1).
5139   ;                   AC2 AND AC3 ARE NOT SAVED BY ROUTINES
5140   ;                   REQUIRING 2 OPERANDS.   THE OVERFLOW FLAG
5141   ;                   IS SET ACCORDING TO THE OVERFLOW CONDITION.
5142   ;
5143   ;        ROUTINES CALLED BY THE EXPRESSION ANALYZER ALSO
5144   ;        HAVE THE FIRST OPERAND IN (OP1,OP1+1) AND THE
5145   ;        SECOND OPERAND IN (OP2,OP2+1).
5146   ;
5147   ;*****************************************************************************
5148   ;
5149   ;        FLCMP   FLOATING COMPARE  (INTEGER RESULT)
5150   ;                (AC0) = SGN((AC0,AC1)-(AC2,AC3))
5151   ;        RESULTS:    1   (AC0,AC1) > (AC2,AC3)
5152   ;                    0   (AC0,AC1) = (AC2,AC3)
5153   ;                   -1   (AC0,AC1) < (AC2,AC3)
5154   ;
5155 10E4 9509*A FLCMP:   JSR     FLSUB         ; COMPARE: (AC0,AC1) - (AC2,AC3)
5156 10E5 5201 A SIGN:    LI      AC2,1         ; DETERMINE SIGN OF (AC0,AC1)
5157 10E6 4103 A          BOC     ZRO,$Z
5158 10E7 4203 A          BOC     POS,$X
5159 10E8 52FF A          LI      AC2,-1        ; NEGATIVE
5160 10E9 1901 A          JMP     $X
5161 10EA 5200 A $Z:      LI      AC2,0         ; ZERO
5162 10EB 5C80 A $X:      RCPY    AC2,AC0       ; COPY RESULT TO AC0
5163 10EC 8000 A          RTS     0
5164 10ED 0084 A          .POOL   2
     10EE 11C5 T
5165   ;
5166   ;        LOGICAL ROUTINES
5167   ;
5168 10EF 94B4*A FLNOT:   JSR     IFIX          ; NOT OPERATOR
5169 10F0 5C00 A          NOP
5170 10F1 7000 A          CAI     AC0,0
```

```
5171 10F2 1907 A          JMP     $RTN
5172  ;
5173 10F3 150D A FLAND:   JSR     $OP1          ; AND
5174 10F4 1510 A          JSR     $OP2
5175 10F5 A85C A          AND     AC0,OP1
5176 10F6 1903 A          JMP     $RTN
5177  ;
5178 10F7 1509 A FLOR:    JSR     $OP1          ; OR
5179 10F8 150C A          JSR     $OP2
5180 10F9 A45C A          OR      AC0,OP1
5181 10FA 94B6*A $RTN:    JSR     INTFL
5182 10FB 8000 A          RTS     0
5183  ;
5184 10FC 1504 A FLXOR:   JSR     $OP1          ; XOR
5185 10FD 1507 A          JSR     $OP2
5186 10FE C45C A          LD      AC1,OP1
5187 10FF 5840 A          RXOR    AC1,AC0
5188 1100 19F9 A          JMP     $RTN
5189  ;
5190 1101 94B4*A $OP1:    JSR     IFIX          ; OP1 = INT (AC0,AC1)
5191 1102 5C00 A          NOP
5192 1103 D05C A          ST      AC0,OP1
5193 1104 8000 A          RTS
5194 1105         $OP2:   DLD     AC0,OP2       ; AC0 = INT (OP2,OP2+1)
5195 1107 94B4*A          JSR     IFIX
5196 1108 5C00 A          NOP
5197 1109 8000 A          RTS     0
5198  ;
5199  ;        COMPARISON OPERATORS
5200  ;
5201  ;          NOTE: STRING COMPARISONS ENTER AT LABEL+1
5202  ;                WITH COMPARISON VALUE ALREADY IN AC0
5203  ;
5204 110A 15D9 A FLNE:    JSR     FLCMP         ; <>
5205 110B 4111 A          BOC     ZR0,FALSE
5206 110C 1913 A          JMP     TRUE
5207 110D 15D6 A FLEQ:    JSR     FLCMP         ; =
5208 110E 4111 A          BOC     ZR0,TRUE
5209 110F 190D A          JMP     FALSE
5210 1110 15D3 A FLGT:    JSR     FLCMP         ; >
5211 1111 410B A          BOC     ZR0,FALSE
5212 1112 4B0A A          BOC     NEG,FALSE
5213 1113 190C A          JMP     TRUE
5214 1114 15CF A FLGE:    JSR     FLCMP         ; >=
5215 1115 420A A          BOC     POS,TRUE
5216 1116 1906 A          JMP     FALSE
5217 1117 15CC A FLLT:    JSR     FLCMP         ; <
5218 1118 4204 A          BOC     POS,FALSE
5219 1119 1906 A          JMP     TRUE
5220 111A 15C9 A FLLE:    JSR     FLCMP         ; <=
5221 111B 4104 A          BOC     ZR0,TRUE
5222 111C 4B03 A          BOC     NEG,TRUE
5223 111D 5000 A FALSE:   LI      AC0,0         ; FALSE: 0 RESULT
5224 111E 5100 A          LI      AC1,0
```

```
5225 111F 8000 A          RTS     0
5226 1120 C0D8*A TRUE:    LD      AC0,=0C000    ; TRUE: -1 RESULT
5227 1121 5101 A          LI      AC1,1
5228 1122 8000 A          RTS     0
5229  ;
5230  ;        MIN, MAX COMPARISONS
5231  ;
5232 1123 15C0 A FLMAX:   JSR     FLCMP         ; COMPARE 2 OPERANDS,
5233 1124 4205 A          BOC     POS,$1        ;    USE LARGER
5234 1125           $2:    DLD     AC0,OP2
5235 1127 8000 A          RTS     0
5236 1128 15BB A FLMIN:   JSR     FLCMP         ; COMPARE 2 OPERANDS,
5237 1129 42FB A          BOC     POS,$2        ;    USE SMALLER
5238 112A           $1:    DLD     AC0,OP1
5239 112C 8000 A          RTS     0
5240  ;
5241  ;        FUNCTIONS
5242  ;
5243 112D 4201 A ABS:     BOC     POS,.+2       ; ABSOLUTE VALUE
5244 112E 94BF*A          JSR     NEGATE
5245 112F 8000 A          RTS     0
5246  ;
5247 1130 15B4 A SGN:     JSR     SIGN          ; SIGN OF NUMBER
5248 1131 98B6*A          JMP     INTFL
5249  ;
5250 1132 C01D A POSFCN:  LD      AC0,OUTCOL    ; PRINT POSITION
5251 1133 98B6*A          JMP     INTFL
5252  ;
5253 1134 94B5*A PEEK:    JSR     ROUND         ; READ MEMORY
5254 1135 5C00 A          NOP
5255 1136 1903 A          JMP     $IN
5256  ;
5257 1137 94B5*A INP:     JSR     ROUND         ; READ PERIPHERAL
5258 1138 5C00 A          NOP
5259 1139 A4D7*A          OR      AC0,=08000    ; FORCE ADDRESS BIT 15
5260 113A 5F00 A $IN:     RCPY    AC0,AC3       ; READ MEMORY OR PERIPHERAL
5261 113B C300 A          LD      AC0,(AC3)
5262 113C 98B6*A          JMP     INTFL
5263  ;
5264 113D 94B5*A USR:     JSR     ROUND         ; CONVERT ARGUMENT TO INTEGER
5265 113E 5C00 A          NOP
5266 113F 9410 A          JSR     @USER         ; CALL USER FUNCTION
5267 1140 98B6*A          JMP     INTFL         ; CONVERT RESULT TO FLOATING PT.
5268  ;
5269 1141 5024 A NOUSER:  LI      AC0,36        ; NO USER FUNCTION
5270 1142 98C5*A          JMP     RUNERR
5271  ;
5272 1143 C417 A BRKFCN:  LD      AC1,BRKMOD    ; BRK FUNCTION
5273 1144 4103 A          BOC     ZR0,$BOFF     ; TEST ARGUMENT
5274 1145 4B03 A          BOC     NEG,$RVAL     ;    X<0: RETURN STATUS
5275 1146 5101 A          LI      AC1,1         ;    X>0: TURN ON BREAK CAPABILI
5276 1147 1901 A          JMP     $RVAL
5277 1148 5100 A $BOFF:   LI      AC1,0         ;    X=0: TURN OFF BREAK CAPABIL
5278 1149 C017 A $RVAL:   LD      AC0,BRKMOD    ; RETURN PREVIOUS STATUS
```

```
5279 114A D417 A          ST      AC1,BRKMOD
5280 114B 98B6*A          JMP     INTFL
5281  ;
5282                      .LOCAL
5283  ;
5284  ;       *** FLOATING POINT ROUTINES ***
5285  ;
5286  ;
5287  ;       FLMULT : (AC0,AC1) * (AC2,AC3)
5288  ;
5289 114C 155A A FLMULT: JSR     SETUP         ; EXTRACT EXPONENTS AND SET UP D
5290 114D 1564 A          JSR     SETUP2
5291 114E                 DST     AC0,X         ; SAVE X1,X2
5292 1150 5DC0 A          RCPY    AC3,AC1
5293 1151 2D02 A          SHR     AC1,1,0
5294 1152 9523*A          JSR     FRMULT
5295 1153                 DST     AC0,TT3       ; TT3 = (Y2*X1)/2
5296 1155 C472 A          LD      AC1,X+1
5297 1156 2D02 A          SHR     AC1,1,0
5298 1157 5C80 A          RCPY    AC2,AC0       ; AC0 = Y1
5299 1158 951D*A          JSR     FRMULT        ; AC0,AC1=(Y1*X2)/2
5300 1159 1516 A          JSR     ADDTT3        ; AC0,AC1=(Y1*X2+Y2*X1)/2
5301 115A 6D00 A          RXCH    AC0,AC1       ; ROUND RESULT TO 15 BITS
5302 115B 4205 A          BOC     POS,$1
5303 115C E4DD*A          ADD     AC1,=1
5304 115D 4C01 A          BOC     OVF,$2
5305 115E 1902 A          JMP     $1
5306 115F 79FF A $2:      AISZ    AC1,-1
5307 1160 5C00 A          NOP
5308 1161 5000 A $1:      LI      AC0,0         ; TT3=PARTIAL PRODUCT TO BE ADDE
5309 1162 9514*A          JSR     DSHL          ;    TO LOW ORDER WORD OF RESULT
5310 1163                 DST     AC0,TT3       ;    ((Y1*X2+Y2*X1)/2**15)
5311 1165 5D80 A          RCPY    AC2,AC1       ; AC1 = Y1
5312 1166 C071 A          LD      AC0,X         ; AC0 = X1
5313 1167 950E*A          JSR     FRMULT        ; AC0,AC1 = X1*Y1
5314 1168 1507 A          JSR     ADDTT3
5315 1169 C873 A          LD      AC2,Y         ; RESULTANT EXPONENT = SUM OF EX
5316 116A E874 A          ADD     AC2,Y2
5317 116B 6200 A          PUSH    AC2           ; PUSH EXPONENT ONTO STACK
5318 116C C875 A FRTN:    LD      AC2,FLSIGN    ; CHECK RESULT SIGN
5319 116D 7A00 A          AISZ    AC2,0
5320 116E 9509*A          JSR     D2C
5321 116F 98B9*A          JMP     IEXP
5322 1170 E485 A ADDTT3:  ADD     AC1,TT3+1     ; (AC0,AC1) = (AC0,AC1)+(TT3,TT3
5323 1171 4A01 A          BOC     CRY,.+2
5324 1172 1901 A          JMP     .+2
5325 1173 E0DD*A          ADD     AC0,=1
5326 1174 E084 A          ADD     AC0,TT3
5327 1175 8000 A          RTS     0
5328  ;
5329 1176 14FC T          .POOL   4
     1177 1219 T
     1178 120A T
     1179 0000 A
```

```
5330  ;
5331  ;
5332  ;        FLDIV : (AC0,AC1) / (AC2,AC3)
5333  ;
5334  ;        ERROR EXITS IF DIVISOR IS ZERO
5335  ;
5336                        .LOCAL
5337 117A 7A00 A FLDIV:   AISZ    AC2,0          ; CHECK FOR ZERO DIVISOR
5338 117B 1903 A          JMP     $FLD           ; INPUTS ASSUMED NORMALIZED
5339 117C 5D00 A          RCPY    AC0,AC1        ; ERROR - DIVISION BY ZERO
5340 117D 5028 A          LI      AC0,40
5341 117E 99FA*A          JMP     MAXVAL
5342 117F 1527 A $FLD:    JSR     SETUP          ; EXTRACT EXPONENTS AND SET UP D
5343 1180 1531 A          JSR     SETUP2
5344 1181 6300 A          PUSH    AC3            ; SHIFT DIVIDEND 1 RIGHT TO PREV
5345 1182 5301 A          LI      AC3,1
5346 1183 94C0*A          JSR     ADSHR
5347 1184 6700 A          PULL    AC3
5348 1185 D082 A          ST      AC0,TT2        ; FRACTIONAL DIVIDE
5349 1186 501F A          LI      AC0,31         ; BIT COUNT
5350 1187 D081 A          ST      AC0,TT1        ; TT1 = BIT COUNT
5351 1188 C082 A          LD      AC0,TT2
5352 1189                 DST     AC2,TT3        ; TT3 = DIVISOR
5353 118B 5200 A          LI      AC2,0          ; AC2,AC3 WILL BE THE
5354 118C 5300 A          LI      AC3,0          ;    DEVELOPING QUOTIENT
5355 118D 2903 A $3:      SHL     AC1,1,1
5356 118E 2003 A          ROL     AC0,1,1
5357 118F D082 A          ST      AC0,TT2        ; THIS WILL BE RESTORING DIVISIO
5358 1190 D483 A          ST      AC1,TT2+1      ; AND TT2 = PREVIOUS DIVIDEND
5359 1191 3780 A          SFLG    CARRY          ; AC0,AC1 = PARTIAL REMAINDER
5360 1192 6D00 A          RXCH    AC0,AC1        ;    (SUBTRACT TT3 FROM AC0,AC1)
5361 1193 9085 A          SUBB    AC0,TT3+1
5362 1194 6D00 A          RXCH    AC0,AC1
5363 1195 9084 A          SUBB    AC0,TT3
5364 1196 2B03 A          SHL     AC3,1,1        ; PREPARE QUOTIENT
5365 1197 2203 A          ROL     AC2,1,1
5366 1198 4207 A          BOC     POS,$1         ; BRANCH IF PARTIAL REMAINDER >=
5367 1199                 DLD     AC0,TT2        ; GET PREVIOUS DIVIDEND
5368 119B AC81 A $2:      DSZ     TT1            ; DONE?
5369 119C 19F0 A          JMP     $3             ;    NOPE
5370 119D 6E00 A          RXCH    AC0,AC2        ;    YES, PREPARE OUTPUT
5371 119E 6F40 A          RXCH    AC1,AC3
5372 119F 1902 A          JMP     $4
5373 11A0 ECDD*A $1:      ADD     AC3,=1         ; SET QUOTIENT BIT = 1
5374 11A1 19F9 A          JMP     $2
5375 11A2 C874 A $4:      LD      AC2,Y2         ; RESULTANT EXPONENT = E(DIVIDEN
5376 11A3 7202 A          CAI     AC2,2          ;                      - E(DIVIS
5377 11A4 E873 A          ADD     AC2,Y
5378 11A5 6200 A          PUSH    AC2            ; PUSH EXPONENT ONTO STACK
5379 11A6 19C5 A          JMP     FRTN
5380  ;
5381 11A7 9550*A SETUP:   JSR     E2S            ; EXTRACT EXPONENTS FROM AC1 AND
5382 11A8 D475 A          ST      AC1,FLSIGN     ;    AND STORE IN Y, Y2
5383 11A9 6500 A          PULL    AC1
```

```
5384 11AA D473 A          ST      AC1,Y
5385 11AB 5DC0 A          RCPY    AC3,AC1
5386 11AC 954B*A          JSR     E2S
5387 11AD 5F40 A          RCPY    AC1,AC3
5388 11AE 6500 A.         PULL    AC1
5389 11AF D474 A          ST      AC1,Y2
5390 11B0 C475 A          LD      AC1,FLSIGN
5391 11B1 8000 A          RTS     0
5392  ;
5393 11B2 6000 A SETUP2:  PUSH    AC0           ; TAKE ABS VALUES OF 2 OPERANDS,
5394 11B3 5000 A          LI      AC0,0         ;    SET SIGN FLAG FOR RESULT
5395 11B4 D075 A          ST      AC0,FLSIGN    ;    (FLSIGN=0--POSITIVE)
5396 11B5 6400 A          PULL    AC0           ;    NOTE: CALL AFTER SETUP
5397 11B6 420Z A          BOC     POS,$S1
5398 11B7 1552 A          JSR     D2C
5399 11B8 8C75 A          ISZ     FLSIGN
5400 11B9 6E00 A $S1:     RXCH    AC0,AC2       ; NOW TEST 2ND OPERAND
5401 11BA 6F40 A          RXCH    AC1,AC3
5402 11BB 4203 A          BOC     POS,$S2
5403 11BC 154D A          JSR     D2C
5404 11BD AC75 A          DSZ     FLSIGN
5405 11BE 5C00 A          NOP
5406 11BF 6E00 A $S2:     RXCH    AC0,AC2       ; RETURN CORRECT REG
5407 11C0 6F40 A          RXCH    AC1,AC3
5408 11C1 8000 A          RTS     0
5409  ;
5410  ;
5411  ;        FLADD : (AC0,AC1) + (AC2,AC3)
5412  ;        FLSUB : (AC0,AC1) - (AC2,AC3)
5413  ;
5414                    .LOCAL
5415 11C2 6E00 A $XCH:    RXCH    AC0,AC2       ; EXCHANGE OPERANDS
5416 11C3 6F40 A          RXCH    AC1,AC3
5417 11C4 8000 A          RTS     0
5418 11C5 15FC A FLSUB:   JSR     $XCH          ; SUBTRACT: NEGATE 2ND OPERAND
5419 11C6 1540 A          JSR     NEGATE
5420 11C7 F8CC*A FLADD:   SKNE    AC2,=0        ; CHECK FOR (AC2,AC3)=0
5421 11C8 8000 A          RTS     0
5422 11C9 41FB A          BOC     ZR0,$XCH      ; CHECK FOR (AC0,AC1)=0
5423 11CA 15DC A          JSR     SETUP         ; SEPARATE FRACTIONS AND EXPONEN
5424 11CB                 DST     AC2,X         ; COMPARE OPERANDS: SMALLER TO (
5425 11CD CC73 A          LD      AC3,Y         ; AC3 = DIFFERENCE IN POWERS
5426 11CE 7301 A          CAI     AC3,1
5427 11CF EC74 A          ADD     AC3,Y2
5428 11D0 C874 A          LD      AC2,Y2        ; AC2 = RESULTANT EXPONENT (LARG
5429 11D1 6F00 A          RXCH    AC0,AC3
5430 11D2 420C A          BOC     POS,$X2       ; CHECK SIGN OF DIFFERENCE
5431 11D3 6F00 A          RXCH    AC0,AC3
5432 11D4           $X1:  DLD     AC2,X         ; EXCHANGE OPERANDS
5433 11D6 6E00 A          RXCH    AC0,AC2
5434 11D7 6F40 A          RXCH    AC1,AC3
5435 11D8                 DST     AC2,X
5436 11DA C873 A          LD      AC2,Y         ; RELOAD EXPONENT, DIFFERENCE
5437 11DB CC74 A          LD      AC3,Y2
```

