```
002                             ORG   :E000
003                     *
004                     *
005                     *
006                     *   ============================
007                     *** ENCODING / UTILITY PACKAGES ***
008                     *   ============================
009                     *
010                     * Called by RST 1; DATA XX. XX indicates the
011                     * offset of E000 for the different entrypoints.
012                     *
013                     * Contains also the key encoding routines and the
014                     * Heap organisation routines.
015                     *
016                     ****************
017                     * ENTRYPOINTS *
018                     ****************
019                     *
020 E000 C324E0         ELINE   JMP   :E024     Encode a BASIC line
021 E003 C32AE7         ELN     JMP   :E72A     Encode a linenr
022 E006 C345E1         ETCON   JMP   :E145     Encode a constant
023 E009 C374EA         USTART  JMP   :EA74     Start Utility package
024 E00C C390EF         DBOOT   JMP   :EF90     Disc bootstrap
025 E00F C39CE9         MHREQ   JMP   :E99C     Heap request
026 E012 C33FE9         MINKEY  JMP   :E93F     Encode a key input
027 E015 C335E9         L3E394  JMP   :E935     Get inputs from keyb
028                                               or DINC
029                     *
030                     *   ================
031                     *** ENCODING PACKAGE ***
032                     *   ================
033                     *
034                     * Generally in the encoding routines, HL points to
035                     * the first free location in the EBUF. C points to
036                     * the input text line.
037                     *
038                     ************************
039                     * UPDATE EBUF POINTER *
040                     ************************
041                     *
042                     * Increments the input pointer and checks if the
043                     * encoded input buffer (EBUF) is full.
044                     *
045                     * Entry: HL: input pointer EBUF.
046                     * Exit:  HL: updated pointer.
047                     *        Other registers preserved.
048                     *
049 E018 23              INXCH   INX   H         Incr pointer
050 E019 F5                     PUSH  PSW
051 E01A 7D                     MOV   A,L       Get lobyte in A
052 E01B FEBE                   CPI   :BE       Max value reached?
053 E01D 3E1A                   MVI   A,:1A     Buffer full ?
054 E01F D2F5D9                 JNC   :D9F5     Then run error 'LINE TOO
055                                               COMPLEX'
056 E022 F1                     POP   PSW
057 E023 C9                     RET
058                     *
059                     ****************************
060                     * ENCODE A BASIC COMMAND *
061                     ****************************
062                     *
                        * Looks for a match between input on line and the
```

```
064                    * BASIC command table #CBBF. Error exit 'COMMAND
065                    * INVALID' if the 2 high order bits of the code in
066                    * the table don't match the mask in D. Else the
067                    * code (#80 + low order 6 bits from code in table)
068                    * are stored in EBUF.
069                    *
070                    * Entry: HL: 1st free position in EBUF.
071                    *        D : Mask for direct command (#80) or
072                    *            program input (#40).
073                    *        C : Position on current line.
074                    * Exit:  HL: Updated.
075                    *        C : Points to 1st not used input in line.
076                    *        E : Code just stored.
077                    *        BD preserved, AF corrupted.
078                    *        Exit with jump to address found in table.
079                    *        #E04F is on stack as next returnaddress,
080                    *        entry DE is next on stack.
081                    *
082 E024 D5            L3E2    PUSH  D
083 E025 E5                    PUSH  H
084 E026 214FE0                LXI   H,:E04F   Returnaddr after finishing
085                                             encoding
086 E029 E3                    XTHL            Save it on stack
087 E02A E5                    PUSH  H         Save HL again
088 E02B 1E02                  MVI   E,:02     Addit. offset for finding
089                                             instruction in table
090 E02D 21BFCB                LXI   H,:CBBF   Startaddr table with strings
091                                             Basiccommands
092 E030 D5                    PUSH  D
093 E031 CD34CA                CALL  :CA34     Find instr in table
094 E034 D1                    POP   D         Get mask for direct/program
095 E035 7E                    MOV   A,M       Get instr code from table
096 E036 E6C0                  ANI   :C0       ) Check if it
097 E038 A2                    ANA   D         ) is a valid command
098 E039 3E18                  MVI   A,:18     If not: Run error
099 E03B CAF5D9                JZ    :D9F5     'COMMAND INVALID'
100 E03E 7E                    MOV   A,M       Get instr code from table
101 E03F 23                    INX   H
102 E040 E63F                  ANI   :3F       ) Make it a token
103 E042 F680                  ORI   :80       )
104 E044 5F                    MOV   E,A       Store token in E
105 E045 7E                    MOV   A,M       )
106 E046 23                    INX   H         ) Get addr of encoding
107 E047 66                    MOV   H,M       ) instruction in HL
108 E048 6F                    MOV   L,A       )
109 E049 E3                    XTHL            and save it on stack
110 E04A 73                    MOV   M,E       Store token in EBUF
111 E04B CD18E0                CALL  :E018     Update EBUF pointer
112 E04E C9                    RET             Go to encoding instruction
113                                             return afterwards to E04F
114                    *
115                    * End of encoding of an input line (after running
116                    * the command specific processing).
117                    * Checks if character after BASIC command is CR or
118                    * ':'.
119                    *
120                    * Entry: C : points to position on current line.
121                    *        On stack: Original DE.
122                    *
123 E04F D1            L3E3    POP   D         Get type of command in D
124 E050 CDD2DD                CALL  :DDD2     Get char from line, neglect
125                                             tab + space
```

```
126 E053 0C                 INR   C          Pntr to next char
127 E054 FE3A               CPI   :3A        ':' ?
128 E056 CA24E0             JZ    :E024      Then encode next instr
129 E059 FE0D               CPI   :0D        'CR' ?
130 E05B C20BDA             JNZ   :DA0B      Run 'SYNTAX ERROR' if not
131 E05E C9                 RET
132                         *
133                         ********************************
134                         * ENCODE 'FOR - TO - (STEP)' *
135                         ********************************
136                         *
137                         * Valid for all specific encoding routines:
138                         * HL points to 1st free EBUF location.
139                         *
140 E05F CDFEE0             EFOR  CALL  :E0FE      Encode 'LET'
141 E062 FE20               CPI   :20        String type ?
142 E064 CA1ADA             JZ    :DA1A      Then run error 'TYPE
143                                           MISMATCH'
144 E067 FE10               CPI   :10        INT type ?
145 E069 1188E0             LXI   D,:E088    Addr 'TO' table
146 E06C C39AE0             JMP   :E09A      Get addr encoding instr and
147                                           go to it
148
149                         * Encode 'TO':
150
151 E06F CC6DE3             L3E5  CZ    :E36D      Encode a INT expr
152 E072 C476E3                   CNZ   :E376      Encode a FPT expr
153 E075 1190E0                   LXI   D,:E090    Addr 'STEP' table
154 E078 C39AE0                   JMP   :E09A      Get addr encoding instr and
155                                                go to it
156
157                         * Encode 'STEP':
158
159 E07B CC6DE3             L3E6  CZ    :E36D      Encode a INT expr
160 E07E C476E3                   CNZ   :E376      Encode a FPT expr
161 E081 C9                       RET
162
163                         * End encoding:
164
165 E082 36FF               L3E363 MVI  M,:FF      FF in EBUF as separator
166 E084 CD18E0                    CALL :E018      Update EBUF pointer
167 E087 C9                        RET
168
169                         * Table:
170
171 E088 02                 L3E395 DATA :02
172 E089 54                        DATA :54        T
173 E08A 4F                        DATA :4F        O
174 E08B 6FE0                      DBL  :E06F      Addr encode 'TO'
175                         *
176 E08D 00                        DATA :00        'TO' not found:
177 E08E 0BDA                      DBL  :DA0B      Run 'SYNTAX ERROR'
178                         *
179 E090 04                 L3E396 DATA :04
180 E091 53                        DATA :53        S
181 E092 54                        DATA :54        T
182 E093 45                        DATA :45        E
183 E094 50                        DATA :50        P
184 E095 7BE0                      DBL  :E07B      Addr encode 'STEP'
185                         *
186 E097 00                        DATA :00        'STEP' not found:
187 E098 82E0                      DBL  :E082      End command
```

```
                        *
                        ************************************************
                        * GET ADDRESS ENCODING INSTRUCTION AND GO TO IT *
                        ************************************************
                        *
                        * Entry: DE : Points to table of format:
                        *             < length name / name / jumpaddr > or
                        *             < 00 / jumpaddr >.
                        *        C  : Points to input.
                        * Exit:  C  : Updated.
                        *        AFBHL preserved. D=0, E=1.
                        *
E09A E5         L3E7    PUSH  H
E09B F5                 PUSH  PSW
E09C EB                 XCHG            Addr table in HL
E09D 1E01               MVI   E,:01
E09F CD34CA             CALL  :CA34     Find instruction in table.