```
5438 11DC 7301 A          CAI     AC3,1
5439 11DD EC73 A          ADD     AC3,Y
5440 11DE 1901 A          JMP     $X3
5441 11DF 6F00 A $X2:     RXCH    AC0,AC3
5442 11E0 6200 A.$X3:     PUSH    AC2             ; PUSH RESULTANT EXPONENT ONTO S
5443 11E1 6F00 A          RXCH    AC0,AC3
5444 11E2 9D13*A          SKG     AC0,=22         ; NUMBER OF SHIFTS > 22 ?
5445 11E3 1903 A          JMP     $FLA1           ;    NO
5446 11E4                 DLD     AC0,X           ;    YES - USE LARGER NUMBER
5447 11E6 1936 A          JMP     IEXP
5448 11E7 6F00 A $FLA1:   RXCH    AC0,AC3         ; SHIFT SMALLER NUMBER TO THE RI
5449 11E8 1527 A          JSR     ADSHR
5450 11E9 E472 A          ADD     AC1,X+1         ; ADD TO X
5451 11EA C871 A          LD      AC2,X
5452 11EB 7480 A          RADC    AC2,AC0
5453 11EC 4C01 A          BOC     OVF,$OVF
5454 11ED 192F A          JMP     IEXP
5455 11EE 5301 A $OVF:    LI      AC3,1
5456 11EF 1520 A          JSR     ADSHR
5457 11F0 CCD7*A          LD      AC3,=08000
5458 11F1 58C0 A          RXOR    AC3,AC0         ; COMPLEMENT SIGN BIT
5459 11F2 6700 A          PULL    AC3
5460 11F3 ECDD*A          ADD     AC3,=1          ; INC EXPO TO COMPENSATE FOR ADS
5461 11F4 6300 A          PUSH    AC3
5462 11F5 1927 A          JMP     IEXP
5463  ;
5464 11F6 0016 A          .PTR    22,0FF7F
     11F7 FF7F A
5465 11F8 1266 T          .POOL   2
     11F9 0000 A
5466  ;
5467  ;        MOD :  OP1 MOD OP2 = OP1 - INT (OP1/OP2) * OP2
5468  ;
5469 11FA        FLMOD:   DST     AC0,OP1         ; SAVE OPERANDS FOR SURE
5470 11FC                 DST     AC2,OP2
5471 11FE 94BE*A          JSR     FLDIV
5472 11FF 94B7*A          JSR     FLINT
5473 1200                 DLD     AC2,OP2
5474 1202 94BD*A          JSR     FLMULT
5475 1203 1503 A          JSR     NEGATE
5476 1204                 DLD     AC2,OP1
5477 1206 19C0 A          JMP     FLADD
5478  ;
5479  ;        NEGATE : FLOATING POINT TWOS COMPLEMENT
5480  ;
5481 1207 155E A NEGATE:  JSR     E2S
5482 1208 1501 A          JSR     D2C
5483 1209 1913 A          JMP     IEXP
5484  ;
5485  ;        D2C : DOUBLE 2'S COMPLEMENT (FRACTIONAL)
5486  ;
5487 120A 7000 A D2C:     CAI     AC0,0           ; COMPLEMENT AC0 & AC1
5488 120B 7100 A          CAI     AC1,0
5489 120C 7901 A          AISZ    AC1,1           ; ADD 1
```

```
5490 120D 8000 A          RTS     0
5491 120E E0DD*A          ADD     AC0,=1          ; ADD CARRY FROM AC1
5492 120F 8000 A          RTS     0
5493  ;
5494  ;        ADSHR : ARITHMETIC DOUBLE SHIFT RIGHT
5495  ;                SHIFTS (AC0,AC1) BY (AC3) BITS
5496  ;
5497 1210 FCCC*A ADSHR:   SKNE    AC3,=0          ; DONE WHEN COUNT=0
5498 1211 8000 A          RTS
5499 1212 3800 A          PFLG    LINK            ; CLEAR LINK
5500 1213 4201 A          BOC     POS,.+2         ; TEST SIGN
5501 1214 3880 A          SFLG    LINK            ; SET LINK IF NEGATIVE
5502 1215 2403 A          ROR     AC0,1,1         ; ROTATE AC0 & LINK
5503 1216 2D03 A          SHR     AC1,1,1
5504 1217 ECDB*A          ADD     AC3,=-1         ; DECREMENT COUNT AND CONTINUE
5505 1218 19F7 A          JMP     ADSHR
5506  ;
5507  ;        DSHL  : DOUBLE SHIFT LEFT 1 BIT
5508  ;
5509 1219 2903 A DSHL:    SHL     AC1,1,1
5510 121A 2003 A          ROL     AC0,1,1
5511 121B 8000 A          RTS     0
5512  ;
5513  ;        FLNORM : FLOATING POINT NORMALIZE
5514  ;        IEXP   : INSERT EXPONENT FROM STACK - COME HERE BY JMP ONLY!
5515  ;        PWR2   : MULTIPLY BY 2**(AC3)
5516  ;
5517  ;        THE LEAST SIGNIFICANT 8 BITS OF THE EXPONENT ARE
5518  ;        INSERTED INTO THE LEAST SIGNIFICANT 8 BITS OF AC1 AFTER
5519  ;        (AC0,AC1) IS NORMALIZED; UNDERFLOW AND OVERFLOW ARE CHECKED.
5520  ;
5521 121C 1549 A FLNORM: JSR     E2S             ; EXPONENT TO STACK
5522 121D 1000 A IEXP:   XCHRS   AC0             ; PULL EXPONENT, SAVE IN XTEMP
5523 121E D080 A          ST      AC0,XTEMP
5524 121F 6400 A          PULL    AC0
5525 1220 151E A          JSR     FRNORM          ; NORMALIZE FRACTION
5526 1221 DC70 A          ST      AC3,TAC3        ; SAVE AC3; RECOVER EXPONENT
5527 1222 CC80 A          LD      AC3,XTEMP
5528 1223 6D00 A          RXCH    AC0,AC1
5529 1224 A8DA*A          AND     AC0,=0FF00      ; MASK OUT EXPONENT FIELD
5530 1225 6D00 A          RXCH    AC0,AC1
5531 1226 7900 A          AISZ    AC1,0           ; CHECK FOR 8000 0000 AGAIN
5532 1227 1905 A          JMP     $CKZ
5533 1228 F0D7*A          SKNE    AC0,=08000
5534 1229 1901 A          JMP     .+2
5535 122A 1902 A          JMP     $CKZ
5536 122B C0D8*A          LD      AC0,=0C000
5537 122C ECDD*A          ADD     AC3,=1
5538 122D 4502 A $CKZ:    BOC     NZRO,$CKE       ; IF MANTISSA IS ZERO, CLEAR EXP
5539 122E F4CC*A          SKNE    AC1,=0
5540 122F 5300 A          LI      AC3,0           ; ZERO - CLEAR EXPONENT
5541 1230 6CC0 A $CKE:    RXCH    AC3,AC0         ; CHECK EXPONENT VALIDITY
5542 1231 9CD3*A          SKG     AC0,=127
5543 1232 9DC4*A          SKG     AC0,=-129
```

```
5544 1233 193D A          JMP     FRANGE          ; OVERFLOW OR UNDERFLOW
5545 1234 A8D5*A          AND     AC0,=0FF
5546 1235 6F00 A          RXCH    AC0,AC3
5547 1236 59C0 A          RXOR    AC3,AC1
5548 1237 CC70 A          LD      AC3,TAC3
5549 1238 3600 A          PFLG    OVFLW           ; NO OVERFLOW
5550 1239 8000 A          RTS     0
5551 123A 152B A PWR2:    JSR     E2S             ; MULTIPLY BY 2**(AC3)
5552 123B 1C00 A          XCHRS   AC0             ;    BY ADDING (AC3) TO EXPONENT
5553 123C 68C0 A          RADD    AC3,AC0
5554 123D 1C00 A          XCHRS   AC0
5555 123E 19DE A          JMP     IEXP
5556  ;
5557  ;        FRNORM : FRACTIONAL NORMALIZE (USED BY IEXP)
5558  ;                 EXPONENT IS IN XTEMP
5559  ;
5560                    .LOCAL
5561 123F B8D6*A FRNORM: SKAZ    AC0,=07FFF      ; 0000 0000 OR 8000 0000 ?
5562 1240 1902 A          JMP     $1
5563 1241 F4CC*A          SKNE    AC1,=0
5564 1242 8000 A          RTS     0               ;    YES - OK AS IS
5565 1243 4203 A $1:      BOC     POS,FRPN        ; IF VALUE IS NEGATIVE THEN:
5566 1244 15C5 A          JSR     D2C             ;    COMPLEMENT
5567 1245 1501 A          JSR     FRPN            ;    NORMALIZE
5568 1246 19C3 A          JMP     D2C             ;    COMPLEMENT AND RETURN
5569 1247 4106 A FRPN:    BOC     ZR0,$4          ; NORMALIZE POSITIVE (AC0,AC1)
5570 1248 B8DC*A $2:      SKAZ    AC0,=04000      ; TEST BIT 14
5571 1249 190D A          JMP     $ROUND          ;    BIT 14 = 1, DONE
5572 124A 15CE A          JSR     DSHL            ;    BIT 14 = 0, SHIFT 1 LEFT
5573 124B AC80 A          DSZ     XTEMP           ; DECREMENT EXPONENT
5574 124C 5C00 A          NOP
5575 124D 19FA A          JMP     $2              ; TEST NEXT CONFIGURATION
5576 124E 5C40 A $4:      RCPY    AC1,AC0         ; AC0=0, SO MOVE AC1 TO AC0
5577 124F 51F1 A          LI      AC1,-15         ; (CUTS MAX # SHIFTS TO 15)
5578 1250 E480 A          ADD     AC1,XTEMP       ; ADJUST EXPONENT (SUBTRACT 15)
5579 1251 D480 A          ST      AC1,XTEMP
5580 1252 5100 A          LI      AC1,0           ; SHIFT AC0 RIGHT 1 BIT
5581 1253 3800 A          PFLG    LINK
5582 1254 2403 A          ROR     AC0,1,1
5583 1255 2D03 A          SHR     AC1,1,1
5584 1256 19F1 A          JMP     $2
5585 1257 E4D4*A $ROUND:  ADD     AC1,=080        ; ROUND RESULT TO 24 BITS
5586 1258 6D00 A          RXCH    AC0,AC1
5587 1259 A8DA*A          AND     AC0,=0FF00      ; MASK OUT EXPONENT FIELD
5588 125A 6D00 A          RXCH    AC0,AC1
5589 125B 6200 A          PUSH    AC2
5590 125C 5200 A          LI      AC2,0
5591 125D 7480 A          RADC    AC2,AC0
5592 125E 6600 A          PULL    AC2
5593 125F 4C01 A          BOC     OVF,$OVF
5594 1260 8000 A          RTS     0
5595 1261 C0DC*A $OVF:    LD      AC0,=04000      ; OVFL (8000 0000)
5596 1262 5100 A          LI      AC1,0           ; CONVERT TO 4000 0000,
5597 1263 8C80 A          ISZ     XTEMP           ; & INCREMENT EXPONENT
```

```
5598 1264 8000 A          RTS     0
5599 1265 8000 A          RTS     0
5600  ;
5601  ;        E2S - EXTRACT EXPONENT TO STACK & CLEAR EXPONENT BITS
5602  ;
5603 1266 D46E A E2S:     ST      AC1,TAC1      ; SIGN EXTEND EXPONENT AND PUSH
5604 1267 6D00 A          RXCH    AC0,AC1       ;    ONTO STACK
5605 1268 A8D5*A          AND     AC0,=0FF
5606 1269 B8D4*A          SKAZ    AC0,=080
5607 126A A4DA*A          OR      AC0,=0FF00
5608 126B 1C00 A          XCHRS   AC0           ; EXCHANGE WITH RETURN
5609 126C 6000 A          PUSH    AC0
5610 126D C06E A          LD      AC0,TAC1
5611 126E A8DA*A          AND     AC0,=0FF00    ; CLEAR EXPONENT FIELD IN AC1
5612 126F 6D00 A          RXCH    AC0,AC1
5613 1270 8000 A          RTS     0
5614  ;
5615 1271 6DC0 A FRANGE:  RXCH    AC3,AC1       ; EXPN INTO AC0, DATA SIGN INTO
5616 1272 4203 A          BOC     POS,OVFERR
5617 1273 5030 A          LI      AC0,48        ; UNDERFLOW
5618 1274 94C4*A          JSR     WARNING
5619 1275 9983*A          JMP     RSLT0         ; RETURN 0 RESULT
5620 1276 5026 A OVFERR:  LI      AC0,38        ; OVERFLOW
5621 1277 94C4*A MAXVAL:  JSR     WARNING
5622 1278 3680 A          SFLG    OVFLW         ; SET OVERFLOW FLAG
5623 1279 5C40 A          RCPY    AC1,AC0
5624 127A 4203 A          BOC     POS,$MAXP
5625 127B C0D7*A          LD      AC0,=08000    ; MAX NEGATIVE VALUE
5626 127C C504*A          LD      AC1,=017F
5627 127D 8000 A          RTS     0
5628 127E C0D6*A $MAXP:   LD      AC0,=07FFF    ; MAX POSITIVE VALUE
5629 127F C502*A          LD      AC1,=0FF7F
5630 1280 8000 A          RTS     0
5631 1281 017F A          .POOL   3
     1282 FF7F A
     1283 0000 A
5632  ;
5633  ;        INTFL : INTEGER TO FLOATING POINT CONVERSION
5634  ;
5635 1284 DC70 A INTFL:   ST      AC3,TAC3
5636 1285 5100 A          LI      AC1,0         ; CLEAR FRACTIONAL BITS
5637 1286 530F A          LI      AC3,15        ; 15 BIT INTEGER MAGNITUDE
5638 1287 6300 A          PUSH    AC3
5639 1288 CC70 A          LD      AC3,TAC3
5640 1289 1993 A          JMP     IEXP
5641  ;
5642  ;        FLINT : INTEGER FUNCTION (FLOATING POINT OPERAND & RESULT)
5643  ;
5644                      .LOCAL
5645 128A 15DB A FLINT:   JSR     E2S
5646 128B D06D A          ST      AC0,TAC0
5647 128C DC70 A          ST      AC3,TAC3
5648 128D 6400 A          PULL    AC0
5649 128E 4101 A          BOC     ZRO,$FN
```

```
5650 128F 4209 A          BOC     POS,$F1
5651 1290 C06D A $FN:     LD      AC0,TAC0      ; EXPONENT <= 0
5652 1291 4203 A          BOC     POS,$F0
5653 1292 C0D8*A          LD      AC0,=0C000    ;    NEG NUM: RETURN -1
5654 1293 5101 A          LI      AC1,1
5655 1294 1902 A          JMP     $F3
5656 1295 5000 A $F0:     LI      AC0,0         ;    POS NUM: RETURN 0
5657 1296 5100 A          LI      AC1,0
5658 1297 CC70 A $F3:     LD      AC3,TAC3
5659 1298 8000 A          RTS     0
5660 1299 5F00 A $F1:     RCPY    AC0,AC3
5661 129A 9DE8*A          SKG     AC0,=23
5662 129B 1902 A          JMP     $F1A
5663 129C C06D A          LD      AC0,TAC0
5664 129D 1905 A          JMP     $F2           ; EXPN > 23 -- NO FRACTIONAL PAR
5665 129E C06D A $F1A:    LD      AC0,TAC0
5666 129F 7301 A          CAI     AC3,1         ; REMOVE FRACTIONAL PART (SHIFT
5667 12A0 ED34*A          ADD     AC3,=31       ;    THEN NORMALIZE)
5668 12A1 94C0*A          JSR     ADSHR
5669 12A2 531F A          LI      AC3,31        ; EXPONENT NOW 31
5670 12A3 6300 A $F2:     PUSH    AC3
5671 12A4 CC70 A          LD      AC3,TAC3
5672 12A5 98B9*A          JMP     IEXP
5673  ;
5674  ;        FLOATING POINT TO INTEGER (SINGLE WORD) CONVERSION
5675  ;                IFIX  : FLOOR FUNCTION (GREATEST INTEGER <= NUMBER)
5676  ;                ROUND : ROUND TO INTEGER
5677  ;
5678  ;                RETURN:  RTS 0   N < -32768  OR  N > 32767
5679  ;                         RTS 1   -32768 <= N <= 32767
5680  ;
5681 12A6 D07F A ROUND:  ST      AC0,SCRB      ; SAVE SIGN
5682 12A7 4201 A          BOC     POS,.+2       ; TAKE ABSOLUTE VALUE
5683 12A8 94BF*A          JSR     NEGATE
5684 12A9 C8DC*A          LD      AC2,=04000    ; ADD 0.5
5685 12AA 5300 A          LI      AC3,0
5686 12AB 94BC*A          JSR     FLADD
5687 12AC 1903 A          JMP     IFIX2
5688 12AD D07F A IFIX:    ST      AC0,SCRB      ; SAVE SIGN
5689 12AE 4201 A          BOC     POS,IFIX2     ; TAKE ABSOLUTE VALUE
5690 12AF 94BF*A          JSR     NEGATE
5691 12B0 15B5 A IFIX2:   JSR     E2S           ; GET EXPONENT
5692 12B1 D06D A          ST      AC0,TAC0
5693 12B2 DC70 A          ST      AC3,TAC3
5694 12B3 6400 A          PULL    AC0
5695 12B4 5F00 A          RCPY    AC0,AC3
5696 12B5 4BDF A          BOC     NEG,$F0
5697 12B6 9CD1*A          SKG     AC0,=15
5698 12B7 1913 A          JMP     $I5
5699 12B8 C07F A          LD      AC0,SCRB
5700 12B9 4209 A          BOC     POS,$I2
5701 12BA FD1B*A          SKNE    AC3,=16       ; -32768 IS NOT ERROR - CHECK
5702 12BB 1901 A          JMP     .+2
5703 12BC 1904 A          JMP     $I1
```

```
5704 12BD C06D A          LD      AC0,TAC0      ; CHECK FOR 32768
5705 12BE 9518*A          JSR     DSHL
5706 12BF F0D7*A          SKNE    AC0,=08000
5707 12C0 1908 A          JMP     $I4
5708 12C1 C4D7*A.$I1:     LD      AC1,=08000    ; RETURN -32768
5709 12C2 1901 A          JMP     $I3
5710 12C3 C4D6*A $I2:     LD      AC1,=07FFF    ; RETURN +32767
5711 12C4 CC70 A $I3:     LD      AC3,TAC3
5712 12C5 5034 A          LI      AC0,52        ; IFIX WARNING
5713 12C6 94C4*A          JSR     WARNING
5714 12C7 5C40 A          RCPY    AC1,AC0       ; NOW RETURN MAX INTEGER
5715 12C8 8000 A          RTS     0
5716 12C9 CC70 A $I4:     LD      AC3,TAC3      ; - 32768, NORMAL RETURN
5717 12CA 8001 A          RTS     1
5718 12CB 7301 A $I5:     CAI     AC3,1         ; SHIFT OUT FRACTIONAL PART
5719 12CC ED08*A          ADD     AC3,=31       ;    LEAVING ONLY INTEGER IN AC1
5720 12CD C06D A          LD      AC0,TAC0
5721 12CE 94C0*A          JSR     ADSHR
5722 12CF C07F A          LD      AC0,SCRB      ; CHECK SIGN FOR RESULT
5723 12D0 4201 A          BOC     POS,.+2
5724 12D1 7101 A          CAI     AC1,1
5725 12D2 5C40 A          RCPY    AC1,AC0       ; RETURN RESULT IN AC0
5726 12D3 CC70 A          LD      AC3,TAC3      ; RESTORE AC3 AND RETURN
5727 12D4 8001 A          RTS     1
5728  ;
5729 12D5 001F A          .POOL   4
     12D6 0010 A
     12D7 1219 T
     12D8 0000 A
5730  ;
5731  ;        RND(X): RANDOM NUMBER FUNCTION
5732  ;
5733  ;           THE ARGUMENT IS IGNORED AND MAY BE OMITTED
5734  ;
5735  ;           RNDNUM = (A * RNDNUM + C) MOD 65536
5736  ;           THEN 16 BIT POSITIVE NUMBER IS TREATED AS A FRACTION.
5737  ;
5738  ;           REFER TO KNUTH'S 'SEMINUMERICAL ALGORITHMS', CHAPTER 3.
5739  ;
5740 12D9 C086 A RND:      LD      AC0,RNDNUM    ; MULTIPLY RNDNUM BY A
5741 12DA C509 A          LD      AC1,A
5742 12DB 94B8*A          JSR     MULT
5743 12DC 5C40 A          RCPY    AC1,AC0
5744 12DD E107 A          ADD     AC0,C
5745 12DE D086 A          ST      AC0,RNDNUM    ; STORE NEW VALUE
5746 12DF 5100 A          LI      AC1,0
5747 12E0 3800 A          PFLG    LINK
5748 12E1 2403 A          ROR     AC0,1,1       ; SHIFT AC0 RIGHT 1 BIT INTO AC1
5749 12E2 2D03 A          SHR     AC1,1,1
5750 12E3 99F4*A          JMP     FLNORM
5751  ;
5752 12E4 0485 A A:       .WORD   1157
5753 12E5 3619 A C:       .WORD   13849
5754  ;  .
```

```
5755  ;
5756  ;       EXPONENTIAL SERIES ROUTINE
5757  ;
5758  ;       ON ENTRY: AC3 POINTS TO DATA AREA; AC0,AC1 HAS ARGUMENT
5759  ;       XX.    = X*X  (X IS THE ARGUMENT IN AC0,AC1)
5760  ;       EPSER = A0 + A2*XX + A4*XX**2 + A6*XX**3 + A8*XX**4 + ...
5761  ;             = ((((...+A8)*XX+A6)*XX+A4)*XX+A2)*XX+A0
5762  ;
5763                        .LOCAL
5764 12E6 CB00 A EPSER:  LD      AC2,(AC3)     ; SAVE COUNT OF COEFFICIENTS
5765 12E7 7AFF A         AISZ    AC2,-1
5766 12E8 D87A A         ST      AC2,COUNT
5767 12E9 6A80 A         RADD    AC2,AC2       ; ALAST = ADDR OF LAST COEFFICIE
5768 12EA 6AC0 A         RADD    AC3,AC2
5769 12EB 7A01 A         AISZ    AC2,1
5770 12EC D879 A         ST      AC2,ALAST
5771 12ED 5E00 A         RCPY    AC0,AC2       ; XX = X * X
5772 12EE 5F40 A         RCPY    AC1,AC3
5773 12EF 94BD*A         JSR     FLMULT
5774 12F0                DST     AC0,XX
5775 12F2 CC79 A         LD      AC3,ALAST     ; LOAD LAST COEFFICIENT
5776 12F3 7BFE A         AISZ    AC3,-2
5777 12F4 DC79 A         ST      AC3,ALAST
5778 12F5 C302 A         LD      AC0,2(AC3)    ; INITIAL VALUE = LAST COEFF
5779 12F6 C703 A         LD      AC1,3(AC3)
5780 12F7          $LOOP: DLD     AC2,XX        ; MULTIPLY LAST VALUE BY XX
5781 12F9 94BD*A         JSR     FLMULT
5782 12FA CC79 A         LD      AC3,ALAST     ; ADD NEXT COEFFICIENT
5783 12FB 7BFE A         AISZ    AC3,-2
5784 12FC DC79 A         ST      AC3,ALAST
5785 12FD CB02 A         LD      AC2,2(AC3)
5786 12FE CF03 A         LD      AC3,3(AC3)
5787 12FF 94BC*A         JSR     FLADD
5788 1300 AC7A A         DSZ     COUNT         ; CHECK IF DONE
5789 1301 19F5 A         JMP     $LOOP
5790 1302 8000 A         RTS     0
5791  ;
5792  ;       TRIG ROUTINES : SIN(X), COS(X), TAN(X)
5793  ;
5794 1303          TAN:  DST     AC0,SCR       ; TAN(X) = SIN(X)/COS(X)
5795 1305 150E A         JSR     COS
5796 1306                DST     AC0,SCRA
5797 1308                DLD     AC0,SCR
5798 130A 1506 A         JSR     SINE
5799 130B                DLD     AC2,SCRA
5800 130D 7A00 A         AISZ    AC2,0         ; NO DIVISION IF
5801 130E 98BE*A         JMP     FLDIV         ;    COS(X)=0
5802 130F 5D00 A         RCPY    AC0,AC1       ; OVERFLOW - SIGN OF SIN(X)
5803 1310 9940*A         JMP     OVFERR        ;    DETERMINES RESULT
5804  ;
5805 1311          SINE: DLD     AC2,NP2       ; SINE: ADD -PI/2 TO DATA
5806 1313 94BC*A         JSR     FLADD         ;    THEN TAKE COSINE
5807 1314          COS:  DLD     AC2,TWOPI     ; TAKE ARG MOD 2*PI
5808 1316 953B*A         JSR     FLMOD
```

```
5809 1317 CD3B*A          LD      AC3,=COSA     ; DO POWER SERIES NOW
5810 1318 19CD A          JMP     EPSER
5811  ;
5812 1319 6487 A TWOPI:   .WORD   06487,0ED03   ; 2 * PI
     131A ED03 A
5813 131B 9B78 A NP2:     .WORD   09B78,01301   ; - PI/2
     131C 1301 A
5814  ;
5815  ;        LN : LOG(X) = NATURAL LOG (BASE E)
5816  ;
5817  ;                LOGE(X) = LOGE(M) + B*LOGE(2) , WHERE X = M*2**B
5818  ;
5819 131D 411D A LN:       BOC     ZRO,ARGERR    ; CHECK OPERAND
5820 131E 4B1C A          BOC     NEG,ARGERR
5821 131F F0DC A          SKNE    AC0,FL1       ; LOGE(1) IS 0
5822 1320 1901 A          JMP     .+2
5823 1321 1902 A          JMP     $1A
5824 1322 F4DD A          SKNE    AC1,FL1+1
5825 1323 192A A          JMP     RSLT0
5826 1324 952F*A $1A:     JSR     E2S           ; REMOVE AND SAVE EXPONENT
5827 1325 6600 A          PULL    AC2
5828 1326 D87F A          ST      AC2,SCRB
5829 1327 1517 A          JSR     XOVER         ; CALCULATE (X-1)/(X+1)
5830 1328                 DST     AC0,SCR
5831 132A CD2A*A          LD      AC3,=LNDATA   ; DO POWER SERIES
5832 132B 15BA A          JSR     EPSER
5833 132C                 DLD     AC2,SCR       ; MULTIPLY BY (X-1)/(X+1)
5834 132E 94BD*A          JSR     FLMULT
5835 132F 5301 A          LI      AC3,1         ; MULTIPLY BY 2 (ADD 1 TO EXPONE
5836 1330 9525*A          JSR     PWR2
5837 1331                 DST     AC0,SCR       ; SAVE LOGE(M)
5838 1333 C07F A          LD      AC0,SCRB      ; TAKE EXPONENT * LOGE(2)
5839 1334 94B6*A          JSR     INTFL
5840 1335                 DLD     AC2,LN2
5841 1337 94BD*A          JSR     FLMULT
5842 1338                 DLD     AC2,SCR       ; NOW ADD 2 PARTS
5843 133A 98BC*A          JMP     FLADD
5844 133B 502A A ARGERR:  LI      AC0,42        ; ARGUMENT OUT OF RANGE FOR FUNC
5845 133C 98C5*A          JMP     RUNERR
5846  ;
5847 133D 58B9 A LN2:     .WORD   058B9,0A00    ; LN(2)
     133E 0A00 A
5848  ;
5849 133F         XOVER:  DST     AC0,SCRA      ; CALCULATE (X-1)/(X+1)
5850 1341                 DLD     AC2,FL1       ;      FOR LN, ARCTAN
5851 1343 94BC*A          JSR     FLADD
5852 1344                 DLD     AC2,SCRA      ; EXCHANGE (X+1) AND (X) IN SCRA
5853 1346                 DST     AC0,SCRA
5854 1348 C0D8*A          LD      AC0,=0C000    ; ADD -1
5855 1349 5101 A          LI      AC1,1
5856 134A 94BC*A          JSR     FLADD
5857 134B                 DLD     AC2,SCRA
5858 134D 98BE*A          JMP     FLDIV
5859  ;
```

```
5860 134E 5000 A RSLTO:  LI      AC0,0        ; ZERO RESULT: SQR(0), LOG(1)
5861 134F 5100 A         LI      AC1,0
5862 1350 8000 A         RTS     0
5863 1351 1276 T         .POOL   6
     1352 11FA T
     1353 1523 T
     1354 1266 T
     1355 153A T
     1356 123A T
5864  ;
5865  ;        EXPON : EXP(X) = EXPONENTIAL FUNCTION
5866  ;
5867 1357        EXPON:  DLD     AC2,L2E      ; XX = X*LOG2(E)
5868 1359 94BD*A         JSR     FLMULT
5869 135A                DST     AC0,XX
5870 135C 94B4*A         JSR     IFIX         ; SCRB = ENTIER(XX)
5871 135D 5C00 A         NOP
5872 135E D07F A         ST      AC0,SCRB
5873 135F 9D7D*A         SKG     AC0,=-127    ; < -126 ?
5874 1360 19ED A         JMP     RSLTO        ;    YES - ZERO RESULT
5875 1361 9D7C*A         SKG     AC0,=126     ; > 126 ?
5876 1362 1902 A         JMP     $EXP1
5877 1363 5100 A         LI      AC1,0        ;    YES - OVERFLOW
5878 1364 99EC*A         JMP     OVFERR
5879 1365 94B6*A $EXP1:  JSR     INTFL
5880 1366                DST     AC0,SCR      ; SCR = FLOAT(SCRB)
5881 1368 94BF*A         JSR     NEGATE       ; XX = XX-SCR
5882 1369                DLD     AC2,XX
5883 136B 94BC*A         JSR     FLADD
5884 136C                DST     AC0,XX
5885 136E 5E00 A         RCPY    AC0,AC2      ; SCRA = XX*XX
5886 136F 5F40 A         RCPY    AC1,AC3
5887 1370 94BD*A         JSR     FLMULT
5888 1371                DST     AC0,SCRA
5889 1373                DLD     AC2,EXPA     ; SCR = SCRA+EXPA
5890 1375 94BC*A         JSR     FLADD
5891 1376                DST     AC0,SCR
5892 1378 5E00 A         RCPY    AC0,AC2      ; SCR = EXPB/SCR
5893 1379 5F40 A         RCPY    AC1,AC3
5894 137A                DLD     AC0,EXPB
5895 137C 94BE*A         JSR     FLDIV
5896 137D                DST     AC0,SCR
5897 137F                DLD     AC0,EXPC     ; SCR = EXPC*SCRA+EXPD-XX-SCR
5898 1381                DLD     AC2,SCRA
5899 1383 94BD*A         JSR     FLMULT
5900 1384                DLD     AC2,EXPD
5901 1386 94BC*A         JSR     FLADD
5902 1387                DLD     AC2,XX
5903 1389 9555*A         JSR     FLSUB
5904 138A                DLD     AC2,SCR
5905 138C 9552*A         JSR     FLSUB
5906 138D 5E00 A         RCPY    AC0,AC2      ; ANSWER = ((XX/SCR)+0.5)
5907 138E 5F40 A         RCPY    AC1,AC3      ;    * 2**(SCRB+1)
5908 138F                DLD     AC0,XX
```

```
5909 1391 94BE*A          JSR     FLDIV
5910 1392 C8DC*A          LD      AC2,=04000    ; ADD 0.5
5911 1393 5300 A          LI      AC3,0
5912 1394 94BC*A          JSR     FLADD
5913 1395 CC7F A.         LD      AC3,SCRB
5914 1396 ECDD*A          ADD     AC3,=1
5915 1397 99BE*A          JMP     PWR2
5916  ;
5917 1398 576A A EXPA:    .WORD   0576A,0E107   ; 87.417497
     1399 E107 A
5918 139A 4D3F A EXPB:    .WORD   04D3F,01D0A   ; 617.9723
     139B 1D0A A
5919 139C 46FA A EXPC:    .WORD   046FA,074FC   ; .03465739
     139D 74FC A
5920 139E 4FAC A EXPD:    .WORD   04FAC,0304    ; 9.954596
     139F 0304 A
5921 13A0 5C55 A L2E:     .WORD   05C55,01D01   ; 1.442695 = LOG2(E)
     13A1 1D01 A
5922  ;
5923  ;        SQUARE ROOT : SQR(X)
5924  ;
5925 13A2 41AB A SQRT:    BOC     ZRO,RSLTO     ; CHECK ARGUMENT
5926 13A3 4B97 A          BOC     NEG,ARGERR
5927 13A4 95AF*A          JSR     E2S           ; TEST EXPONENT FIELD
5928 13A5 6400 A          PULL    AC0
5929 13A6 D45D A          ST      AC1,OP1+1     ; OP1 = MANTISSA
5930 13A7 5301 A          LI      AC3,1
5931 13A8 430B A          BOC     BIT0,SQODD    ; TEST IF EXP IS EVEN OR ODD
5932 13A9 94C0*A          JSR     ADSHR         ; EVEN: OP2=(E/2)-1
5933 13AA E0DB*A          ADD     AC0,=-1
5934 13AB D05E A          ST      AC0,OP2
5935 13AC                 DLD     AC0,OP1       ; SCR = SB2+SA2*OP1
5936 13AE                 DLD     AC2,SA2
5937 13B0 94BD*A          JSR     FLMULT
5938 13B1 C929 A          LD      AC2,SB2
5939 13B2 CD29 A          LD      AC3,SB2+1
5940 13B3 190D A          JMP     SQ2
5941 13B4 94C0*A SQODD:   JSR     ADSHR         ; ODD EXPONENT
5942 13B5 D05E A          ST      AC0,OP2       ; OP2 = E/2
5943 13B6                 DLD     AC0,OP1       ; OP1=OP1/2
5944 13B8 53FF A          LI      AC3,-1
5945 13B9 959C*A          JSR     PWR2
5946 13BA                 DST     AC0,OP1
5947 13BC                 DLD     AC2,SA1       ; SCR = SB1+SA1*OP1
5948 13BE 94BD*A          JSR     FLMULT
5949 13BF                 DLD     AC2,SB1
5950 13C1 94BC*A SQ2:     JSR     FLADD
5951 13C2 1508 A          JSR     NWTN          ; SCR = NWTN(SCR)/2
5952 13C3 53FF A          LI      AC3,-1
5953 13C4 9591*A          JSR     PWR2
5954 13C5 1505 A          JSR     NWTN          ; ITERATE AGAIN
5955 13C6 53FF A          LI      AC3,-1
5956 13C7 958E*A          JSR     PWR2
5957 13C8 1502 A          JSR     NWTN          ; RESULT = NWTN(SCR) * 2**OP2
```

```
5958 13C9 CC5E A          LD      AC3,OP2
5959 13CA 998B*A          JMP     PWR2
5960  ;
5961 13CB         NWTN:   DST     AC0,SCR       ; SCR = (AC0,AC1)
5962 13CD 5E00 A          RCPY    AC0,AC2       ; (AC0,AC1)=(SCR+OP1/SCR)
5963 13CE 5F40 A          RCPY    AC1,AC3
5964 13CF                 DLD     AC0,OP1
5965 13D1 94BE*A          JSR     FLDIV
5966 13D2                 DLD     AC2,SCR
5967 13D4 98BC*A          JMP     FLADD
5968  ;
5969 13D5 7000 A SA1:     .WORD   07000,00000   ; 0.875
     13D6 0000 A
5970 13D7 4754 A SB1:     .WORD   04754,04BFF   ; 0.27863
     13D8 4BFF A
5971 13D9 4A00 A SA2:     .WORD   04A00,00000   ; 0.578125
     13DA 0000 A
5972 13DB 6C00 A SB2:     .WORD   06C00,000FF   ; 0.421875
     13DC 00FF A
5973 13DD FF81 A          .POOL   3
     13DE 007E A
     13DF 11C5 T
5974  ;
5975  ;
5976  ;         POWER : (OP1) ** (OP2)
5977  ;
5978  ;         THIS ROUTINE IS CALLED ONLY BY THE EXPRESSION ANALYZER;
5979  ;         THEREFORE THE ARGUMENTS ARE IN BOTH THE REGISTERS AND
5980  ;         IN BASE PAGE.
5981  ;
5982  ;         IF THE POWER IS A POSITIVE INTEGER, THEN THE CALCULATION
5983  ;         IS BY REPEATED MULTIPLICATION; OTHERWISE A SERIES IS USED.
5984  ;
5985 13E0 F0CC*A POWER:  SKNE    AC0,=0        ; OP1=0 -- 0 RESULT
5986 13E1 995C*A          JMP     RSLT0
5987 13E2 F8CC*A          SKNE    AC2,=0        ; OP2=0 -- 1 RESULT
5988 13E3 191C A          JMP     RSLT1
5989 13E4 5C80 A          RCPY    AC2,AC0       ; USE SERIES IF EXPONENT IS NEGA
5990 13E5 4B07 A          BOC     NEG,$PSER
5991 13E6 5DC0 A          RCPY    AC3,AC1
5992 13E7 94B7*A          JSR     FLINT         ; IS THE POWER AN INTEGER?
5993 13E8 F05E A          SKNE    AC0,OP2
5994 13E9 1901 A          JMP     .+2
5995 13EA 1902 A          JMP     $PSER         ;    NO
5996 13EB F45F A          SKNE    AC1,OP2+1
5997 13EC 1907 A          JMP     $PMULT        ;    YES
5998 13ED         $PSER:  DLD     AC0,OP1       ; X**Y = EXP(Y*LN(X))
5999 13EF 954F*A          JSR     LN
6000 13F0                 DLD     AC2,OP2
6001 13F2 94BD*A          JSR     FLMULT
6002 13F3 994C*A          JMP     EXPON
6003 13F4 94B4*A $PMULT:  JSR     IFIX          ; CONVERT POWER TO INTEGER FORMA
6004 13F5 994B*A          JMP     ARGERR        ; ... TOO BIG
6005 13F6 D07A A          ST      AC0,COUNT.
```

```
6006 13F7                    DLD     AC0,OP1
6007 13F9 AC7A A $PM:        DSZ     COUNT          ; MULTIPLY (COUNT) TIMES
6008 13FA 1901 A             JMP     .+2
6009 13FB 8000 A             RTS     0
6010 13FC                    DLD     AC2,OP1
6011 13FE 94BD*A             JSR     FLMULT
6012 13FF 19F9 A             JMP     $PM
6013 1400         RSLT1:     DLD     AC0,FL1        ; RETURN 1 RESULT
6014 1402 8000 A             RTS     0
6015  ;
6016  ;        ARCTAN : ARC TANGENT
6017  ;
6018 1403 5201 A ARCTAN:     LI      AC2,1          ; TAKE ABS(X), SAVING SIGN FLAG
6019 1404 4202 A             BOC     POS,ATN0
6020 1405 94BF*A             JSR     NEGATE
6021 1406 52FF A             LI      AC2,-1
6022 1407         ATN0:      DST     AC0,SCR
6023 1409 D84E A             ST      AC2,ESIGN
6024 140A                    DLD     AC2,AY1        ; TEST RANGE OF ARGUMENT
6025 140C 94BB*A             JSR     FLCMP          ; X <= (SQR(2)-1) ?
6026 140D 4116 A             BOC     ZRO,ATN1
6027 140E 4B15 A             BOC     NEG,ATN1
6028 140F                    DLD     AC0,SCR
6029 1411                    DLD     AC2,AY2
6030 1413 CD1E A             LD      AC3,AY2+1
6031 1414 94BB*A             JSR     FLCMP          ; X <= (SQR(2)+1) ?
6032 1415 4112 A             BOC     ZRO,ATN2
6033 1416 4B11 A             BOC     NEG,ATN2
6034 1417                    DLD     AC0,FL1        ; X > (SQR(2)+1) :
6035 1419                    DLD     AC2,SCR        ; ATN(X) = (PI/2)
6036 141B 94BE*A             JSR     FLDIV          ;    - ATN(1/X)
6037 141C 151A A             JSR     ATAN
6038 141D 94BF*A             JSR     NEGATE
6039 141E                    DLD     AC2,P2
6040 1420 94BC*A ATN3:       JSR     FLADD
6041 1421 AC4E A ATN4:       DSZ     ESIGN          ; NEGATE RESULT IF ORIGINAL SIGN
6042 1422 94BF*A             JSR     NEGATE         ;    WAS NEGATIVE
6043 1423 8000 A             RTS     0
6044 1424         ATN1:      DLD     AC0,SCR        ; X <= (SQR(2)-1) :
6045 1426 1510 A             JSR     ATAN           ; USE SERIES AS IS
6046 1427 19F9 A             JMP     ATN4
6047 1428 C07B A ATN2:       LD      AC0,SCR        ; (SQR(2)-1) <= X
6048 1429 C47C A             LD      AC1,SCR+1      ;    <= (SQR(2)+1) :
6049 142A 9517*A             JSR     XOVER          ; ATN(X) = (PI/4)
6050 142B 150B A             JSR     ATAN           ;    + ATN((X-1)/(X+1))
6051 142C                    DLD     AC2,P4
6052 142E 19F1 A             JMP     ATN3
6053  ;
6054 142F 6A09 A AY1:        .WORD   06A09,0E4FF    ; SQR(2)-1
     1430 E4FF A
6055 1431 4D41 A AY2:        .WORD   04D41,03C02    ; SQR(2)+1
     1432 3C02 A
6056 1433 6487 A P2:         .WORD   06487,0ED01    ; PI/2
     1434 ED01 A
```

```
6057 1435 6487 A P4:      .WORD   06487,0ED00  ; PI/4
     1436 ED00 A
6058  ;
6059 1437         ATAN:   DST     AC0,SCRA     ; DO SERIES FOR ATN FUNCTION
6060 1439 CD09*A.         LD      AC3,=ATDATA
6061 143A 9509*A          JSR     EPSER
6062 143B                 DLD     AC2,SCRA     ; (ODD POWERS)
6063 143D 98BD*A          JMP     FLMULT
6064  ;
6065 143E 134E T          .POOL   7
     143F 131D T
     1440 1357 T
     1441 133B T
     1442 133F T
     1443 1563 T
     1444 12E6 T
```