                                        On exit, HL points to addr
                                        of encoding routine
E0A2 7E                 MOV   A,M       )
E0A3 23                 INX   H         ) Get addr encoding instr
E0A4 66                 MOV   H,M       )  in HL
E0A5 6F                 MOV   L,A       )
E0A6 F1                 POP   PSW
E0A7 E3                 XTHL            Addr encoding instr on stack
E0A8 C9                 RET             Go to it
                        *
                        ****************************************
                        * ENCODE 'NEXT' AND 'NEXT <VARIABLE>' *
                        ****************************************
                        *
E0A9 CD59E8     ENEXT   CALL  :E859     Next char ':' or 'CR' ?
E0AC C8                 RZ              Then ready

                        * If NEXT <variable>:

E0AD 2B                 DCX   H
E0AE 34                 INR   M         Token +1 (#AC)
E0AF 23                 INX   H
E0B0 CDBCE5             CALL  :E5BC     Encode variable or array ref
E0B3 3A3601             LDA   :0136     Get type latest expression
E0B6 FE20               CPI   :20       String type ?
E0B8 CA1ADA             JZ    :DA1A     Then run error 'TYPE
                                        MISMATCH'
E0BB C9                 RET
                        *
                        ****************************************
                        * ENCODE 'IF - THEN' AND 'IF - GOTO' *
                        ****************************************
                        *
E0BC E5         EIF     PUSH  H
E0BD D5                 PUSH  D
E0BE CD9CE3             CALL  :E39C     Encode boolean expr (type
                                        #30)
E0C1 11EDE0             LXI   D,:E0ED   Addr 'THEN' table
E0C4 C39AE0             JMP   :E09A     Get addr encoding instr
                                        and go to it

                        * Encode 'THEN':

E0C7 CD31E7     L3E10   CALL  :E731     Read a linenr into EBUF
E0CA D1                 POP   D
```

```
250 E0CB D2D4E0              JNC   :E0D4     Jump if no linenr given
251
252                  * If linenr:
253
254 E0CE E3                   XTHL
255 E0CF 2B                   DCX   H
256 E0D0 34                   INR   M
257 E0D1 34                   INR   M        Token +2 (#A8)
258 E0D2 E1                   POP   H
259 E0D3 C9                   RET
260
261                  * If statement:
262
263 E0D4 F1           L3E11   POP   PSW
264 E0D5 E5                   PUSH  H        Preserve EBUF pntr
265 E0D6 CD18E0               CALL  :E018    Update EBUF pointer
266 E0D9 CD24E0               CALL  :E024    Encode BASIC command
267 E0DC 7D                   MOV   A,L      Lobyte new EBUF pntr in A
268 E0DD E3                   XTHL           Old EBUF pntr in HL
269 E0DE 95                   SUB   L
270 E0DF 3D                   DCR   A
271 E0E0 77                   MOV   M,A      Length string in EBUF entry
272 E0E1 E1                   POP   H
273 E0E2 0D                   DCR   C
274 E0E3 C9                   RET
275
276                  * Encode 'GOTO':
277
278 E0E4 F1           L3E12   POP   PSW
279 E0E5 E3                   XTHL
280 E0E6 2B                   DCX   H
281 E0E7 34                   INR   M        Token +1 (#A7)
282 E0E8 E1                   POP   H
283 E0E9 CD2AE7               CALL  :E72A    Get linenr
284 E0EC C9                   RET
285
286                  * Table:
287
288 E0ED 04          L3E397  DATA  :04
289 E0EE 54                  DATA  :54       T
290 E0EF 48                  DATA  :48       H
291 E0F0 45                  DATA  :45       E
292 E0F1 4E                  DATA  :4E       N
293 E0F2 C7E0                DBL   :E0C7     Addr encode 'THEN'
294                  *
295 E0F4 04                  DATA  :04
296 E0F5 47                  DATA  :47       G
297 E0F6 4F                  DATA  :4F       O
298 E0F7 54                  DATA  :54       T
299 E0F8 4F                  DATA  :4F       O
300 E0F9 E4E0                DBL   :E0E4     Addr encode 'GOTO'
301                  *
302 E0FB 00                  DATA  :00       If no 'THEN' or 'GOTO' found
303 E0FC 0BDA                DBL   :DA0B     Run 'SYNTAX ERROR'
304                  *
305                  ****************
306                  * ENCODE 'LET' *
307                  ****************
308                  *
309                  * Encodes a variable or array with arguments. Then
310                  * finds '=' in input (error if not). Then encodes
311                  * an expression, depending on variable type (error
```

```
                        * if type non-existing).