```
6066                        .PAGE   'FLOATING POINT TO ASCII CONVERSION'
6067                        .LOCAL
6068  ;
6069  ;         FLDEC : CONVERT NUMBER IN AC0,AC1 TO ASCII
6070  ;
6071 1445 9499*A FLDEC:   JSR     SAVE            ; SAVE REGISTERS
6072 1446 5220 A         LI      AC2,BLANK       ; CHECK SIGN
6073 1447 4202 A         BOC     POS,$P
6074 1448 522D A         LI      AC2,'-'/256
6075 1449 94BF*A         JSR     NEGATE
6076 144A          $P:   DST     AC0,VALUE       ; SAVE ABSOLUTE VALUE
6077 144C 5C80 A         RCPY    AC2,AC0         ; PRINT SIGN CHARACTER
6078 144D 949E*A         JSR     OUT1
6079 144E                DLD     AC0,VALUE       ; TEST VALUE OF NUMBER
6080 1450 4504 A         BOC     NZRO,$NORM
6081 1451 5030 A         LI      AC0,'0'/256     ; IF ZERO, PUT OUT 1 DIGIT
6082 1452 949E*A         JSR     OUT1
6083 1453 9900 A         JMP     @.+1
6084 1454 14E9 T         .WORD   $DONE
6085 1455 5200 A $NORM:  LI      AC2,0           ; NORMALIZE TO RANGE
6086 1456 D849 A         ST      AC2,EXPN        ;     1,000,000 TO 9,999,999
6087 1457          $N1:  DLD     AC0,VALUE       ; DIVIDE BY 10 IF >= 10**7
6088 1459                DLD     AC2,PWR7
6089 145B 94BB*A         JSR     FLCMP
6090 145C 4201 A         BOC     POS,.+2         ; IF AC0 >= 0 THEN DIVIDE
6091 145D 1905 A         JMP     $N2
6092 145E 5001 A         LI      AC0,1
6093 145F                DLD     AC2,TENTH
6094 1461 150C A         JSR     SCALE
6095 1462 19F4 A         JMP     $N1
6096 1463          $N2:  DLD     AC0,VALUE       ; MULT BY 10 IF < 10**6
6097 1465                DLD     AC2,PWR6
6098 1467 94BB*A         JSR     FLCMP
6099 1468 4215 A         BOC     POS,$2BCD
6100 1469 50FF A         LI      AC0,-1
6101 146A                DLD     AC2,FL10
6102 146C 1501 A         JSR     SCALE
6103 146D 19F5 A         JMP     $N2
6104  ;
6105 146E E049 A SCALE:  ADD     AC0,EXPN        ; ADD (AC0) TO EXPONENT
6106 146F D049 A         ST      AC0,EXPN
6107 1470                DLD     AC0,VALUE       ; MULTIPLY VALUE BY (AC2,AC3)
6108 1472 94BD*A         JSR     FLMULT
6109 1473                DST     AC0,VALUE
6110 1475 8000 A         RTS     0
6111  ;
6112 1476 7A12 A PWR6:   .WORD   07A12,00014     ; 10**6
     1477 0014 A
6113 1478 4C4B A PWR7:   .WORD   04C4B,04018     ; 10**7
     1479 4018 A
6114 147A 5000 A FL10:   .WORD   05000,00004     ; 10
     147B 0004 A
6115 147C 6666 A TENTH:  .WORD   06666,066FD     ; 0.1
     147D 66FD A
```

```
6116  ;
6117 147E 506B A $2BCD:  LI    AC0,TEMP+6   ; CONVERT TO BCD - 7 DIGITS
6118 147F D079 A         ST    AC0,ALAST    ;    IN TEMP ... TEMP+6
6119 1480 5007 A         LI    AC0,7
6120 1481 D07A A         ST    AC0,COUNT
6121 1482        MODX:   DLD   AC0,VALUE    ; SAVE OLD VALUE
6122 1484                DST   AC0,SCRA
6123 1486                DLD   AC2,FL10     ; NEW VALUE = INT (VALUE/10)
6124 1488 94BE*A         JSR   FLDIV
6125 1489 94B7*A         JSR   FLINT
6126 148A                DST   AC0,VALUE
6127 148C                DLD   AC2,FL10     ; MOD = OLD VALUE
6128 148E 94BD*A         JSR   FLMULT       ;       - NEW VALUE * 10
6129 148F 94BF*A         JSR   NEGATE
6130 1490                DLD   AC2,SCRA
6131 1492 94BC*A         JSR   FLADD
6132 1493 94B4*A         JSR   IFIX
6133 1494 5C00 A         NOP
6134 1495 B079 A         ST    AC0,@ALAST   ; STORE DIGIT
6135 1496 AC79 A         DSZ   ALAST        ; DECREMENT ADDRESS FOR NEXT
6136 1497 AC7A A         DSZ   COUNT        ; DECREMENT DIGIT COUNT
6137 1498 19E9 A         JMP   MODX         ;    NOT DONE YET
6138 1499 536B A         LI    AC3,TEMP+6   ; ROUND 7 DIGITS TO 6
6139 149A 5107 A         LI    AC1,7        ;    DIGITS
6140 149B C300 A         LD    AC0,(AC3)    ; LEAST SIGNIF GETS 5 ADDED,
6141 149C 7804 A         AISZ  AC0,4        ;    THEN BCD CARRY IS CHECKED
6142 149D D300 A         ST    AC0,(AC3)
6143 149E C300 A $ROUND: LD    AC0,(AC3)    ; INCR DIGIT & CHECK CARRY
6144 149F 7801 A         AISZ  AC0,1
6145 14A0 D300 A         ST    AC0,(AC3)
6146 14A1 9CCE*A         SKG   AC0,=9       ; CARRY NEEDED?
6147 14A2 1905 A         JMP   $CHK1        ;    NO - ROUNDING DONE
6148 14A3 5000 A         LI    AC0,0        ; DIGIT IS ZERO
6149 14A4 D300 A         ST    AC0,(AC3)
6150 14A5 7BFF A         AISZ  AC3,-1       ; GO TO NEXT DIGIT
6151 14A6 79FF A         AISZ  AC1,-1
6152 14A7 19F6 A         JMP   $ROUND
6153 14A8 536A A $CHK1:  LI    AC3,TEMP+5   ; CHECK NUMBER OF SIGNIFICANT
6154 14A9 5106 A         LI    AC1,6        ;    DIGITS (6 MAX)
6155 14AA C300 A $CHKD:  LD    AC0,(AC3)
6156 14AB 4507 A         BOC   NZRO,$PRT
6157 14AC 7BFF A         AISZ  AC3,-1
6158 14AD 79FF A         AISZ  AC1,-1
6159 14AE 19FB A         JMP   $CHKD
6160 14AF 5101 A         LI    AC1,1        ; ZERO COUNT: MANTISSA IS 1,
6161 14B0 D465 A         ST    AC1,TEMP     ;    EXPONENT IS INCREMENTED
6162 14B1 8C49 A         ISZ   EXPN
6163 14B2 5C00 A         NOP
6164 14B3 D47A A $PRT:   ST    AC1,COUNT    ; SAVE COUNT
6165 14B4 C049 A         LD    AC0,EXPN     ; FIX PROPER EXPONENT
6166 14B5 E13E*A         ADD   AC0,=6
6167 14B6 D049 A         ST    AC0,EXPN
6168 14B7 5365 A         LI    AC3,TEMP
6169 14B8 9D3C*A         SKG   AC0,=5       ; CHECK IF E-FORMAT IS NEEDED
```

```
6170 14B9 9D3C*A          SKG     AC0,=-3
6171 14BA 1912 A          JMP     $ETYPE
6172 14BB E0DD*A          ADD     AC0,=1          ; ** FIXED FORMAT **
6173 14BC 9C7A A          SKG     AC0,COUNT       ; IF (EXPN+1) > COUNT, FIX COUNT
6174 14BD 1901 A          JMP     .+2             ;    SO ALL DIGITS ARE PRINTED
6175 14BE D07A A          ST      AC0,COUNT
6176 14BF C07A A          LD      AC0,COUNT
6177 14C0 51FE A          LI      AC1,-2
6178 14C1 F449 A          SKNE    AC1,EXPN
6179 14C2 1529 A          JSR     $PTO            ; PRINT '.0' IF NEEDED
6180 14C3 51FF A          LI      AC1,-1
6181 14C4 F449 A $PDIG:   SKNE    AC1,EXPN        ; PRINT DIGITS: FIRST CHECK FOR
6182 14C5 152A A          JSR     $DEC            ;    '.' NEEDED
6183 14C6 C300 A          LD      AC0,(AC3)
6184 14C7 152A A          JSR     PUTD
6185 14C8 ECDD*A          ADD     AC3,=1
6186 14C9 E4DD*A          ADD     AC1,=1
6187 14CA AC7A A          DSZ     COUNT
6188 14CB 19F8 A          JMP     $PDIG
6189 14CC 191C A          JMP     $DONE
6190 14CD C300 A $ETYPE:  LD      AC0,(AC3)       ; E-FORMAT: D.DDDDDE-XX
6191 14CF 1523 A          JSR     PUTD
6192 14CF FD27*A          SKNE    AC3,=TEMP
6193 14D0 151F A          JSR     $DEC
6194 14D1 7B01 A          AISZ    AC3,1
6195 14D2 AC7A A          DSZ     COUNT
6196 14D3 19F9 A          JMP     $ETYPE
6197 14D4 5045 A          LI      AC0,'E'/256     ; PRINT EXPONENT FIELD
6198 14D5 949E*A          JSR     OUT1
6199 14D6 C049 A          LD      AC0,EXPN        ; PREPARE EXPONENT
6200 14D7 512B A          LI      AC1,'+'/256
6201 14D8 4202 A          BOC     POS,.+3
6202 14D9 512D A          LI      AC1,'-'/256
6203 14DA 7001 A          CAI     AC0,1
6204 14DB D049 A          ST      AC0,EXPN        ; EXPN NOW HAS ABS VALUE
6205 14DC 5C40 A          RCPY    AC1,AC0
6206 14DD 949E*A          JSR     OUT1            ; PRINT SIGN
6207 14DE C049 A          LD      AC0,EXPN        ; PRINT 2 EXPONENT DIGITS
6208 14DF 5100 A          LI      AC1,0
6209 14E0 9CCE*A $1:      SKG     AC0,=9
6210 14E1 1903 A          JMP     $2
6211 14E2 E115*A          ADD     AC0,=-10
6212 14E3 7901 A          AISZ    AC1,1
6213 14E4 19FB A          JMP     $1
6214 14E5 6D00 A $2:      RXCH    AC0,AC1
6215 14E6 150B A          JSR     PUTD
6216 14E7 5C40 A          RCPY    AC1,AC0
6217 14E8 1509 A          JSR     PUTD
6218 14E9 5020 A $DONE:   LI      AC0,BLANK       ; BLANK FOLLOWS NUMBER
6219 14EA 949E*A          JSR     OUT1
6220 14EB 989A*A          JMP     RESTOR          ; RESTORE REG & RETURN
6221  ;
6222 14EC 502E A $PTO:    LI      AC0,'.'/256
6223 14ED 949E*A          JSR     OUT1
```

```
6224 14EE 5030 A          LI      AC0,'0'/256
6225 14EF 989E*A          JMP     OUT1
6226 14F0 502E A $DEC:    LI      AC0,'. '/256
6227 14F1 989E*A          JMP     OUT1
6228 14F2 A506*A.PUTD:    OR      AC0,=030      ; CONVERT BINARY TO ASCII
6229 14F3 989E*A          JMP     OUT1
6230  ;
6231 14F4 0006 A          .POOL   8
     14F5 0005 A
     14F6 FFFD A
     14F7 0065 A
     14F8 FFF6 A
     14F9 0030 A
     14FA 0000 A
     14FB 0000 A
6232  ;
```

```
6233                        .PAGE   'INTEGER MATH ROUTINES'
6234                        .LOCAL
6235  ;
6236  ;         MULT    : SIGNED INTEGER MULTIPLY
6237  ;         FRMULT : SINGLE FRACTIONAL MULTIPLY
6238  ;         (AC0,AC1) := (AC0) * (AC1)
6239  ;
6240 14FC 1501 A FRMULT: JSR     MULT          ; MULTIPLY INTEGERS
6241 14FD 99FC*A         JMP     DSHL          ; CORRECT TO FRACTIONAL
6242 14FE D86F A MULT:   ST      AC2,TAC2
6243 14FF DC70 A         ST      AC3,TAC3
6244 1500 5200 A         LI      AC2,0         ; CLEAR AC2
6245 1501 D876 A         ST      AC2,FRSIGN    ; FRSIGN=0 IF RESULT IS POSITIVE
6246 1502 4202 A         BOC     POS,$M1       ; TAKE ABSOLUTE VALUES OF BOTH O
6247 1503 8C76 A         ISZ     FRSIGN
6248 1504 7001 A         CAI     AC0,1
6249 1505 6D00 A $M1:    RXCH    AC0,AC1
6250 1506 4203 A         BOC     POS,$M2
6251 1507 7001 A         CAI     AC0,1
6252 1508 AC76 A         DSZ     FRSIGN
6253 1509 5C00 A         NOP
6254 150A 5310 A $M2:    LI      AC3,16        ; BIT COUNT=16
6255 150B 7000 A         CAI     AC0,0         ; COMPLEMENT AC0 TO SIMPLIFY
6256                                           ; BRANCHING ON MULTIPLIER BIT0
6257 150C 3800 A $LOOP:  PFLG    LINK
6258 150D 4301 A         BOC     BIT0,.+2      ; BRANCH IF AC0 COMPLEMENTED=0
6259 150E 6A40 A         RADD    AC1,AC2       ; AC1+AC2 --> AC2
6260 150F 2603 A         ROR     AC2,1,1       ; ROTATE RESULT OF ADD INTO LINK
6261 1510 2C03 A         SHR     AC0,1,1       ; SHIFT LINK INTO AC0
6262 1511 7BFF A         AISZ    AC3,-1        ; DECR COUNT, SKIP IF ZERO
6263 1512 19F9 A         JMP     $LOOP
6264 1513 5D00 A         RCPY    AC0,AC1       ; MOVE LO ORDER RESULT TO AC1
6265 1514 5C80 A         RCPY    AC2,AC0       ; MOVE HI ORDER RESULT TO AC0
6266 1515 C876 A         LD      AC2,FRSIGN    ; TEST SIGN OF RESULT
6267 1516 7A00 A         AISZ    AC2,0
6268 1517 95E3*A         JSR     D2C           ; COMPLEMENT FOR NEGATIVE RESULT
6269 1518 C86F A         LD      AC2,TAC2      ; DONE - RESTORE REG
6270 1519 CC70 A         LD      AC3,TAC3
6271 151A 8000 A         RTS     0
6272  ;
6273  ;         BCDADD  : BCD ADDITION ROUTINE
6274  ;         (AC1) = (AC2) + (AC3)
6275  ;
6276  ;                 RTS 0 = OVERFLOW
6277  ;                 RTS 1 = NORMAL RETURN
6278  ;
6279 151B DC70 A BCDADD: ST      AC3,TAC3
6280 151C 5C80 A         RCPY    AC2,AC0
6281 151D 3700 A         PFLG    CARRY
6282 151E 8870 A         DECA    AC0,TAC3
6283 151F 5D00 A         RCPY    AC0,AC1
6284 1520 4A01 A         BOC     CRY,.+2       ; DONE - CHECK CARRY
6285 1521 8001 A         RTS     1
6286 1522 8000 A         RTS     0
```

```
6287  ;
```

```
6288                        .PAGE   'MATH PACKAGE DATA'
6289      8001 A            .LIST   LOPT
6290  ;
6291  ;         DATA FOR POWER SERIES APPROXIMATIONS
6292  ;
6293 1523 000B A COSA:      .WORD   11
6294                        ;   1    -1/2!   1/4!
6295 1524 4000 A            .WORD   04000,00001,0C000,00000,05555,055FC
6296                        ; -1/6!  1/8!  -1/10!
6297 152A A4FA A            .WORD   0A4FA,050F7,06806,080F1,0B606,0C2EB
6298                        ;   1/12! -1/14!  1/16!
6299 1530 47BB A            .WORD   047BB,063E4,09B1A,02BDC,06B9F,0D3D4
6300                        ; -1/18!  1/20!
6301 1536 A5F6 A            .WORD   0A5F6,01BCC,07950,0B3C3
6302  ;
6303 153A 0014 A LNDATA:    .WORD   20
6304                        ; 1     1/3   1/5
6305 153B 4000 A            .WORD   04000,00001,05555,055FF,06666,066FE
6306                        ; 1/7   1/9   1/11
6307 1541 4924 A            .WORD   04924,092FE,071C7,01CFD,05D17,045FD
6308                        ; 1/13  1/15  1/17
6309 1547 4EC4 A            .WORD   04EC4,0ECFD,04444,044FD,07878,078FC
6310                        ; 1/19  1/21  1/23
6311 154D 6BCA A            .WORD   06BCA,01AFC,06186,018FC,0590B,021FC
6312                        ; 1/25  1/27  1/29
6313 1553 51EB A            .WORD   051EB,085FC,04BDA,012FC,0469E,0E5FC
6314                        ; 1/31  1/33  1/35
6315 1559 4210 A            .WORD   04210,084FC,07C1F,007FB,07507,050FB
6316                        ; 1/37  1/39
6317 155F 6EB3 A            .WORD   06EB3,0E4FB,06906,090FB
6318  ;
6319 1563 000F A ATDATA:    .WORD   15
6320                        ;   1     -1/3    1/5
6321 1564 4000 A            .WORD   04000,00001,0AAAA,0ABFF,06666,066FE
6322                        ; -1/7    1/9   -1/11
6323 156A B6DB A            .WORD   0B6DB,06EFE,071C7,01CFD,0A2E8,0BBFD
6324                        ;   1/13 -1/15   1/17
6325 1570 4EC4 A            .WORD   04EC4,0EDFD,0BBBB,0BCFD,07878,078FC
6326                        ; -1/19   1/21  -1/23
6327 1576 9435 A            .WORD   09435,0E6FC,06186,018FC,0A6F4,0DFFC
6328                        ;   1/25 -1/27   1/29
6329 157C 51EB A            .WORD   051EB,085FC,0B425,0EEFC,0469E,0E5FC
6330  ;
```

```
6331                            .PAGE   'STATE/ROUTINE ADDRESS TABLES'
6332        8001 A              .LIST   LOPT
6333  ;
6334  ;           STATE ADDRESS TABLE
6335  ;
6336 1582 095C T TAB1:      .WORD   SYNTAB,ST01,ST02,ST03,ST04
6337 1587 09D9 T            .WORD   ST05,ST06,ST07,ST08,ST09
6338 158C 09F3 T            .WORD   ST10,ST11,ST12,ST13,ST14
6339 1591 0A14 T            .WORD   ST15,ST16,ST17,ST18,ST19
6340 1596 0A1E T            .WORD   ST20,ST21,ST22,ST23,ST24
6341 159B 0A34 T            .WORD   ST25,ST26,ST27,ST28,ST29
6342 15A0 0A48 T            .WORD   ST30,ST31,ST32,ST33,ST34
6343 15A5 0A77 T            .WORD   ST35,ST36,ST37,ST38,ST39
6344 15AA 0A94 T            .WORD   ST40,ST41,ST42,ST43,ST44
6345 15AF 0AA3 T            .WORD   ST45,ST46,ST47,ST48,ST49
6346 15B4 0ABE T            .WORD   ST50,ST51,ST52,ST53,ST54
6347 15B9 0ADB T            .WORD   ST55,ST56,ST57,ST58,ST59
6348 15BE 0AF9 T            .WORD   ST60,ST61,ST62,ST63,ST64
6349 15C3 0B14 T            .WORD   ST65,ST66,ST67,ST68,ST69
6350 15C8 0B50 T            .WORD   ST70,ST71,ST72,ST73,ST74
6351 15CD 0B43 T            .WORD   ST75,ST76,ST77,ST78,ST79
6352 15D2 0B52 T            .WORD   ST80,ST81,ST82,ST83,ST84
6353 15D7 0B69 T            .WORD   ST85,ST86,ST87,ST88,ST89
6354 15DC 0B7E T            .WORD   ST90,ST91,ST92,ST93,ST94
6355 15E1 0B91 T            .WORD   ST95,ST96,ST97,ST98,ST99
6359  ;
6360  ;
6361  ;           EXECUTION ROUTINE TABLE
6362  ;
6363 15E6 0000 A TAB2:      .WORD   0               ; NO ROUTINE 0
6364 15E7 0B9B T            .WORD   XQT1            ; 1: GOSUB - PUSH
6365 15E8 0BA2 T            .WORD   XQT2            ; 2: RETURN - PULL
6366 15E9 0BA9 T            .WORD   XQT3            ; 3: GOTO/GOSUB - NEXT
6367 15EA 0BB0 T            .WORD   XQT4            ; 4: STOP/END
6368 15EB 0BBF T            .WORD   XQT5            ; 5: COMMAND - ADDRESS
6369 15EC 0BC8 T            .WORD   XQT6            ; 6: COMMAND - 1ST STMT#
6370 15ED 0BCD T            .WORD   XQT7            ; 7: COMMAND - 2ND STMT#
6371 15EE 0BD2 T            .WORD   XQT8            ; 8: COMMAND - EXEC
6372 15EF 0BD5 T            .WORD   XQT9            ; 9: CHECK DIRECT EXEC COMMAND
6373 15F0 036B T            .WORD   CRLF            ; 10: PRINT CR-LF
6374 15F1 0BEC T            .WORD   XQT11           ; 11: PRINT DELIMITER
6375 15F2 0C77 T            .WORD   XQT12           ; 12: LET - COPY TEMP STRING TO
6376 15F3 0C01 T            .WORD   XQT13           ; 13: PRINT STRING VARIABLE
6377 15F4 0C19 T            .WORD   XQT14           ; 14: PRINT FUNCTION
6378 15F5 0C44 T            .WORD   XQT15           ; 15: PRINT VALUE
6379 15F6 0BDB T            .WORD   XQT16           ; 16: CHECK STRING (CLASSES 17,1
6380 15F7 0BD7 T            .WORD   XQT17           ; 17: CHECK MODE=8 (EXEC)
6381 15F8 0C4E T            .WORD   XQT18           ; 18: LET - STORE VALUE
6382 15F9 0C5D T            .WORD   XQT19           ; 19: LET - STORE STRING
6383 15FA 0C61 T            .WORD   XQT20           ; 20: LET - CONCATENATE STRING
6384 15FB 0C7E T            .WORD   XQT21           ; 21: SUBSCR - SAVE ARRAY ID
6385 15FC 0C87 T            .WORD   XQT22           ; 22: SUBSCR - 1 OF 2
6386 15FD 0D43 T            .WORD   RPAREN          ; 23: SUBSCR - 1 SUBSCR
6387 15FE 0D43 T            .WORD   RPAREN          ; 24: SUBSCR - 2 OF 2
```

STATE/ROUTINE ADDRESS TABLES

```
6388 15FF 0C88 T          .WORD   LPAREN          ; 25: EXPR - LPAREN
6389 1600 0C8E T          .WORD   XQT26           ; 26: EXPR - IDENT
6390 1601 0C95 T          .WORD   XQT27           ; 27: EXPR - CONST
6391 1602 0CA1 T          .WORD   XQT28           ; 28: EXPR - UNARY OP
6392 1603 0CB1 T          .WORD   XQT29           ; 29: EXPR - BUILT IN FUNCTION
6393 1604 0CBB T          .WORD   XQT30           ; 30: EXPR - USER FCN
6394 1605 0CE3 T          .WORD   GETARY          ; 31: EXPR - CALCULATE ARRAY ITE
6395 1606 0D29 T          .WORD   XQT32           ; 32: EXPR - EXECUTE B. I. FCN
6396 1607 0D45 T          .WORD   XQT33           ; 33: OPERATOR OR RPAREN
6397 1608 0DA6 T          .WORD   XQT34           ; 34: CHECK STACKTOP=ST39 (OPER)
6398 1609 0DB0 T          .WORD   XQT35           ; 35: DIM ARRAY ID
6399 160A 0DB7 T          .WORD   XQT36           ; 36: DIM SUBSCRIPT 1
6400 160B 0DBB T          .WORD   XQT37           ; 37: DIM SUBSCRIPT 2
6401 160C 0DC7 T          .WORD   XQT38           ; 38: TEST IF VALUE
6402 160D 0DCC T          .WORD   XQT39           ; 39: FOR IDENTIFIER
6403 160E 0DE9 T          .WORD   XQT40           ; 40: FOR INIT VALUE
6404 160F 0DEE T          .WORD   XQT41           ; 41: FOR END VALUE
6405 1610 0DF3 T          .WORD   XQT42           ; 42: FOR STEP=1
6406 1611 0DF7 T          .WORD   XQT43           ; 43: FOR STEP=VALUE
6407 1612 0E1F T          .WORD   XQT44           ; 44: NEXT
6408 1613 10C7 T          .WORD   TRCON           ; 45: TRACE ON
6409 1614 10C9 T          .WORD   TRCOFF          ; 46: TRACE OFF
6410 1615 0000 A          .WORD   DUMP            ; 47: DUMP
6411 1616 0E59 T          .WORD   XQT48           ; 48: INPUT - SET MODE
6412 1617 0E62 T          .WORD   XQT49           ; 49: INPUT/READ - IDENT
6413 1618 0E65 T          .WORD   XQT50           ; 50: INPUT/READ - ARRAY ELEMENT
6414 1619 0E5F T          .WORD   XQT51           ; 51: READ - SET MODE
6415 161A 0F2E T          .WORD   RSDATA          ; 52: RESTORE
6416 161B 0F26 T          .WORD   XQT53           ; 53: RANDOMIZE
6417 161C 0F33 T          .WORD   XQT54           ; 54: WAIT
6418 161D 0F3E T          .WORD   XQT55           ; 55: ON VALUE
6419 161E 0F47 T          .WORD   XQT56           ; 56: ON STMT#
6420 161F 0F53 T          .WORD   XQT57           ; 57: ON - END OF LIST
6421 1620 0F55 T          .WORD   XQT58           ; 58: SAVE STRING ID OPERAND
6422 1621 0F58 T          .WORD   XQT59           ; 59: SAVE STRING CONST OPERAND
6423 1622 0C9E T          .WORD   XQT60           ; 60: SAVE STRING OPERATOR
6424 1623 0FB6 T          .WORD   XQT61           ; 61: EXECUTE STRING COMPARE
6425 1624 0FDE T          .WORD   XQT62           ; 62: EXEC STRING->NUMERIC FUNCT
6426 1625 1010 T          .WORD   XQT63           ; 63: DEF - FUNCTION NAME
6427 1626 1018 T          .WORD   XQT64           ; 64: DEF - PARAMETER NAME
6428 1627 101E T          .WORD   XQT65           ; 65: DEF - EXPRESSION START
6429 1628 105D T          .WORD   XQT66           ; 66: EXECUTE POKE/OUT
6430 1629 1024 T          .WORD   XQT67           ; 67: START USER FCN
6431 162A 104C T          .WORD   XQT68           ; 68: END USER FCN
6432 162B 10B3 T          .WORD   SIZE            ; 69: SIZE
6433 162C 0E6B T          .WORD   XQT70           ; 70: STRING READ/INPUT
6434 162D 0D27 T          .WORD   XQT71           ; 71: SPECIAL IDENTIFIER (RND)
6435 162E 08E0 T          .WORD   LINIT           ; 72: NEW STATEMENT INITIALIZATI
6436 162F 0CEA T          .WORD   XQT73           ; 73: STRING ARRAY: UPDATE STACK
6437 1630 108E T          .WORD   XQT74           ; 74: EXECUTE NUMERIC->STRING FU
6438 1631 0D12 T          .WORD   XQT75           ; 75: EXECUTE SUBSTRING FUNCTION
6439 1632 0C80 T          .WORD   XQT76           ; 76: SAVE STRING ARRAY POINTER
6440 1633 10CC T          .WORD   XQT77           ; 77: SET TERMINAL WIDTH
6441 1634 1057 T          .WORD   XQT78           ; 78: WAIT (3 ARGUMENTS)
```

```
6442 1635 105B T          .WORD   XQT79          ; 79: WAIT (2 ARGUMENTS)
6443 1636 10DC T          .WORD   XQT80          ; 80: CALL
6444 1637 1A42 T          .WORD   XQT81          ; 81: FILENAME (DISK COMMANDS)
6448  ;
```

```
6449                            .PAGE   'ERROR NAME TABLE'
6450  ;
6451  ;         ERROR NAMES - 4 CHAR PER NAME
6452  ;
6453 1638 4152 A ERRTAB: .ASCII  'AREA'      ; 0: OUT OF MEMORY
6454 163A 434F A         .ASCII  'CONT'      ; 2: NO CONTINUE
6455 163C 5354 A         .ASCII  'STMT'      ; 4: STATEMENT TYPE
6456 163E 4348 A         .ASCII  'CHAR'      ; 6: CHAR AFTER LOGICAL END OF S
6457 1640 534E A         .ASCII  'SNTX'      ; 8: SYNTAX
6458 1642 5641 A         .ASCII  'VALU'      ; 10: CONSTANT FORMAT OR VALUE
6459 1644 454E A         .ASCII  'END"'      ; 12: NO CLOSING QUOTE ON STRING
6460 1646 4E4F A         .ASCII  'NOGO'      ; 14: UNDEFINED TARGET FOR GOTO
6461 1648 5254 A         .ASCII  'RTRN'      ; 16: RETURN WITHOUT GOSUB
6462 164A 4E45 A         .ASCII  'NEST'      ; 18: NEST LIMIT EXCEEDED
6463 164C 5459 A         .ASCII  'TYPE'      ; 20: TYPE MISMATCH
6464 164E 5241 A         .ASCII  'RANG'      ; 22: SUBSCRIPT OUT OF RANGE
6465 1650 4444 A         .ASCII  'DDIM'      ; 24: DUPLICATE DIM FOR ARRAY
6466 1652 4E45 A         .ASCII  'NEXT'      ; 26: NEXT WITHOUT FOR
6467 1654 464F A         .ASCII  'FOR '      ; 28: FOR WITHOUT NEXT
6468 1656 4441 A         .ASCII  'DATA'      ; 30: OUT OF DATA
6469 1658 4444 A         .ASCII  'DDEF'      ; 32: DOUBLY DEFINED FN-
6470 165A 5544 A         .ASCII  'UDEF'      ; 34: UNDEFINED FN-
6471 165C 5553 A         .ASCII  'USR '      ; 36: UNDEFINED USR FUNCTION
6472 165E 4F56 A         .ASCII  'OVFL'      ; 38: OVERFLOW
6473 1660 4449 A         .ASCII  'DIVO'      ; 40: DIVISION BY ZERO
6474 1662 4152 A         .ASCII  'ARG '      ; 42: ARGUMENT OUT OF RANGE
6475 1664 504F A         .ASCII  'POKE'      ; 44: POKE INTO BASIC
6476 1666 4449 A         .ASCII  'DISK'      ; 46: DISK ERROR
6477 1668 554E A         .ASCII  'UNFL'      ; 48: UNDERFLOW (WARNING)
6478 166A 5354 A         .ASCII  'STOV'      ; 50: STRING OVERFLOW
6479 166C 4946 A         .ASCII  'IFIX'      ; 52: IFIX WARNING
```

```
6480                            .PAGE   'SCANNER TABLES'
6481    ;
6482    ;       1-CHAR PUNCTUATION: SYMB, PREC & CLASS
6483    ;
6484 166E 003C A PCTAB:  .WORD   '<'/256,0605
6485 1670 003E A        .WORD   '>'/256,0605
6486 1672 003D A        .WORD   '='/256,0605
6487 1674 002B A        .WORD   '+'/256,0801
6488 1676 002D A        .WORD   '-'/256,0801
6489 1678 002A A        .WORD   '*'/256,0A02
6490 167A 002F A        .WORD   '/'/256,0A02
6491 167C 005E A        .WORD   05E,0E02        ; UP-ARROW
6492 167E 0028 A        .WORD   '('/256,10
6493 1680 0029 A        .WORD   ')'/256,0FF0B
6494 1682 002C A        .WORD   ','/256,12
6495 1684 003B A        .WORD   ';'/256,12
6496 1686 000D A        .WORD   0D,13           ; CARRIAGE RETURN
6497 1688 0027 A        .WORD   REMCHAR,13      ; APOSTROPHE: REMARK FOLLOWS
6498 168A 003A A PCLAST: .WORD   ':'/256,14     ; COLON: STATEMENT SEPARATOR
6499    ;
6500    ;       RESERVED WORD TABLE - USED BY PRESCANNER
6501    ;               ENTRIES ARE DELIMITED BY A NULL BYTE WHICH
6502    ;               MAY BE PART OF THE FOLLOWING KEYWORD CODE WORD
6503    ;
6504 168C 0080 A RESTAB: .WORD   128,'RE','M'-B
6505 168F 0081 A        .WORD   129,'DI','M'-B
6506 1692 0082 A        .WORD   130,'LE','T'-B
6507 1695 0083 A        .WORD   131,'GO','TO'
6508 1698 0084 A        .WORD   132,'GO','SU','B'-B
6509 169C 0085 A        .WORD   133,'RE','TU','RN'
6510 16A0 0086 A        .WORD   134,'IF'
6511 16A2 0087 A        .WORD   135,'TH','EN'
6512 16A5 0088 A        .WORD   136,'FO','R'-B
6513 16A8 0089 A        .WORD   137,'TO'
6514 16AA 008A A        .WORD   138,'ST','EP'
6515 16AD 008B A        .WORD   139,'NE','XT'
6516 16B0 008C A        .WORD   140,'IN','PU','T'-B
6517 16B4 008D A        .WORD   141,'PR','IN','T'-B
6518 16B8 008E A        .WORD   142,'ST','OP'
6519 16BB 008F A        .WORD   143,'EN','D'-B
6520 16BE 0090 A        .WORD   144,'DA','TA'
6521 16C1 0091 A        .WORD   145,'RE','AD'
6522 16C4 0092 A        .WORD   146,'RE','ST','OR','E'-B
6523 16C9 0093 A        .WORD   147,'DE','F'-B
6524 16CC 0094 A        .WORD   148,'ON'
6525 16CE 0095 A        .WORD   149,'TR','AC','E'-B
6526 16D2 0096 A        .WORD   150,'RA','ND','OM','IZ','E'-B
6527 16D8 0097 A        .WORD   151,'TW','AI','T'-B
6528 16DC 0098 A        .WORD   152,'OF','F'-B
6529 16DF 0099 A        .WORD   153,'PO','KE'
6530 16E2 009A A        .WORD   154,'OU','T'
6534 16E5 009C A        .WORD   156,'SI','ZE'
6535 16E8 009D A        .WORD   157,'GO'
6536 16EA 009E A        .WORD   158,'SU','B'-B
```

```
6537 16ED 009F A          .WORD   159,'WI','DT','H'-B
6538 16F1 00A0 A          .WORD   160,'NO','T'-B
6539 16F4 00A1 A          .WORD   161,'AN','D'-B
6540 16F7 00A2 A          .WORD   162,'OR'
6541 16F9 00A3 A          .WORD   163,'XO','R'-B
6542 16FC 00A4 A          .WORD   164,'TA','B'-B
6543 16FF 00A5 A          .WORD   165,'LE','N'-B
6544 1702 00A6 A          .WORD   166,'MA','X'-B
6545 1705 00A7 A          .WORD   167,'MI','N'-B
6546 1708 00A8 A          .WORD   168,'MO','D'-B
6547 170B 00A9 A          .WORD   169,'TA','N'-B
6548 170E 00AA A          .WORD   170,'IN','T'-B
6549 1711 00AB A          .WORD   171,'AB','S'-B
6550 1714 00AC A          .WORD   172,'SG','N'-B
6551 1717 00AD A          .WORD   173,'EX','P'-B
6552 171A 00AE A          .WORD   174,'LO','G'-B
6553 171D 00AF A          .WORD   175,'SQ','R'-B
6554 1720 00B0 A          .WORD   176,'RN','D'-B
6555 1723 00B1 A          .WORD   177,'SI','N'-B
6556 1726 00B2 A          .WORD   178,'AT','N'-B
6557 1729 00B3 A          .WORD   179,'CO','S'-B
6558 172C 00B4 A          .WORD   180,'MI','D$'
6559 172F 00B5 A          .WORD   181,'CH','R$'
6560 1732 00B6 A          .WORD   182,'ST','R$'
6561 1735 00B7 A          .WORD   183,'AS','C'-B
6562 1738 00B8 A          .WORD   184,'VA','L'-B
6563 173B 00B9 A          .WORD   185,'PO','S'-B
6564 173E 00BA A          .WORD   186,'US','R'-B
6565 1741 00BB A          .WORD   187,'PE','EK'
6566                ;     .WORD   188,'IN','P'-B  ; REPLACED BY 140 (INPUT)
6567 1744 00BD A          .WORD   189,'SP','C'-B
6568 1747 00BE A          .WORD   190,'LI','N'-B
6569 174A 00BF A          .WORD   191,'LE','FT','$'-B
6570 174E 00C0 A          .WORD   192,'RI','GH','T$'
6571 1752 00C1 A          .WORD   193,'BR','K'-B
6572 1755 00C2 A          .WORD   194,'HE','X$'
6573 1758 00C7 A          .WORD   199,'WA','IT'
6574 175B 00C8 A          .WORD   200,'CA','LL'
6575 175E 00C9 A          .WORD   201,'SC','RA','TC','H'-B
6576 1763 00CA A          .WORD   202,'DE','LE','TE'
6577 1767 00CB A          .WORD   203,'CL','EA','R'-B
6578 176B 00CC A          .WORD   204,'RU','N'-B
6579 176E 00CD A          .WORD   205,'CO','NT','IN','UE'
6580 1773 00CF A          .WORD   207,'TA','PE'
6581 1776 00D0 A          .WORD   208,'LI','ST'
6582 1779 00D1 A          .WORD   209,'HL','IS','T'-B
6583 177D 00D2 A          .WORD   210,'PU','NC','H'-B
6584 1781 00D3 A          .WORD   211,'AU','TO'
6585 1784 00D4 A          .WORD   212,'BY','E'-B
6586 1787 00FA A          .WORD   250,'<='      ; PUNCTUATION: ALTERNATE SPELLIN
6587 1789 00FA A          .WORD   250,'=<'      ;    WILL BE TRANSLATED TO THE S
6588 178B 00FB A          .WORD   251,'>='
6589 178D 00FB A          .WORD   251,'=>'
6590 178F 00FC A          .WORD   252,'<>'
```

```
6591 1791 00FC A          .WORD   252,'><'
6592 1793 005E A          .WORD   05E,'**'      ; TRANSLATE TO CIRCUMFLEX
6594 1795 00CE A DSKCOM:  .WORD   206,'LO','AD' ; DISK COMMANDS
6595 1798 00D5 A          .WORD   213,'SA','VE'
6596 179B 00D6 A          .WORD   214,'EX','EC'
6597 179E 00D7 A          .WORD   215,'ME','RG','E'-B
6599 17A2 0000 A          .WORD   0
6600  ;
6601  ;         IDENTIFIER TABLE - KEYWORDS, FUNCTIONS, COMMANDS
6602  ;         ENTRIES: SYMB, PREC & CLASS
6603  ;                  CLASS IS 25 IF NOT IN THIS TABLE
6604  ;
6605  ;         OPERATORS:
6606 17A3 00A0 A VALTAB:  .WORD   160,12*256+3 ; NOT
6607 17A5 00A1 A          .WORD   161,4*256+4  ; AND
6608 17A7 00A2 A          .WORD   162,2*256+4  ; OR
6609 17A9 00A3 A          .WORD   163,2*256+4  ; XOR
6610 17AB 00A6 A          .WORD   166,7*256+2  ; MAX
6611 17AD 00A7 A          .WORD   167,7*256+2  ; MIN
6612 17AF 00A8 A          .WORD   168,11*256+2 ; MOD
6613 17B1 00FA A          .WORD   250,0605     ; <=
6614 17B3 00FB A          .WORD   251,0605     ; >=
6615 17B5 00FC A          .WORD   252,0605     ; <>
6616  ;         FUNCTIONS:
6617 17B7 00A9 A          .WORD   169,6        ; TAN
6618 17B9 00AA A          .WORD   170,6        ; INT
6619 17BB 00AB A          .WORD   171,6        ; ABS
6620 17BD 00AC A          .WORD   172,6        ; SGN
6621 17BF 00AD A          .WORD   173,6        ; EXP
6622 17C1 00AE A          .WORD   174,6        ; LOG
6623 17C3 00AF A          .WORD   175,6        ; SQR
6624 17C5 00B0 A          .WORD   176,6        ; RND
6625 17C7 00B1 A          .WORD   177,6        ; SIN
6626 17C9 00B2 A          .WORD   178,6        ; ATN
6627 17CB 00B3 A          .WORD   179,6        ; COS
6628 17CD 00A4 A          .WORD   164,8        ; TAB
6629 17CF 00A5 A          .WORD   165,9        ; LEN
6630 17D1 00B4 A          .WORD   180,19       ; MID$
6631 17D3 00B5 A          .WORD   181,23       ; CHR$
6632 17D5 00B6 A          .WORD   182,23       ; STR$
6633 17D7 00B7 A          .WORD   183,9        ; ASC
6634 17D9 00B8 A          .WORD   184,9        ; VAL
6635 17DB 00B9 A          .WORD   185,6        ; POS
6636 17DD 00BA A          .WORD   186,6        ; USR
6637 17DF 00BB A          .WORD   187,6        ; PEEK
6638 17E1 008C A          .WORD   140,6        ; INP/INPUT
6639 17E3 00BD A          .WORD   189,8        ; SPC
6640 17E5 00BE A          .WORD   190,8        ; LIN
6641 17E7 00BF A          .WORD   191,19       ; LEFT$
6642 17E9 00C0 A          .WORD   192,19       ; RIGHT$
6643 17EB 00C1 A          .WORD   193,6        ; BRK
6644 17ED 00C2 A          .WORD   194,23       ; HEX$
6645  ;         COMMANDS:
6646 17EF 00C9 A          .WORD   201,27       ; SCRATCH
```