                        *
                        * Exit: DE: Offset of left-side variable.
                        *       A and B: Left-side type.
                        *       C, HL updated. F corrupted.
                        *
E0FE CDBCE5     ELET    CALL  :E5BC     Encode var or array ref
E101 CD67E8             CALL  :E867     Check next char is '='
E104 3D                 DATA  :3D
E105 3A3601             LDA   :0136     Get type latest expression
E108 FE00               CPI   :00       FPT ?
E10A CA76E3             JZ    :E376     Then encode FPT expr
E10D FE10               CPI   :10       INT ?
E10F CA6DE3             JZ    :E36D     Then encode INT expr
E112 C3A1E3     .       JMP   :E3A1     Else encode STR expr
                        *
                        ****************************************
                        * ENCODE 'INPUT' AND 'INPUT <STRING>' *
                        ****************************************
                        *
E115 CDD2DD     EINPUT  CALL  :DDD2     Get char from line, neglect
                                        tab + space
E118 FE22               CPI   :22       " ?
E11A C227E1             JNZ   :E127     Encode 'READ' if no string
                                        in INPUT statement

                        * If 'INPUT <string>':

E11D 2B                 DCX   H
E11E 34                 INR   M         Token +1 (#A1)
E11F 23                 INX   H
E120 CDA1E3             CALL  :E3A1     Encode STR expr
E123 CD67E8             CALL  :E867     Check if next char is ';'
E126 3B                 DATA  :3B
                                        Into EREAD
                        *
                        ******************
                        * ENCODE 'READ' *
                        ******************
                        *
                        * From L3E16 used by several routines, with another
                        * address in LXI D, ...., pointing to various kinds
                        * of encoding/get routines.
                        *
E127 11BCE5     EREAD   LXI   D,:E5BC   Addr routine encode variable
                                        or array reference
                        *
E12A E5         L3E16   PUSH  H
E12B 3600               MVI   M,:00     00 into EBUF (count)
E12D CD18E0             CALL  :E018     Update EBUF pointer
E130 D5         L3E17   PUSH  D
E131 CD64E1             CALL  :E164     Go to encoding routine

                        * Return here after encoding:

E134 D1                 POP   D
E135 E3                 XTHL
E136 34                 INR   M         Count in EBUF +1
E137 E3                 XTHL
E138 CDD2DD             CALL  :DDD2     Get char from line, neglect
                                        tab + space
E13B 0C                 INR   C         Points to next char on line
```

```
374 E13C FE2C               CPI   :2C       ',' ?
375 E13E CA30E1             JZ    :E130     Again if more items
376 E141 0D                 DCR   C         Correct line pointer
377 E142 33                 INX   SP
378 E143 33                 INX   SP        SP to returnaddr
379 E144 C9                 RET
380                 *
381                 ****************************************
382                 * ENCODE A NUMBER OR STRING CONSTANT *
383                 ****************************************
384                 *
385                 * Entry: A: Type.
386                 *
387 E145 F5         L3E18   PUSH  PSW       Preserve type
388 E146 B7                 ORA   A
389 E147 CA5EE1             JZ    :E15E     Jump if FPT type
390 E14A FE10               CPI   :10
391 E14C CA88E8             JZ    :E888     Jump if INT type
392
393                 * If STR type:
394
395 E14F CDD2DD             CALL  :DDD2     Get char from line, neglect
396                                         tab + space
397 E152 FE22               CPI   :22       '"' (quoted string) ?
398 E154 F5                 PUSH  PSW
399 E155 CC80E8             CZ    :E880     Then store quoted string
400                                         in EBUF
401 E158 F1                 POP   PSW
402 E159 C4A6E6             CNZ   :E6A6     Else store unquoted string
403                                         in EBUF
404 E15C F1         L3E19   POP   PSW
405 E15D C9                 RET
406
407                 * If FPT type:
408
409 E15E CD01E5      L3E20   CALL  :E501     Encode FPT nr into EBUF
410 E161 C393E8             JMP   :E893     Quit with evt 'SYNTAX ERROR'
411                 *
412                 ***************************
413                 * part of EREAD (3E12A) *
414                 ***************************
415                 *
416 E164 D5         L3E21   PUSH  D         Addr encoding routine on
417                                         stack
418 E165 C9                 RET             Go to it
419                 *
420                 ****************
421                 * ENCODE 'DIM' *
422                 ****************
423                 *
424 E166 116CE1     EDIM    LXI   D,:E16C   Addr routine 'encoding array
425                                         reference'
426 E169 C32AE1             JMP   :E12A     Continu encoding
427                 *
428                 ******************************
429                 * ENCODE AN ARRAY REFERENCE *
430                 ******************************
431                 *
432 E16C CDBCE5     L3E23   CALL  :E5BC     Encode var or array ref
433 E16F 78                 MOV   A,B       Type in A
434 E170 E640               ANI   :40       Array type ?
435 E172 CA29DA             JZ    :DA29     Run 'SUBSCRIPT ERROR' if not
```

```
436 E175 C9                 RET
437                 *
438                 ****************************************
439                 * ENCODE 'ON - GOTO' AND 'ON - GOSUB' *
440                 ****************************************
441                 *
442 E176 E5         EON     PUSH  H
443 E177 CD6DE3             CALL  :E36D     Encode INT expr
444 E17A 118DE1             LXI   D,:E18D   Addr table
445 E17D C39AE0             JMP   :E09A     Get addr encoding instr
446                                          and go to it
447
448                 * Encode linenr in GOSUB (L3E25) and GOTO (L3E411):
449
450 E180 E3         L3E25   XTHL
451 E181 2B                 DCX   H
452 E182 34                 INR   M         Token +1 (#AF)
453 E183 23                 INX   H
454 E184 E3                 XTHL
455 E185 E3         L3E411  XTHL
456 E186 E1                 POP   H
457 E187 112AE7             LXI   D,:E72A   Addr routine 'get linenr'
458 E18A C32AE1             JMP   :E12A     Continu encoding
459
460                 * Table:
461
462 E18D 04         L3E399  DATA  :04
463 E18E 47                 DATA  :47       G
464 E18F 4F                 DATA  :4F       O
465 E190 54                 DATA  :54       T
466 E191 4F                 DATA  :4F       O
467 E192 85E1               DBL   :E185     Addr encode 'GOTO'
468                 *
469 E194 05                 DATA  :05
470 E195 47                 DATA  :47       G
471 E196 4F                 DATA  :4F       O
472 E197 53                 DATA  :53       S
473 E198 55                 DATA  :55       U
474 E199 42                 DATA  :42       B
475 E19A 80E1               DBL   :E180     Addr encode 'GOSUB'
476                 *
477 E19C 00                 DATA  :00       If no GOTO or GOSUB found:
478 E19D 0BDA               DBL   :DA0B     Run 'SYNTAX ERROR'
479                 *
480                 ******************
481                 * ENCODE 'PRINT' *
482                 ******************
483                 *
484 E19F E5         EPRINT  PUSH  H
485 E1A0 3600               MVI   M,:00     00 into EBUF (init length)
486 E1A2 CD59E8             CALL  :E859     Next char ':' or 'CR' ?
487 E1A5 CACBE1             JZ    :E1CB     Then ready
488
489                 * If statement after PRINT:
490
491 E1A8 CD18E0     L3E27   CALL  :E018     Update EBUF pointer
492 E1AB E3                 XTHL
493 E1AC 34                 INR   M         Length +1
494 E1AD E3                 XTHL
495 E1AE CDB2E3             CALL  :E3B2     Encode non-boolean expr
496                                          preceeded by its type
497 E1B1 36FF               MVI   M,:FF     FF into EBUF
```

```
498 E1B3 CD59E8             CALL  :E859     Next char ':' or 'CR' ?
499 E1B6 CACBE1             JZ    :E1CB     Then ready
500
501                 * If more statements:
502
503 E1B9 77                 MOV   M,A       Char into EBUF
504 E1BA FE2C               CPI   :2C       ',' ?
505 E1BC CAC4E1             JZ    :E1C4     Then continu
506 E1BF FE3B               CPI   :3B       ';' ?
507 E1C1 C20BDA             JNZ   :DA0B     Run 'SYNTAX ERROR' if not
508 E1C4 0C         L3E28   INR   C         Update line pntr
509 E1C5 CD59E8             CALL  :E859     Next char ':' or 'CR' ?
510 E1C8 C2A8E1             JNZ   :E1A8     Continu if not
511                 *
512 E1CB E3         L3E29   XTHL
513 E1CC E1                 POP   H
514 E1CD CD18E0             CALL  :E018     Update EBUF pointer
515 E1D0 C9                 RET
516                 *
517                 ******************
518                 * ENCODE 'MODE' *
519                 ******************
520                 *
521 E1D1 16FF       EMODE   MVI   D,:FF     Default mode 0
522 E1D3 CDD2DD             CALL  :DDD2     Get char from line, neglect
523                                         tab + space
524 E1D6 0C                 INR   C         Points to next char
525 E1D7 D630               SUI   :30
526 E1D9 CAF1E1             JZ    :E1F1     If char is '0': FF in EBUF
527 E1DC DA0BDA             JC    :DA0B     Run 'SYNTAX ERROR' if not
528                                         number or printable char
529 E1DF FE07               CPI   :07       Between 0 and 7?
530 E1E1 D215DA             JNC   :DA15     Run error 'NUMBER OUT OF
531                                         RANGE'if not
532 E1E4 3D                 DCR   A         )
533 E1E5 87                 ADD   A         ) Calc code for mode
534 E1E6 57                 MOV   D,A       in D
535 E1E7 CDE0DD             CALL  :DDE0     Get next char from line
536 E1EA FE41               CPI   :41       'A' ?
537 E1EC C2F1E1             JNZ   :E1F1     Jump if not
538 E1EF 0C                 INR   C         Points to next char
539 E1F0 14                 INR   D         Set D for A-mode
540 E1F1 72         L3E31   MOV   M,D       Mode code in EBUF
541 E1F2 CD18E0             CALL  :E018     Update EBUF pointer
542 E1F5 C9                 RET
543                 *
544                 *
545                 *
546 E1F6                    END
```

```
****************************
* S Y M B O L   T A B L E *
****************************

DBOOT  E00C   EDIM   E166   EFOR   E05F   EIF    E0BC
EINPUT E115   ELET   E0FE   ELINE  E000   ELN    E003
EMODE  E1D1   ENEXT  E0A9   EON    E176   EPRINT E19F
EREAD  E127   ETCON  E006   INXCH  E018   L3E10  E0C7
L3E11  E0D4   L3E12  E0E4   L3E16  E12A   L3E17  E130
L3E18  E145   L3E19  E15C   L3E2   E024   L3E20  E15E
L3E21  E164   L3E23  E16C   L3E25  E180   L3E27  E1A8
L3E28  E1C4   L3E29  E1CB   L3E3   E04F   L3E31  E1F1
```

```
L3E363 E082   L3E394 E015   L3E395 E088   L3E396 E090
L3E397 E0ED   L3E399 E18D   L3E411 E185   L3E5   E06F
L3E6   E07B   L3E7   E09A   MHREQ  E00F   MINKEY E012
USTART E009
```

```
002                             ORG   :E1F6
003                     *
004                     *
005                     *
006                     **********************
007                     * ENCODE 'ENVELOPE' *
008                     **********************
009                     *
010                     * Encodes <ENV> (<V>,<T>;) <V>,<T> or
011                     * <ENV> (<V>,<T>;) <V>.
012                     *
013                     * Exit: HL points beyond expression in EBUF.
014                     *       AFBCDE corrupted.
015                     *
016 E1F6 CD6DE3         EENV    CALL  :E36D     Encode ENV nr
017 E1F9 E5                     PUSH  H         Preserve EBUF pntr
018 E1FA CD18E0                 CALL  :E018     Update EBUF pointer
019 E1FD 1600                   MVI   D,:00     Init length
020 E1FF CD22E2         L3E33   CALL  :E222     Encode <V>
021 E202 CD59E8                 CALL  :E859     Next char ':' or 'CR' ?
022 E205 CA1EE2                 JZ    :E21E     Then ready
023 E208 CD62E8                 CALL  :E862     Check if next char is ','
024                                               run error if not
025 E20B CD22E2                 CALL  :E222     Encode <T>
026 E20E CD67E8                 CALL  :E867     Check if next char is ';'
027 E211 3B                     DATA  :3B
028 E212 14                     INR   D         Length +1
029 E213 36FF                   MVI   M,:FF     FF into EBUF
030 E215 CD59E8                 CALL  :E859     Next char ':' or 'CR' ?
031 E218 C2FFE1                 JNZ   :E1FF     Again if not
032 E21B CD18E0                 CALL  :E018     Update EBUF pointer
033 E21E E3             L3E34   XTHL
034 E21F 72                     MOV   M,D       Length in EBUF after token
035 E220 E1                     POP   H
036 E221 C9                     RET
037                     *
038                     * ENCODE A <V> OR <T> ELEMENT:
039                     *
040                     * Exit: DE preserved.