```
6647 17F1 00CA A          .WORD   202,27          ; DELETE
6648 17F3 00CB A          .WORD   203,27          ; CLEAR
6649 17F5 00CC A          .WORD   204,27          ; RUN
6650 17F7 00CD A          .WORD   205,27          ; CONTINUE
6652 17F9 00CE A          .WORD   206,28          ; LOAD
6654 17FB 00CF A          .WORD   207,27          ; TAPE
6655 17FD 00D0 A          .WORD   208,27          ; LIST
6656 17FF 00D1 A          .WORD   209,27          ; HLIST
6657 1801 00D2 A          .WORD   210,27          ; PUNCH
6658 1803 00D3 A          .WORD   211,27          ; AUTO
6660 1805 00D4 A          .WORD   212,27          ; BYE
6661 1807 00D5 A          .WORD   213,28          ; SAVE
6662 1809 00D7 A          .WORD   215,28          ; MERGE
6664 180B 0000 A          .WORD   0
6665  ;
```

```
                        .PAGE   'OPERATOR TABLE'
;
;       STRING OPERATORS (COMPARISONS)
;
180C 00FA A TAB3A:  .WORD   250,FLLE+1    ; <=
180E 00FB A         .WORD   251,FLGE+1    ; >=
1810 00FC A         .WORD   252,FLNE+1    ; <>
1812 003D A         .WORD   '='/256,FLEQ+1
1814 003C A         .WORD   '<'/256,FLLT+1
1816 003E A         .WORD   '>'/256,FLGT+1
;
;       FLOATING POINT OPERATORS
;
1818 002B A TAB3:   .WORD   '+'/256,FLADD
181A 002D A         .WORD   '-'/256,FLSUB
181C 002A A         .WORD   '*'/256,FLMULT
181E 002F A         .WORD   '/'/256,FLDIV
1820 005E A         .WORD   05E,POWER
1822 012D A         .WORD   '-'/256+0100,NEGATE ; UNARY MINUS
1824 01A0 A         .WORD   160+0100,FLNOT
1826 00A1 A         .WORD   161,FLAND
1828 00A2 A         .WORD   162,FLOR
182A 00A3 A         .WORD   163,FLXOR
182C 00A6 A         .WORD   166,FLMAX
182E 00A7 A         .WORD   167,FLMIN
1830 00A8 A         .WORD   168,FLMOD
1832 00FA A         .WORD   250,FLLE      ; <=
1834 00FB A         .WORD   251,FLGE      ; >=
1836 00FC A         .WORD   252,FLNE      ; <>
1838 003D A         .WORD   '='/256,FLEQ
183A 003C A         .WORD   '<'/256,FLLT
183C 003E A TAB3L:  .WORD   '>'/256,FLGT
;
;       BUILT-IN-FUNCTION ADDRESS TABLE
;
183E 00A9 A TAB4:   .WORD   169,TAN
1840 00AA A         .WORD   170,FLINT
1842 00AB A         .WORD   171,ABS
1844 00AC A         .WORD   172,SGN
1846 00AD A         .WORD   173,EXPON
1848 00AE A         .WORD   174,LN
184A 00AF A         .WORD   175,SQRT
184C 00B0 A         .WORD   176,RND
184E 00B1 A         .WORD   177,SINE
1850 00B2 A         .WORD   178,ARCTAN
1852 00B3 A         .WORD   179,COS
1854 00B9 A         .WORD   185,POSFCN
1856 00BA A         .WORD   186,USR
1858 00BB A         .WORD   187,PEEK
185A 008C A         .WORD   140,INP
185C 00C1 A TAB4L:  .WORD   193,BRKFCN
;
;       COMMAND ADDRESS TABLE
;       ENTRIES FOR 201 THROUGH 215 - IN ORDER
```

```
6720  ;
6727  ;
6728 185E 0114 T CMDTAB: .WORD   SCRATCH
6729 185F 04F2 T         .WORD   DELETE
6730 1860 0117 T         .WORD   RESET
6731 1861 01A8 T         .WORD   RUN
6732 1862 023D T         .WORD   CONT
6733 1863 0133 T         .WORD   LOAD
6734 1864 013C T         .WORD   PTAPE
6735 1865 02FB T         .WORD   TLIST
6736 1866 02E1 T         .WORD   HLIST
6737 1867 02EA T         .WORD   PUNCH
6738 1868 0140 T         .WORD   AUTO
6739 1869 024D T         .WORD   BYE
6740 186A 02F3 T         .WORD   SAVFIL
6741 186B 0131 T         .WORD   EXEC
6742 186C 0135 T         .WORD   MERGE
6743  ;
```

```
                                .PAGE   'MESSAGES AND BUFFERS'
 ;
186D 0D0A A READY:  .WORD   0D0A,0D0A
186F 5245 A         .ASCII  'READY'
1872 0D0A A.        .WORD   0D0A,0
1874 5052 A MSZ1:   .WORD   'PR','OG',0
1877 0D0A A MSZ3:   .WORD   0D0A,'AR','AY',0
187B 0D0A A MSZ4:   .WORD   0D0A,'SY','MB',0
187F 2D3E A POINT:  .WORD   '->',0
1881 2057 A WARNM:  .ASCII  ' WARNING'
1885 2000 A         .WORD   ' '-B
1886 2045 A ERM1:   .ASCII  ' ERROR'
1889 2000 A         .WORD   ' '-B
188A 494E A ERM2:   .WORD   'IN',' '-B
188C 5354 A STOPM:  .WORD   'ST','OP',' '-B
188F 4252 A BRK:    .WORD   'BR','EA','K ',0
1893 4152 A ALOCM:  .ASCII  'ARY/STR NOT ALLOCATED'
189E 0D0A A         .WORD . 0D0A,0
18A0 4552 A INERR:  .ASCII  'ERROR: RETYPE LINE'
18A9 0D0A A         .WORD   0D0A,0
18AB 3F00 A MOREIN: .WORD   '?'-B
18AC 0D0A A SYSERM: .WORD   0D0A,'SY','ST','EM',' '-B
18B1 4552 A ATLOC:  .ASCII  'ERROR AT LOC'
18B7 2000 A         .WORD   ' '-B
18B8 0D0A A OPERM:  .WORD   0D0A
18B9 4F50 A         .ASCII  'OPERAND '
18BD 0000 A         .WORD   0
 ;
 ;
 ;
18BE        INBUF:  .=.+133             ; BUFFER FOR INPUT (ONE CHAR PER
1943        PBUF:   .=.+67              ; BUFFER FOR PRESCANNER
1986        STR3:   .=.+41              ; TEMPORARY AREA FOR STRING ASSI
19AF        STACK:  .=.+20              ; HARDWARE STACK EXTENSION
19C3        STACK0: .=.+32              ; REGISTER SAVE STACK
19E3        STACK1: .=.+15              ; SYNTAX ANALYZER STACK
19F2        STACK2: .=.+30              ; EXPRESSION OPERAND STACK
1A10        STACK3: .=.+20              ; EXPRESSION OPERATOR STACK
1A24        STACK4: .=.+10              ; USER FUNCTION STACK
1A2E        STACK5: .=.+10              ; GOSUB STACK
1A38        STACK6: .=.+10              ; FOR-NEXT STACK (PASS 1 ONLY)
    1A42 T STACKX  =        .
 ;
```

```
6797                            .PAGE   'DISK ROUTINES'
6798        8011 A              .LIST   LOPT!010
6799 1A42                       .TSECT
6800                            .LOCAL
6801  ;
6802  ;         PARSE FILENAME AND SET UP FILE STATUS TABLE
6803  ;
6804                  XQT81:
6812 1A42 CCB1*A                LD      AC3,=INBUF
6813 1A43 C843 A                LD      AC2,SYMA        ; GET PACKED STRING ADDR
6814 1A44 C600 A                LD      AC1,0(AC2)      ;    & COUNT
6815 1A45 F4CC*A                SKNE    AC1,=0
6816 1A46 190E A                JMP     $END
6817 1A47 7A01 A $UNP:          AISZ    AC2,1           ; UNPACK STRING INTO INBUF
6818 1A48 C200 A                LD      AC0,(AC2)
6819 1A49 2C10 A                SHR     AC0,8,0
6820 1A4A D300 A                ST      AC0,(AC3)
6821 1A4B 7B01 A                AISZ    AC3,1
6822 1A4C 79FF A                AISZ    AC1,-1
6823 1A4D 1901 A                JMP     .+2
6824 1A4E 1906 A                JMP     $END
6825 1A4F C200 A                LD      AC0,(AC2)
6826 1A50 A8D5*A                AND     AC0,=0FF
6827 1A51 D300 A                ST      AC0,(AC3)
6828 1A52 7B01 A                AISZ    AC3,1
6829 1A53 79FF A                AISZ    AC1,-1
6830 1A54 19F2 A                JMP     $UNP
6831 1A55 500D A $END:          LI      AC0,0D          ; TERMINATE JUST IN CASE
6832 1A56 D300 A                ST      AC0,(AC3)
6833 1A57 C8CA*A                LD      AC2,=FST
6834 1A58 C110 A                LD      AC0,$F1         ; SET DEFAULT MODIFIER
6835 1A59 D204 A                ST      AC0,4(AC2)      ;    TO 'BAS'
6836 1A5A C10F A                LD      AC0,$F2
6837 1A5B D205 A                ST      AC0,5(AC2)
6838 1A5C C12B*A                LD      AC0,=02000      ; SET PROT LEVEL
6839 1A5D D206 A                ST      AC0,6(AC2)
6840 1A5E C12A*A                LD      AC0,=DSKBUF     ; SET BUFFER ADDRESS
6841 1A5F D208 A                ST      AC0,8(AC2)
6842 1A60 5000 A                LI      AC0,0           ; CLEAR FLAGS IN FST
6843 1A61 D20B A                ST      AC0,11(AC2)
6844 1A62 D20C A                ST      AC0,12(AC2)
6845 1A63 D20D A                ST      AC0,13(AC2)
6846 1A64 CCB1*A                LD      AC3,=INBUF
6847 1A65 9524*A                JSR     PGFR            ; PARSE FILE REFERENCE
6848 1A66 1904 A                JMP     DSKERR
6849 1A67 1903 A                JMP     DSKERR
6850 1A68 8000 A                RTS     0
6851 1A69 4241 A $F1:           .WORD   'BA'
6852 1A6A 5305 A $F2:           .WORD   'S'-B+5         ; SYMBOLIC TYPE
6853  ;
6854 1A6B 502E A DSKERR:        LI      AC0,46          ; DISK ERROR: PRINT
6855 1A6C 94C3*A                JSR     ERROR           ;    MESSAGE
6856 1A6D C8CA*A                LD      AC2,=FST
6857 1A6E 951C*A                JSR     DERROR
```

```
6858 1A6F 9895*A          JMP     BEGIN
6859  ;
6860  ;
6861  ;         DISK INPUT ROUTINE   (READS COMPLETE LINE)
6862  ;
6863  ;         LD      AC2,BUFADR   ; BUFFER ADDRESS IS IN AC2
6864  ;         JSR     DISKIN
6865  ;         <ERROR RETURN>
6866  ;         <EOF RETURN>
6867  ;         <NORMAL RETURN>
6868  ;
6869 1A70 9499*A DISKIN: JSR     SAVE
6870 1A71 6200 A $NXTC:  PUSH    AC2          ; GET CHAR FROM DISK
6871 1A72 C8CA*A         LD      AC2,=FST
6872 1A73 151A A         JSR     RCHAR
6873 1A74 19F6 A         JMP     DSKERR
6874 1A75 6600 A         PULL    AC2
6875 1A76 F0D0*A         SKNE    AC0,=0D
6876 1A77 1908 A         JMP     $CRRET       ; END OF LINE
6877 1A78 F113*A         SKNE    AC0,=01C
6878 1A79 1909 A         JMP     $EOF
6879 1A7A D07B A         ST      AC0,SCR
6880 1A7B F820 A         SKNE    AC2,ENDBUF   ; DON'T STORE IF PAST END OF BUF
6881 1A7C 19F4 A         JMP     $NXTC
6882 1A7D D200 A         ST      AC0,(AC2)
6883 1A7E 7A01 A         AISZ    AC2,1
6884 1A7F 19F1 A         JMP     $NXTC
6885  ;
6886 1A80 D200 A $CRRET: ST      AC0,(AC2)
6887 1A81 949A*A         JSR     RESTOR
6888 1A82 8002 A         RTS     2
6889 1A83 C8CA*A $EOF:   LD      AC2,=FST     ; CLOSE THE FILE
6890 1A84 9508*A         JSR     CLOSE
6891 1A85 19E5 A         JMP     DSKERR
6892 1A86 949A*A         JSR     RESTOR
6893 1A87 8001 A         RTS     1
6894  ;
6895 1A88 2000 A         .POOL   6
     1A89 1AF3 T
     1A8A D404 A
     1A8B D430 A
     1A8C 001C A
     1A8D D416 A
```

```
6896                            .PAGE
6897    ;*****************************************************************************
6898    ;
6899    ;       CHARACTER-ORIENTED DISK I/O ROUTINES:
6900    ;          .RCHAR   READ CHARACTER
6901    ;           WCHAR   WRITE CHARACTER
6902    ;
6903    ;       THESE ROUTINES PASS THE CHARACTER IN AC0 (RIGHT BYTE).
6904    ;       REGISTERS AC2 AND AC3 ARE PRESERVED (AC2 IS THE FILE
6905    ;       STATUS TABLE ADDRESS).
6906    ;
6907    ;       THESE ROUTINES USE AN EXTENDED FILE STATUS TABLE.
6908    ;       ADDED WORDS IN TABLE ARE:
6909    ;          10   CURRENT WORD SAVE
6910    ;          11   LEFT/RIGHT BYTE FLAG (0=LEFT)
6911    ;          12   BLANK COUNT
6912    ;          13   # CHAR WRITTEN TO LINE (WCHAR ONLY)
6913    ;
6914    ;*****************************************************************************
6915    ;
6916    ;
6917    ;       READ CHARACTER FROM DISK
6918    ;
6919    ;       LD      AC2,AFST
6920    ;       JSR     RCHAR
6921    ;       <ERROR RETURN>
6922    ;       <NORMAL RETURN>
6923    ;
6924    ;       CHARACTER IS RETURNED IN AC0; AC2 AND AC3 ARE PRESERVED.
6925    ;
6926                            .LOCAL
6927 1A8E C20C A RCHAR:  LD      AC0,12(AC2)   ; CHECK BLANK COUNT
6928 1A8F 4513 A         BOC     NZR0,$BL
6929 1A90 C20B A         LD      AC0,11(AC2)   ; CHECK L/R BYTE FLAG
6930 1A91 4105 A         BOC     ZR0,$L
6931 1A92 5000 A         LI      AC0,0         ; RIGHT: RESET FLAG
6932 1A93 D20B A         ST      AC0,11(AC2)
6933 1A94 C20A A         LD      AC0,10(AC2)   ;    GET WORD
6934 1A95 A8D5*A         AND     AC0,=0FF
6935 1A96 1905 A         JMP     $CH
6936 1A97 8E0B A $L:     ISZ     11(AC2)       ; LEFT: INCR. FLAG
6937 1A98 9558*A         JSR     RWORD         ;    READ WORD
6938 1A99 8000 A         RTS     0
6939 1A9A D20A A         ST      AC0,10(AC2)   ;    SAVE WORD
6940 1A9B 2C10 A         SHR     AC0,8,0       ; RETURN LEFT BYTE
6941 1A9C 41F1 A $CH:    BOC     ZR0,RCHAR     ; IGNORE NULL CHAR
6942 1A9D B8D4*A         SKAZ    AC0,=080      ; BLANK COUNT ?
6943 1A9F 1901 A         JMP     .+2           ;    YES
6944 1A9F 8001 A .       RTS     1             ;    NO
6945 1AA0 A8D3*A         AND     AC0,=07F      ; STRIP OFF FLAG BIT
6946 1AA1 41EC A         BOC     ZR0,RCHAR     ; IGNORE ZERO COUNT
6947 1AA2 D20C A         ST      AC0,12(AC2)   ; SAVE COUNT
6948 1AA3 AE0C A $BL:    DSZ     12(AC2)       ; BLANK: DECR. COUNT
6949 1AA4 5C00 A         NOP
```

```
6950 1AA5 5020 A          LI      AC0,BLANK
6951 1AA6 8001 A          RTS     1
6952  ;
6953  ;
6954  ;        WRITE CHARACTER TO DISK
6955  ;
6956  ;        LD      AC0,CHARACTER
6957  ;        LD      AC2,AFST
6958  ;        JSR     WCHAR
6959  ;        <ERROR RETURN>
6960  ;        <NORMAL RETURN>
6961  ;
6962                    .LOCAL
6963 1AA7 A8D3*A WCHAR: AND     AC0,=07F      ; REMOVE PARITY
6964 1AA8 F0D2*A       SKNE    AC0,=BLANK    ; BLANK ?
6965 1AA9 1915 A       JMP     $BL
6966 1AAA 4111 A       BOC     ZR0,$RTS1     ; IGNORE NULL
6967 1AAB F0D0*A       SKNE    AC0,=0D       ; CR ?
6968 1AAC 1919 A       JMP     $CR
6969 1AAD F0CF*A       SKNE    AC0,=0A       ; LF ?
6970 1AAE 1913 A       JMP     $LF
6971 1AAF F1DC*A       SKNE    AC0,=01C      ; EOF ?
6972 1AB0 1921 A       JMP     $EOF
6973 1AB1 6000 A       PUSH    AC0           ; SAVE CHAR
6974 1AB2 C20C A       LD      AC0,12(AC2)   ; CHECK BLANK COUNT
6975 1AB3 4105 A       BOC     ZR0,$OUT      ;    NONE
6976 1AB4 A4D4*A       OR      AC0,=080      ;    YES - SEND IT OUT
6977 1AB5 152B A       JSR     PUTDC
6978 1AB6 1906 A       JMP     $SPER
6979 1AB7 5000 A       LI      AC0,0         ;    ZERO COUNT NOW
6980 1AB8 D20C A       ST      AC0,12(AC2)
6981 1AB9 6400 A $OUT: PULL    AC0           ; RECOVER CHAR
6982 1ABA 1526 A       JSR     PUTDC         ; SEND IT OUT
6983 1ABB 8000 A       RTS     0
6984 1ABC 8001 A $RTS1: RTS    1
6985 1ABD 6400 A $SPER: PULL   AC0           ; ERROR WHEN SENDING SPACE
6986 1ABE 8000 A       RTS     0
6987 1ABF 8E0C A $BL:  ISZ     12(AC2)       ; INCREMENT BLANK COUNT
6988 1AC0 5C00 A       NOP
6989 1AC1 8001 A       RTS     1
6990 1AC2 C60D A $LF:  LD      AC1,13(AC2)   ; LF - DON'T SEND IF NEW
6991 1AC3 7900 A       AISZ    AC1,0         ;    LINE (1ST LF AFTER
6992 1AC4 19F5 A       JMP     $OUT+1        ;    CR IS IGNORED)
6993 1AC5 8001 A       RTS     1
6994 1AC6 C20B A $CR:  LD      AC0,11(AC2)   ; CURRENT RIGHT BYTE ?
6995 1AC7 4103 A       BOC     ZR0,$CR2      ; (NOTE TRAIL BLANKS IGNORED)
6996 1AC8 C20A A       LD      AC0,10(AC2)
6997 1AC9 9528*A       JSR     WWORD
6998 1ACA 8000 A       RTS     0
6999 1ACB 500D A $CR2: LI      AC0,0D
7000 1ACC 9525*A       JSR     WWORD
7001 1ACD 8000 A       RTS     0
7002 1ACE 5000 A $CR3: LI      AC0,0         ; CLEAR FLAGS ON NEW LINE
7003 1ACF D20B A       ST      AC0,11(AC2)
```