041                     *
042 E222 D5             L3E35   PUSH  D
043 E223 CD6DE3                 CALL  :E36D     Encode INT expr
044 E226 D1                     POP   D
045 E227 C9                     RET
046                     *
047                     ****************************
048                     * ENCODE 'LIST' AND 'EDIT' *
049                     ****************************
050                     *
051                     * Checks the expression after the token and updates
052                     * the token on it.
053                     * On exit, the token is:
054                     *                       EDIT     LIST
055                     * Without linenr:        B6       93
056                     * One line:              B7       94
057                     * Part of program:       B8       95
058                     *
059                     * Entry: HL    : Points after token in EBUF.
060                     * Exit:  C, HL: Updated.
061                     *        D    : Token.
062                     *        AF corrupted, BE preserved.
063                     *
```

```
064                     ELIST
065 E228 2B             EEDIT   DCX   H          Pnts to token
066 E229 56                     MOV   D,M        Token in D
067 E22A E5                     PUSH  H          Preserve EBUF pntr
068 E22B 23                     INX   H
069 E22C CD59E8                 CALL  :E859      Next char ':' or 'CR' ?
070 E22F CA44E2                 JZ    :E244      Then ready
071 E232 CD48E2                 CALL  :E248      Read linenr into EBUF
072 E235 14                     INR   D          Token +1
073 E236 CD59E8                 CALL  :E859      Next char ':' or 'CR' ?
074 E239 CA44E2                 JZ    :E244      Then ready
075 E23C CD67E8                 CALL  :E867      Next char '-' ?
076 E23F 2D                     DATA  :2D
077 E240 CD48E2                 CALL  :E248      Read linenr into EBUF
078 E243 14                     INR   D          Again token +1
079 E244 E3             L3E37   XTHL
080 E245 72                     MOV   M,D        Token into EBUF
081 E246 E1                     POP   H
082 E247 C9                     RET
083                     *
084                     * READ LINENUMBER INTO EBUF:
085                     *
086                     * Reads a linenumber from the input line into
087                     * the EBUF. If no linenumber is given, 0000 is
088                     * inserted.
089                     *
090                     * Entry: HL: Points to 1st free location in EBUF.
091                     *         C : Points to input line.
092                     * Exit:   C, HL: Updated.
093                     *         AF corrupted, BDE preserved.
094                     *
095 E248 D5             L3E38   PUSH  D
096 E249 CD31E7                 CALL  :E731      Read linenr into EBUF
097 E24C D1                     POP   D
098 E24D D8                     RC               Ready if nr given
099 E24E 3600                   MVI   M,:00      00 into EBUF
100 E250 CD18E0                 CALL  :E018      Update EBUF pointer
101 E253 3600                   MVI   M,:00      00 into EBUF
102 E255 CD18E0                 CALL  :E018      Update EBUF pointer
103 E258 C9                     RET
104                     *
105                     ************************************************
106                     * ENCODE 'WAIT', 'WAIT TIME' AND 'WAIT MEM' *
107                     ************************************************
108                     *
109 E259 115FE2         EWAIT   LXI   D,:E25F    Addr table
110 E25C C39AE0                 JMP   :E09A      Get addr encoding instr
111                                              and go to it
112
113                     * Table:
114
115 E25F 04             L3E401  DATA  :04
116 E260 54                     DATA  :54        T
117 E261 49                     DATA  :49        I
118 E262 4D                     DATA  :4D        M
119 E263 45                     DATA  :45        E
120 E264 85E2                   DBL   :E285      Addr encode 'TIME'
121                     *
122 E266 03                     DATA  :03
123 E267 4D                     DATA  :4D        M
124 E268 45                     DATA  :45        E
125 E269 4D                     DATA  :4D        M
```

```
126 E26A 6FE2                   DBL   :E26F     Addr encode 'MEM'
127                     *
128 E26C 00                     DATA  :00
129 E26D 72E2                   DBL   :E272     Addr encode 'port'
130
131                     * Encode 'MEM':