```
7004 1AD0 D20D A          ST      AC0,13(AC2)
7005 1AD1 8001 A          RTS     1
7006 1AD2 C20B A $EOF:    LD      AC0,11(AC2)   ; EOF - SEND CURRENT WORD IF
7007 1AD3 4104 A          BOC     ZR0,$EF2      ;    ANY, THEN CR IF LINE
7008 1AD4 C20A A          LD      AC0,10(AC2)   ;    NOT EMPTY
7009 1AD5 951C*A          JSR     WWORD
7010 1AD6 8000 A          RTS     0
7011 1AD7 1902 A          JMP     .+3
7012 1AD8 C20D A $EF2:    LD      AC0,13(AC2)
7013 1AD9 4103 A          BOC     ZR0,$EF3
7014 1ADA 500D A          LI      AC0,0D
7015 1ADB 9516*A          JSR     WWORD
7016 1ADC 8000 A          RTS     0
7017 1ADD 501C A $EF3:    LI      AC0,01C       ; SEND EOF CHAR
7018 1ADE 9513*A          JSR     WWORD
7019 1ADF 8000 A          RTS     0
7020 1AE0 19ED A          JMP     $CR3
7021  ;
7022 1AE1 C60B A PUTDC:   LD      AC1,11(AC2)   ; CHECK WHICH BYTE TO WRITE
7023 1AE2 7900 A          AISZ    AC1,0
7024 1AE3 1905 A          JMP     $R
7025 1AE4 2810 A          SHL     AC0,8,0       ; LEFT BYTE: SAVE
7026 1AE5 D20A A          ST      AC0,10(AC2)
7027 1AE6 8E0B A          ISZ     11(AC2)       ; INCR BYTE FLAG
7028 1AE7 8E0D A          ISZ     13(AC2)       ; INCR CHAR COUNT
7029 1AE8 8001 A          RTS     1
7030 1AE9 A8D5*A $R:      AND     AC0,=0FF      ; RIGHT: ADD LEFT IN
7031 1AEA A60A A          OR      AC0,10(AC2)
7032 1AEB 9506*A          JSR     WWORD         ; WRITE WORD
7033 1AEC 8000 A          RTS     0
7034 1AED 5100 A          LI      AC1,0         ; CLEAR BYTE FLAG
7035 1AEE D60B A          ST      AC1,11(AC2)
7036 1AEF 8E0D A          ISZ     13(AC2)       ; INCREMENT COUNT
7037 1AF0 8001 A          RTS     1
7038 1AF1 D422 A          .POOL   2
     1AF2 D424 A
7039  ;
7040 1AF3         DSKBUF: .=.+256
7044  ;
```

```
7045                        .PAGE   'INITIALIZATION ENTRY'
7046  ;
7047  ;         *** THESE ROUTINES BECOME PART OF THE EDIT ***
7048  ;         *** STORAGE AS SOON AS INITIALIZATION IS   ***
7049  ;         *** FINISHED.  (EXCEPT FOR ROM VERSION).   ***
7050  ;
7051  ;
7052  ;         RDSCAN - READS A LINE FROM THE TERMINAL AND MAKES
7053  ;                  THE FIRST CALL ON SCAN
7054  ;
7055  ;         USED BY INITIALIZATION
7056  ;
7057  ;         RETURNS:  RTS 0 -- ERROR (LINE ABORT OR SCAN ERROR)
7058  ;                   RTS 1 -- CONTROL-C ABORT
7059  ;                   RTS 2 -- BLANK LINE
7060  ;                   RTS 3 -- NORMAL RETURN
7061  ;
7062 1BF3 5000 A RDSCAN: LI      AC0,0          ; READ FROM TTY
7063 1BF4 D01A A         ST      AC0,UNIT
7064 1BF5 94AD*A         JSR     RDLINE
7065 1BF6 8000 A         RTS     0
7066 1BF7 8001 A         RTS     1
7067 1BF8 94AE*A         JSR     PACK
7068 1BF9 C0B1*A         LD      AC0,=INBUF     ; SET PTRS FOR SCAN
7069 1BFA D03E A         ST      AC0,LSTART
7070 1BFB 5000 A         LI      AC0,0
7071 1BFC D03F A         ST      AC0,COL
7072 1BFD D03A A         ST      AC0,NTYPE
7073 1BFE 94A3*A         JSR     SCAN
7074 1BFF 8000 A         RTS     0
7075 1C00 500D A         LI      AC0,13         ; CHECK FOR EOL (BLANK LINE)
7076 1C01 F041 A         SKNE    AC0,CLASS
7077 1C02 8002 A         RTS     2
7078 1C03 8003 A         RTS     3
7079  ;
7080  ;         INITIALIZATION ENTRY: INITIALIZE ALL POINTERS
7081  ;         ASK IF DISK INPUT NEEDED; THEN ASK FOR MEMORY SIZE.
7082  ;
7083             ENTRY:
7095 1C04 94C1*A         JSR     CLRSTK         ; INITIALIZE STACKS
7096 1C05 5107 A         LI      AC1,7
7097 1C06 94C2*A         JSR     CLRST1
7098 1C07 5000 A         LI      AC0,0          ; INITIALIZE SOME VARIABLES
7099 1C08 D01A A         ST      AC0,UNIT
7100 1C09 D01C A         ST      AC0,OUTDVC
7101 1C0A D03A A         ST      AC0,NTYPE
7102 1C0B D014 A         ST      AC0,CURLIN
7103 1C0C 5001 A         LI      AC0,1
7104 1C0D D017 A         ST      AC0,BRKMOD
7105 1C0E D011 A         ST      AC0,PROT
7110 1C0F C132*A         LD      AC0,=RDSCAN    ; SAVE BUFFER START ADDRESS
7112 1C10 D033 A         ST      AC0,ESTART
7113 1C11 C131*A         LD      AC0,=NOUSER    ; NO USER FUNCTION INITIALLY
7114 1C12 D010 A         ST      AC0,USER
```

```
7115 1C13 5048 A          LI      AC0,72
7116 1C14 D05A A          ST      AC0,NSTRC     ; SET STORED STRING LENGTH VARIA
7117 1C15 952E*A          JSR     TWIDTH
7118 1C16 5025 A          LI      AC0,37
7119 1C17 D05B A          ST      AC0,NSTRW
7120 1C18 50CD A          LI      AC0,-51       ; RESERVE LAST 50 WORDS OF MEMOR
7121 1C19 D05C A          ST      AC0,OP1       ;    FOR FIRMWARE (MORE IF DISK
7122 1C1A 94A1*A          JSR     MESG          ; ANNOUNCE PROGRAM NAME
7123 1C1B 1C78 T          .WORD   MSG1
7124 1C1C C928*A          LD      AC2,=EBUF1    ; CHECK MEMORY (NONDESTRUCTIVE)
7125 1C1D C200 A CSIZE:   LD      AC0,(AC2)     ; COMPLEMENT MEMORY DATA
7126 1C1E 7000 A          CAI     AC0,0
7127 1C1F D200 A          ST      AC0,(AC2)     ; STORE AND READ COMPLEMENTED DA
7128 1C20 F200 A          SKNE    AC0,(AC2)     ;   ORIGINAL DATA UNTIL NON-VERI
7129 1C21 1901 A          JMP     .+2
7130 1C22 1907 A          JMP     CSIZE2
7131 1C23 7000 A          CAI     AC0,0
7132 1C24 D200 A          ST      AC0,(AC2)     ; MEMORY SHOULD NOW HAVE ORIGINA
7133 1C25 F200 A          SKNE    AC0,(AC2)
7134 1C26 1901 A          JMP     .+2
7135 1C27 1902 A          JMP     CSIZE2
7136 1C28 7A01 A          AISZ    AC2,1
7137 1C29 19F3 A          JMP     CSIZE
7138 1C2A D838 A CSIZE2:  ST      AC2,LAST      ; SAVE LAST MEM ADDR
7139 1C2B D851 A          ST      AC2,SY2
7140              DSKASK:
7142 1C2C A111 A          LD      AC0,@AMON     ; CHECK IF DOS PRESENT
7143 1C2D 450B A          BOC     NZRO,NODSK    ;    (CHECKS START OF FIRMWARE)
7144 1C2E 94A1*A          JSR     MESG          ; ASK IF DISK INPUT NEEDED
7145 1C2F 1C8D T          .WORD   ASKDSK
7146 1C30 9515*A          JSR     GECO          ; GET RESPONSE - SINGLE CHARACTE
7147 1C31 A8D3*A          AND     AC0,=07F
7148 1C32 F114*A          SKNE    AC0,='Y'/256  ; 'Y' OR CR = YES
7149 1C33 190B A          JMP     USEDSK
7150 1C34 F0D0*A          SKNE    AC0,=0D
7151 1C35 1909 A          JMP     USEDSK
7152 1C36 F111*A          SKNE    AC0,='N'/256  ; 'N' = NO
7153 1C37 1901 A          JMP     .+2
7154 1C38 19F3 A          JMP     DSKASK
7155 1C39 5000 A NODSK:   LI      AC0,0         ; REMOVE DISK COMMANDS
7159 1C3A B10E*A          ST      AC0,DSKCOM    ;   FROM END OF TABLE
7160 1C3B C10E*A          LD      AC0,=XQT81    ; MAKE DISK I/O ROUTINE AREA
7161 1C3C D033 A          ST      AC0,ESTART    ;    PART OF EDIT BUFFER
7163 1C3D 190F A          JMP     CSIZE4
7164 1C3E D000 A AMON:    .WORD   0D000         ; DOS FIRMWARE START ADDR
7166 1C3F C10B*A USEDSK:  LD      AC0,=-1025    ; SAVE AREA FOR DISK FIRMWARE
7167 1C40 D05C A          ST      AC0,OP1       ;    INFO & BAD SECTOR TABLE
7168 1C41 190B A          JMP     CSIZE4
7173 1C42 1BF3 T          .POOL   11
     1C43 1141 T
     1C44 10CF T
     1C45 1C9E T
     1C46 7E59 A
     1C47 0059 A
```

```
     1C48 004E A
     1C49 1795 T
     1C4A 1A42 T
     1C4B FBFF A
     1C4C 0000 A·
7174  ;
7175 1C4D 949F*A CSIZE4:  JSR     CRLF
7176 1C4E C438 A          LD      AC1,LAST
7177 1C4F E45C A          ADD     AC1,OP1       ; SAVE LAST MEMORY LOCS FOR FIRM
7178 1C50 D438 A          ST      AC1,LAST
7179 1C51 D451 A          ST      AC1,SY2
7180 1C52 D439 A          ST      AC1,TOPRAM
7181 1C53 C033 A          LD      AC0,ESTART    ; MAKE SURE STILL HAVE ROOM
7182 1C54 94A4*A          JSR     SKL
7183 1C55 191F A          JMP     NOROOM
7184 1C56 94A1*A ASK:     JSR     MESG          ; ASK FOR MEMORY SIZE (PORTION)
7185 1C57 1C98 T          .WORD   ASKSIZ
7186 1C58 159A A          JSR     RDSCAN
7187 1C59 19FC A          JMP     ASK           ;    ERROR
7188 1C5A 19FB A          JMP     ASK           ;    CTL-C
7189 1C5B 99F0*A          JMP     SCRATCH       ; BLANK LINE - USE CALCULATED VA
7190 1C5C 5015 A          LI      AC0,21        ; CHECK FOR NUMBER
7191 1C5D F041 A          SKNE    AC0,CLASS
7192 1C5E 1901 A          JMP     $NUM
7193 1C5F 19F6 A          JMP     ASK
7194 1C60         $NUM:   DLD     AC0,VALUE     ; NUMBER ENTERED
7195 1C62 94B5*A          JSR     ROUND
7196 1C63 5C00 A          NOP
7197 1C64 D065 A          ST      AC0,TEMP
7198 1C65 94A3*A          JSR     SCAN          ; NUMBER MUST BE FOLLOWED BY
7199 1C66 19EF A          JMP     ASK           ;    CARRIAGE RETURN
7200 1C67 500D A          LI      AC0,13
7201 1C68 F041 A          SKNE    AC0,CLASS
7202 1C69 1901 A          JMP     .+2
7203 1C6A 19EB A          JMP     ASK
7204 1C6B C033 A          LD      AC0,ESTART    ; MAKE SURE ESTART < TEMP
7205 1C6C C465 A          LD      AC1,TEMP
7206 1C6D 94A4*A          JSR     SKL
7207 1C6E 19E7 A          JMP     ASK
7208 1C6F C065 A          LD      AC0,TEMP      ; MAKE SURE TEMP <= LAST
7209 1C70 C438 A          LD      AC1,LAST      ;    (IF NOT, USE LAST)
7210 1C71 94A4*A          JSR     SKL
7211 1C72 99D9*A          JMP     SCRATCH
7212 1C73 D038 A          ST      AC0,LAST
7213 1C74 99D7*A          JMP     SCRATCH
7214 1C75 5000 A NOROOM:  LI      AC0,0         ; AREA ERROR
7215 1C76 94C3*A          JSR     ERROR
7216 1C77 198C A          JMP     ENTRY
7217  ;
7218  ;
7219  ;        INITIALIZATION MESSAGES
7220  ;
7221      8001 A          .LIST   LOPT
7222 1C78 0D0A A MSG1:    .WORD   0D0A,0A0A
```

```
 7223 1C7A 4E41 A          .ASCII  'NATIONAL BASIC'
 7224 1C81 0D0A A          .WORD   0D0A
 7225 1C82 5245 A          .ASCII  'REV A  11 FEB 1977'
 7226 1C8B 0D0A A          .WORD   0D0A,0A00
 7228 1C8D 0D0A A ASKDSK:  .WORD   0D0A          ; 'DISK FILES NEEDED?'
 7229 1C8E 4469 A          .WORD   04469,0736B,02046,0696C,06573,0204E
 7230 1C94 6565 A          .WORD   06565,06465,0643F,0
 7232               ASKSIZ: ; 'MEMORY SIZE'
 7233 1C98 4D65 A          .WORD   04D65,06D6F,07279,02053,0697A,06500
 7234  ;
 7254 1C9E 0000 A EBUF1:   .WORD   0             ; START MEMORY CHECK FROM HERE
 7256  ;
 7257      1C04 T          .END    ENTRY
```

```
01A7 T  08E0 T    025A T  1A70 T    025B T  7E3B A    02DF T  8048 A
02E0 T  7E3B A    03C9 T  1638 T    0410 T  19B2 T    0423 T  002E A
0424 T  19C3 T    0425 T  0427 T    0426 T  19AF T    076D T  0024 A
07E0 T  0CCB A    0D90 T  0A81 T    0E4E T  0CE3 T    0EF3 T  0BAE T
0FB5 T  180C T    1179 T  1277 T    11F9 T  134E T    1283 T  0017 A
12D8 T  121C T    14FA T  1219 T    14FB T  120A T    1C4C T  0114 T
```

A	12E4	T	ABS	112D	T	AC0	0000	A	AC1	0001	A
AC2	0002	A	AC3	0003	A	ADDTT3	1170	T	ADSHR	1210	T
ALAST	0079	A	ALOC	025C	T	ALOCM	1893	T	AMON	1C3E	T
ANEXT	0055	A	ARCTAN	1403	T	ARGERR	133B	T	ASCFCN	0FEE	T
ASK	1C56	T	ASKDSK	1C8D	T	ASKSIZ	1C98	T	ATAN	1437	T
ATDATA	1563	T	ATLOC	18B1	T	ATN0	1407	T	ATN1	1424	T
ATN2	1428	T	ATN3	1420	T	ATN4	1421	T	AUTO	0140	T
AY1	142F	T	AY2	1431	T	B	0020	A	BACK	02BD	T
BCDADD	151B	T	BEGIN	0125	T	BIG	0001	A	BIT0	0003	A
BIT1	0004	A	BLANK	0020	A	BREAK	022A	T	BRK	188F	T
BRKFCN	1143	T	BRKFLG	0021	A	BRKMOD	0017	A	BYE	024D	T
C	12E5	T	CALCUL	0D91	T	CARRY	0007	A	CHKLC	0589	T
CHKMOD	0921	T*	CHKTER	07C9	T	CHKTKN	08FB	T	CHRFCN	10A7	T
CKCR	0539	T	CLASS	0041	A	CLCSUB	0CF3	T	CLOSE	D416	A
CLRST1	0109	T	CLRSTK	0459	T	CMDTAB	185E	T	COL	003F	A
COMMA	002C	A	CONST	07E1	T	CONT	023D	T	COPYS	0E7D	T
COS	1314	T	COSA	1523	T	COUNT	007A	A	CRLF	036B	T
CRLF1	02C9	T	CRY	000A	A	CSIZE	1C1D	T	CSIZE2	1C2A	T
CSIZE4	1C4D	T	CTRL	091A	T	CURLIN	0014	A	D2C	120A	T
DEBUG	0000	A	DELETE	04F2	T	DERROR	D430	A	DIGIT	004A	A
DISKIN	1A70	T	DPTR	0054	A	DRESET	D406	A	DSCAN1	0EF4	T
DSCAN2	0EF6	T	DSHL	1219	T	DSIGN	0068	A	DSKASK	1C2C	T
DSKBUF	1AF3	T	DSKCOM	1795	T	DSKERR	1A6B	T	DSTART	0053	A
DTEMP	0067	A	DUMP	0000	A	E2S	1266	T	EBUF1	1C9E	T
ECUR	0034	A	EDEL	04B7	T	EEND	0035	A	EFIND	050A	T
ELEN	0036	A	ENDBUF	0020	A	ENTRY	1C04	GT	EPSER	12E6	T
EPUSH2	0CB7	T	EPUSH3	0CB8	T	EQUAL	003D	A	ERM1	1886	T
ERM2	188A	T	ERRFLG	0019	A	ERROR	03CA	T	ERRTAB	1638	T
ESIGN	004E	A	ESTART	0033	A	ESTOR	04AF	T	EXEC	0131	T
EXPA	1398	T	EXPB	139A	T	EXPC	139C	T	EXPD	139E	T
EXPERR	0CAF	T	EXPN	0049	A	EXPON	1357	T	F10	0838	T
FALSE	111D	T	FINDCR	052F	T	FIRST	0100	T	FL1	00DC	A
FL10	147A	T	FLADD	11C7	T	FLAND	10F3	T	FLCMP	10E4	T
FLDEC	1445	T	FLDIV	117A	T	FLEQ	110D	T	FLGE	1114	T
FLGT	1110	T	FLINT	128A	T	FLLE	111A	T	FLLT	1117	T
FLMAX	1123	T	FLMIN	1128	T	FLMOD	11FA	T	FLMULT	114C	T
FLNE	110A	T	FLNORM	121C	T	FLNOT	10EF	T	FLOR	10F7	T
FLSIGN	0075	A	FLSUB	11C5	T	FLXOR	10FC	T	FNERR	0CC7	T
FNERR2	104A	T	FNLINE	0052	A	FORPTR	0DD3	T	FORSAV	0DDF	T
FRANGE	1271	T	FRMULT	14FC	T	FRNORM	123F	T	FRPN	1247	T
FRSIGN	0076	A	FRTN	116C	T	FST	0087	A	G	0061	A
GECO	7E59	A	GETARY	0CE3	T	GETC	7E3B	A	GETCH	0576	T
GETDST	078A	T	GETFCN	0FD6	T	GETHX	087B	T	GETS	0565	T
GETS2	0568	T	GETS3	057D	T	GETSTR	0796	T	GETSUB	0CC9	T
GETUC	0572	T	GETVAL	0D93	T	GTDATA	0EC7	T	HEXFCN	10A2	T
HLIST	02E1	T	HSPRT	0402	T	IDENT	0045	A	IEXP	121D	T
IFIX	12AD	T	IFIX2	12B0	T	IMP16	0000	A*	INBUF	18BE	T
INERR	18A0	T	INLINE	03E5	T	INP	1137	T	INPUT	0E85	T
INTEST	7ECC	A	INTFL	1284	T	KEY	071C	T	L2E	13A0	T
LAST	0038	A	LEFFCN	0F92	T	LENFCN	0FEB	T*	LINFCN	0C40	T
LINIT	08E0	T	LINK	0008	A	LINLEN	001E	A	LIST	033B	T
LIST1	0062	A	LIST2	0063	A	LISTIT	02FE	T	LN	131D	T
LN2	133D	T	LNDATA	153A	T	LOAD	0133	T	LOOKUP	089D	T
LOPT	8001	A	LPAREN	0C88	T	LSTART	003E	A	LZONE	001F	A

MAXSTR	0059	A	MAXVAL	1277	T	MERGE	0135	T	MESG	0411	T
MESG3	0417	T	MIDFCN	0F80	T*	MODE	0013	A	MODEIN	0158	T
MODX	1482	T	MOREIN	18AB	T	MSG1	1C78	T	MSZ1	1874	T
MSZ3	1877	T	MSZ4	187B	T	MULT	14FE	T	NDATA	0EB7	T
NEG	000B	A	NEGATE	1207	T	NEWLIN	0184	T	NEXT1	01BF	T
NEXT2	020F	T	NEXTAD	0040	A	NEXTIN	0159	T	NODSK	1C39	T
NOGO	0BAE	T	NOROOM	1C75	T	NOUSER	1141	T	NPZ	131B	T
NPTR	0049	A*	NSTRC	005A	A	NSTRW	005B	A	NTYPE	003A	A
NWDS	000F	A	NWTN	13CB	T	NXTLIN	0037	A	NXTRES	066C	T
NXTSTA	0930	T	NXTSYN	08F0	T	NZ1	0BF8	T	NZZ	0BFA	T
NZRO	0005	A	OP1	005C	A	OP2	005E	A	OP3	0060	A
OPENR	D40E	A	OPENW	D410	A	OPER	0D47	T	OPERM	18B8	T
OPERR	0CE0	T	OPPUSH	0CAA	T	OUT1	038F	T	OUT2	03ED	T
OUTCOL	001D	A	OUTDVC	001C	A	OUTDVX	001B	A	OVF	000C	A
OVFERR	1276	T	OVFLW	0006	A	P2	1433	T	P4	1435	T
PACE	0001	A	PACK	067E	T	PASS2	01CD	T	PASS2B	0205	T
PBUF	1943	T	PCLAST	168A	T	PCTAB	166E	T	PEEK	1134	T
PFLAG	004F	A	PGFR	D404	A	PLC	004C	A	POINT	187F	T
POS	0002	A	POSFCN	1132	T	POWER	13E0	T	PREC	0044	A
PROT	0011	GA	PSCAN	0594	T	PTAPE	013C	T	PULL	0496	T
PUNCH	02EA	T	PUSH	0488	T	PUTC	7E44	A	PUTD	14F2	T
PUTDC	1AE1	T	PUTS	0542	T	PUTW	03F3	T	PUTW4	03F1	T
PWR2	123A	T	PWR6	1476	T	PWR7	1478	T	RCHAR	1A8E	T
RDLINE	027A	T	RDSCAN	1BF3	T	READIN	0153	T	READY	186D	T
REMCHA	0027	A	RESET	0117	T	RESTAB	168C	T	RESTOR	0479	T
RESULT	0D39	T	RIGFCN	0F98	T	RND	12D9	T	RNDNUM	0086	A
ROM	0000	A	ROUND	12A6	T	RPAREN	0D43	T	RSCAN	0E54	T
RSDATA	0F2E	T	RSLTO	134E	T	RSLT1	1400	T	RUN	01A8	T
RUNERR	0124	T	RUNFLG	0018	A	RWORD	D422	A	SA1	13D5	T
SA2	13D9	T	SAVE	0468	T	SAVFIL	02F3	T	SB1	13D7	T
SB2	13DB	T	SCALE	146E	T	SCAN	0690	T	SCAN2	06A3	T
SCOL	0059	A	SCR	007B	A	SCRA	007D	A	SCRATC	0114	T
SCRB	007F	A	SDATA	0EC0	T	SETUP	11A7	T	SETUP2	11B2	T
SFL	0000	A	SGN	1130	T	SIGN	10E5	T	SINE	1311	T
SIZE	10B3	T	SK0	002A	A	SK1	002B	A	SK2	002C	A
SKIPPD	0E0B	T	SKL	0527	T	SKRET	002D	A	SPCFCN	0C35	T
SQ2	13C1	T	SQODD	13B4	T	SQRT=	13A2	T	SSCAN	0E4F	T
SSTART	0057	A	ST01	095E	T	ST02	0979	T	ST03	09CE	T
ST04	09D2	T	ST05	09D9	T	ST06	09DD	T	ST07	09E5	T
ST08	09E9	T	ST09	09EE	T	ST10	09F3	T	ST11	09F7	T
ST12	09FB	T	ST13	0A01	T	ST14	0A0C	T	ST15	0A14	T
ST16	0A16	T	ST17	0A18	T	ST18	0A1A	T	ST19	0A1B	T
ST20	0A1E	T	ST21	0A21	T	ST22	0A27	T	ST23	0A2B	T
ST24	0A32	T	ST25	0A34	T	ST26	0A36	T	ST27	0A3A	T
ST28	0A3C	T	ST29	0A46	T	ST30	0A48	T	ST31	0A4A	T
ST32	0A51	T	ST33	0A55	T	ST34	0A5B	T	ST35	0A77	T
ST36	0A79	T	ST37	0A7B	T	ST38	0A7D	T	ST39	0A81	T
ST40	0A84	T	ST41	0A99	T	ST42	0A9B	T	ST43	0A9F	T
ST44	0AA1	T	ST45	0AA3	T	ST46	0AA7	T	ST47	0AAD	T
ST48	0ABA	T	ST49	0ABC	T	ST50	0ABE	T	ST51	0AC0	T
ST52	0AC4	T	ST53	0AD0	T	ST54	0AD7	T	ST55	0ADB	T
ST56	0AE2	T	ST57	0AE6	T	ST58	0AEA	T	ST59	0AEF	T
ST60	0AF9	T	ST61	0B01	T	ST62	0B05	T	ST63	0B09	T
ST64	0B0D	T	ST65	0B14	T	ST66	0B18	T	ST67	0B1C	T

ST68	0B1F	T	ST69	0B26	T	ST70	0B50	T	ST71	0B2B	T
ST72	0B30	T	ST73	0B32	T	ST74	0B36	T	ST75	0B43	T
ST76	0B45	T	ST77	0B4A	T	ST78	0B4C	T	ST79	0B4E	T
ST80	0B52	T	ST81	0B54	T	ST82	0B97	T	ST83	0B5E	T
ST84	0B62	T	ST85	0B69	T	ST86	0B6D	T	ST87	0B6F	T
ST88	0B76	T	ST89	0B7A	T	ST90	0B7E	T	ST91	0B81	T
ST92	0B86	T	ST93	0B88	T	ST94	0B8C	T	ST95	0B91	T
ST96	0B93	T	ST97	0AC6	T	ST98	0AC8	T	ST99	0ACC	T
STACK	19AF	T	STACK0	19C3	T	STACK1	19E3	T	STACK2	19F2	T
STACK3	1A10	T	STACK4	1A24	T	STACK5	1A2E	T	STACK6	1A38	T
STACKX	1A42	T	START	0095	A	STFIRS	04A7	T	STK5	002E	A
STKINT	0427	T	STKPTR	0022	A	STKSTR	0F60	T	STLAST	049F	T
STLIN	0189	T	STOP	0235	T	STOPM	188C	T	STPTR	0023	A
STR3	1986	T	STRFCN	109E	T*	STROP	0F66	T	STRTMP	0270	T
STYPE	003B	A	SUBERR	0D10	T	SVSCAN	003C	A	SY1	0038	A
SY2	0051	A	SYMA	0043	A	SYMB	0042	A	SYNERR	0917	T
SYNRTN	0050	A	SYNTAB	095C	T	SYNTAX	08E5	T	SYSER2	0D83	T
SYSERM	18AC	T	SYSERR	0D81	T	TAB	0C2B	T	TAB1	1582	T
TAB2	1556	T	TAB3	1818	T	TAB3A	180C	T	TAB3L	183C	T
TAB4	183E	T	TAB4L	185C	T	TABFCN	0C24	T*	TAC0	006D	A
TAC1	006E	A	TAC2	006F	A	TAC3	0070	A	TAN	1303	T
TECHO	02D0	T	TEMP	0065	A	TENTH	147C	T	TLIST	02FB	T
TLOC	025E	T	TMPADR	0064	A	TNEXT	0056	A	TOPRAM	0039	A
TRCOFF	10C9	T	TRCON	10C7	T	TRMODE	0015	A	TRUE	1120	T
TT1	0081	A	TT2	0082	A	TT3	0084	A	TWIDTH	10CF	T
TWOPI	1319	T	TXTMOD	0016	A	TYPFLG	0069	A	TYPLIN	0375	T
TYPLSP	0376	T	UNIT	001A	A	USEDSK	1C3F	T	USER	0010	GA
USR	113D	T	VALFCN	0FF5	T	VALTAB	17A3	T	VALUE	0047	A
WARNIN	03DB	T	WARNM	1881	T	WCHAR	1AA7	T	WWORD	D424	A
X	0071	A	XCRLF	0366	T	XOPER	0D74	T	XOVER	133F	T
XQT	0225	T	XQT1	0B9B	T	XQT11	0BEC	T	XQT12	0C77	T
XQT13	0C01	T	XQT14	0C19	T	XQT15	0C44	T	XQT16	0BDB	T
XQT17	0BD7	T	XQT18	0C4E	T	XQT19	0C5D	T	XQT2	0BA2	T
XQT20	0C61	T	XQT21	0C7E	T	XQT22	0C87	T	XQT26	0C8E	T
XQT27	0C95	T	XQT28	0CA1	T	XQT29	0CB1	T	XQT3	0BA9	T
XQT30	0CBB	T	XQT32	0D29	T	XQT33	0D45	T	XQT34	0DA6	T
XQT35	0DB0	T	XQT36	0DB7	T	XQT37	0DBB	T	XQT38	0DC7	T
XQT39	0DCC	T	XQT4	0BB0	T	XQT40	0DE9	T	XQT41	0DEE	T
XQT42	0DF3	T	XQT43	0DF7	T	XQT44	0E1F	T	XQT48	0E59	T
XQT49	0E62	T	XQT5	0BBF	T	XQT50	0E65	T	XQT51	0E5F	T
XQT53	0F26	T	XQT54	0F33	T	XQT55	0F3E	T	XQT56	0F47	T
XQT57	0F53	T	XQT58	0F55	T	XQT59	0F58	T	XQT6	0BC8	T
XQT60	0C9E	T	XQT61	0FB6	T	XQT62	0FDE	T	XQT63	1010	T
XQT64	1018	T	XQT65	101E	T	XQT66	105D	T	XQT67	1024	T
XQT68	104C	T	XQT7	0BCD	T	XQT70	0E6B	T	XQT71	0D27	T
XQT73	0CEA	T	XQT74	108E	T	XQT75	0D12	T	XQT76	0C80	T
XQT77	10CC	T	XQT78	1057	T	XQT79	105B	T	XQT8	0BD2	T
XQT80	10DC	T	XQT81	1A42	T	XQT9	0BD5	T	XQTFLG	0012	A
XTEMP	0080	A	XX	0077	A	Y	0073	A	Y2	0074	A
ZONE	0BF0	T	ZR0	0001	A	$0	0D9F	T	$0	1002	T
$1	033F	T	$1	0530	T	$1	0BBA	T	$1	0E8A	T
$1	112A	T	$1	1161	T	$1	11A0	T	$1	1243	T
$1	14E0	T	$1A	0346	T	$1A	1324	T	$2	0534	T
$2	0906	T	$2	0E29	T	$2	0E90	T	$2	1125	T

$2	115F	T	$2	119B	T	$2	1248	T	$2	14E5	T
$2A	0E99	T	$2B	0E9C	T	$2BCD	147E	T	$3	118D	T
$4	11A2	T	$4	124E	T	$44	1081	T	$7	0F71	T
$8002	040D	T	$8006	040C	T	$A1	016A	T	$A1	01D2	T
$A2	01EB	T	$A3	01F2	T	$A4	01FC	T	$ADD	08B9	T
$AER4	026E	T	$AERR	0200	T	$ARY	01E9	T	$AY	06FC	T
$AY2	06FF	T	$BEGI	0211	T	$BL	1AA3	T	$BL	1ABF	T
$BOFF	1148	T	$BOTH	055A	T	$BS	03AB	T	$BYER	0254	T
$CAT	0C6D	T	$CE	0D03	T	$CH	1A9C	T	$CHK	0177	T
$CHK1	14A8	T	$CHKD	14AA	T	$CK1A	0778	T	$CKAR	0700	T
$CKE	1230	T	$CKGO	0652	T	$CKP	0746	T	$CKZ	122D	T
$CLOO	0FC5	T	$CLR	0269	T	$CNT	0242	T	$CPY	0FA2	T
$CPY2	0FA4	T	$CPYN	0E74	T	$CPYS	0E7B	T	$CR	03A8	T
$CR	1AC6	T	$CR2	1ACB	T	$CR3	1ACE	T	$CRLF	02AF	T
$CRRE	1A80	T	$CTL2	0936	T	$D0	07EF	T	$D1	06A9	T
$D1A	080D	T	$D2	085D	T*	$D3N	086B	T	$D3N1	086D	T
$D3P	0864	T	$D4	0873	T	$D5	0875	T	$DEC	14F0	T
$DEL1	04F8	T	$DEL2	04FC	T	$DIM	0DBF	T	$DN	04BD	T
$DONE	085E	T	$DONE	14E9	T	$DS1	0F06	T	$DS2	0F08	T
$DS3	0F15	T	$DS4	0F18	T	$DS5	0F23	T	$DUP	1022	T
$E	07D5	T	$EF2	1AD8	T	$EF3	1ADD	T	$EMP	0431	T
$EMP	0D5F	T	$END	1A55	T	$ENDD	0330	T	$ENDL	0322	T
$EOF	0165	T	$EOF	1A83	T	$EOF	1AD2	T	$ERR	0232	T
$ERR	02C4	T	$ERR	08DE	T	$ERR	0916	T	$ERR2	0879	T
$ESC	02B9	T	$ETX	02B3	T	$ETYP	14CD	T	$EXP1	1365	T
$F0	1295	T	$F1	0D2F	T	$F1	1299	T	$F1	1A69	T
$F1A	129E	T	$F2	0D35	T	$F2	12A3	T	$F2	1A6A	T
$F3	0D75	T	$F3	1297	T	$F4	0D7B	T	$FIN	04EA	T
$FIX	0FAB	T	$FLA1	11E7	T	$FLD	117F	T	$FLT1	07FC	T
$FLT2	0801	T	$FN	070E	T	$FN	1290	T	$FND	0524	T
$FND	08B6	T	$FNER	1047	T	$FOR	0518	T	$FOUN	0781	T
$FR1	081F	T	$FSV	0DE7	T	$FULL	0440	T	$G1	0ECE	T
$G2	0EDA	T	$G3	0EDE	T	$G4	0EE0	T	$GE	0FCB	T
$GET	0290	T	$GT	0FD3	T	$HEX	05D4	T	$HX1	05D9	T
$HX2	05DC	T	$I1	12C1	T	$I2	12C3	T	$I3	12C4	T
$I4	12C9	T	$I5	12CB	T	$IN	04C5	T	$IN	0752	T
$IN	0E67	T	$IN	113A	T	$IN1	0289	T*	$IN1	04C4	T
$IN2	0E6F	T	$INER	0EA4	T	$JUMP	0130	T	$K0	0348	T
$K1	034A	T	$K1	071F	T	$K2	034E	T	$K2	0725	T
$K3	072B	T	$K3A	072E	T	$K4	0734	T	$K5	0736	T
$K6	0738	T	$KB	0294	T	$KGO	0754	T	$KGSU	0760	T
$KGTO	075E	T	$L	1A97	T	$L1	0404	T	$L1	06C5	T
$L1	08A8	T	$L2	06CA	T*	$L2A	06D1	T	$L2B	06D8	T
$L3	06DE	T	$LC1	06E2	T	$LC2	06EF	T	$LDOL	06E4	T
$LF	1AC2	T	$LINE	007B	A	$LIST	030D	T	$LOOP	0519	T
$LOOP	12F7	T	$LOOP	150C	T	$LP	0D5D	T	$LP1	0442	T
$LP1	06E0	T	$LP2	0448	T	$LP2	06EC	T	$LP3	0453	T
$LSP	06E7	T	$LSPE	08CC	T	$M	0903	T	$M1	0418	T
$M1	1505	T	$M2	150A	T	$MAXP	127E	T	$MINU	07D7	T
$MORE	0EAF	T	$MSGE	0422	T	$N	0900	T	$N1	1457	T
$N2	1463	T	$NC	0808	T	$NCO	07EA	T	$NC2	0848	T
$NC3	084D	T	$ND2	083A	T	$NERR	0E27	T	$NEXT	0952	T
$NO	0DAE	T	$NOLI	03EB	T	$NORM	06F7	T	$NORM	1455	T
$NOT	08DB	T	$NOTD	081C	T	$NOTF	04EE	T	$NOTF	077E	T

$NUM	1005	T	$NUM	1C60	T	$NXA	01D8	T	$NXLI	0E47	T
$NXTC	1A71	T	$01	0394	T	$0K	0450	T	$0P1	0D56	T
$0P1	1101	T	$0P2	0D66	T	$0P2	1105	T	$0P3	0D6E	T
$0UT	0EEC	T	$0UT	1083	T	$0UT	1AB9	T	$0VER	02B1	T*
$0VER	0DDB	T	$0VF	089A	T	$0VF	11EE	T	$0VF	1261	T
$P	144A	T	$P0	05AA	T	$P00	05AF	T	$P1	03AE	T
$P1	05B2	T	$P2	05C1	T	$P2	0DDD	T	$PAK	0680	T
$PD	03BE	T	$PDIG	14C4	T	$PER	07D4	T	$PERR	0494	T
$PK2	0687	T	$PK3	068D	T	$PL1	05E3	T	$PL1A	05F0	T
$PL2	05F7	T	$PL2A	0604	T	$PL2B	0611	T	$PL3	0615	T
$PL3A	061D	T	$PL3B	0622	T	$PL3C	0642	T	$PL3R	0630	T
$PL4	0647	T	$PL00	0C0C	T	$PLUS	07D6	T	$PM	13F9	T
$PMUL	13F4	T	$POKE	107D	T	$POP	093E	T	$PP1	05C6	T
$PR	03B6	T	$PRT	14B3	T	$PS	03B8	T	$PSER	13ED	T
$PT	03B4	T	$PT0	14EC	T	$PUSH	0946	T	$PUT	0CF0	T
$PW	03F7	T	$PW1	03F6	T	$R	1AE9	T	$R1	0745	T
$RD1	027C	T	$RD2	0281	T	$REM	0677	T	$REST	0436	T
$RIGH	0555	T	$RND	074F	T	$ROUN	1257	T	$ROUN	149E	T
$RST	0561	T	$RTN	08B8	T	$RTN	0C3F	T	$RTN	0C73	T
$RTN	0CDF	T	$RTN	0F3D	T	$RTN	10FA	T	$RTS	0DF2	T
$RTS1	1ABC	T	$RVAL	1149	T	$S1	0101	T	$S1	02AA	T
$S1	11B9	T	$SZ	010D	T	$SZ	11BF	T	$SAVE	06F8	T
$SCMP	0FD4	T	$SEND	0C3C	T	$SEQN	06AE	T	$SERR	019A	T
$SERR	07C3	T	$SFUL	0957	T	$SIN	0EAB	T	$SOVF	0484	T
$SPEC	076E	T	$SPER	1ABD	T	$SQ1	06B0	T	$SQ2	06B5	T
$SQ3	06BD	T	$SQ4	06BE	T	$ST	04E4	T	$ST1B	0486	T
$ST1L	0487	T	$STER	04F0	T	$STK	0F59	T	$STOP	0EB2	T
$STOR	04DA	T	$STR	01E2	T	$STR1	079B	T	$STR2	07A8	T
$STR2	109F	T	$STR3	07B0	T	$STR4	07B9	T	$STRD	0E7A	T
$STRN	05CB	T	$SUNF	0485	T	$T1	037B	T	$T1	0C26	T
$T2	0364	T	$TZ	0385	T	$T2	0C31	T	$T3	0388	T
$T4	038E	T	$TAB	035Z	T	$TEMP	0070	A	$TERR	0EBD	T
$TR	021F	T	$TRIM	0596	T	$TTY	02D5	T	$UNFL	043E	T
$UNP	1A47	T	$UP	04D3	T	$V2	0CFD	T	$VAL	0FEC	T
$VLER	05C9	T	$W1	0F38	T	$WAIT	1088	T	$X	10EB	T
$X1	0882	T	$X1	11D4	T*	$X2	088B	T	$X2	11DF	T
$X3	0887	T	$X3	11E0	T	$X4	0892	T	$XCH	11C2	T
$Z	10EA	T									

NO ERROR LINES
SOURCE CHECKSUM= B00B
OBJECT CHECKSUM= 2960

INPUT FILE 1:BASIC.SRC ON BJE P3
LISTING FILE 2:BASIC.LST ON BJE P4
OBJECT FILE 1:BASIC.LM ON BJE P3