132
133 E26F 2B             L3E40   DCX   H
134 E270 34                     INR   M         Token +1 (#91)
135 E271 23                     INX   H
136 E272 CD02E3         L3E412  CALL  :E302     Encode <INT expr>,<INT expr>
137 E275 36FF                   MVI   M,:FF     FF into EBUF
138 E277 CD18E0                 CALL  :E018     Update EBUF pointer
139 E27A CDD2DD                 CALL  :DDD2     Get char from line, neglect
140                                             tab + space
141 E27D FE2C                   CPI   :2C       ',' ?
142 E27F C0                     RNZ             Ready if not
143 E280 2B                     DCX   H
144 E281 03                     INX   B
145 E282 C36DE3                 JMP   :E36D     Encode INT expr
146
147                     * Encode 'TIME':
148
149 E285 2B             L3E41   DCX   H
150 E286 34                     INR   M
151 E287 34                     INR   M         Token +2 (#92)
152 E288 23                     INX   H
153 E289 C36DE3                 JMP   :E36D     Encode INT expr
154                     *
155                     ***********************************
156                     * ENCODE 'DRAW', 'FILL', 'DOT' *
157                     ***********************************
158                     *
159                     EFILL
160 E28C CD02E3         EDRAW   CALL  :E302     Encode <INT expr>,<INT expr>
161
162                     * Entry for encode 'DOT':
163
164 E28F CD02E3         EDOT    CALL  :E302     Idem
165 E292 C314E3                 JMP   :E314     Encode INT expr
166                     *
167                     ***************************************
168                     * ENCODE 'RUN' AND 'RUN <LINENR>' *
169                     ***************************************
170                     *
171 E295 CD59E8         ERUN    CALL  :E859     Next char ':' or 'CR' ?
172 E298 C8                     RZ              Then ready
173
174                     * If 'RUN <linenr>':
175
176 E299 2B                     DCX   H
177 E29A 34                     INR   M         Token +1 (#88)
178 E29B 23                     INX   H
179 E29C C32AE7                 JMP   :E72A     Get linenr into EBUF
180                     *
181                     ****************
182                     * ENCODE 'IMP' *
183                     ****************
184                     *
185                     * Note: This command is not encoded, but has
186                     * immediate effect.
```

```
188 E29F E5          EIMP    PUSH  H
189 E2A0 21F1E2              LXI   H,:E2F1   Startaddr table var.types
190 E2A3 1E00                MVI   E,:00     1 info byte
191 E2A5 CD34CA              CALL  :CA34     Find type in table
192 E2A8 7E                  MOV   A,M       Get IMP type from table
193 E2A9 B7                  ORA   A
194 E2AA FA0BDA              JM    :DA0B     'SYNTAX ERROR' if not found
195 E2AD F5                  PUSH  PSW       Preserve IMP type
196 E2AE FE20                CPI   :20       STR type ?
197 E2B0 CAB9E2              JZ    :E2B9     Then jump
198 E2B3 CD59E8              CALL  :E859     IMP INT/FPT alone ?
199 E2B6 CAD9E2              JZ    :E2D9     Then ready
200 E2B9 CDE6E2      EIM10   CALL  :E2E6     Get char from line; check if
201                                          upper case; INR C
202 E2BC F5                  PUSH  PSW       Save char in IMP instruction
203 E2BD CD67E8              CALL  :E867     Next char is '-' ?
204 E2C0 2D                  DATA  :2D
205 E2C1 CDE6E2              CALL  :E2E6     Get next char from line;
206                                          check if upper case; INR C
207 E2C4 213402              LXI   H,:0234   Base addr IMP table
208 E2C7 54                  MOV   D,H       ) into DE
209 E2C8 5D                  MOV   E,L       )
210 E2C9 CD30DE              CALL  :DE30     Calc offset end addr in HL
211 E2CC 23                  INX   H         +1
212 E2CD EB                  XCHG            In DE
213 E2CE F1                  POP   PSW       Get char
214 E2CF CD30DE              CALL  :DE30     Calc offset begin addr
215 E2D2 EB                  XCHG
216 E2D3 F1                  POP   PSW       Get IMP type
217 E2D4 CD7CDE      EIM20   CALL  :DE7C     Load given range with IMP
218                                          type
219 E2D7 E1                  POP   H         Re-instate 'encoded' pntr
220 E2D8 C9                  RET
221
222                  * If no range given:
223
224 E2D9 F1          L3E47   POP   PSW       Get IMP type
225 E2DA 328F02              STA   :028F     Store default number type
226 E2DD 117502              LXI   D,:0275   Startaddr impl. type table
227 E2E0 218F02              LXI   H,:028F   Addr after end table
228 E2E3 C3D4E2              JMP   :E2D4     Fill A-Z with reqd type
229
230                  * Get character from line and check if it is a
231                  * upper case character:
232
233 E2E6 CDD2DD      EALPHA  CALL  :DDD2     Get char from line, neglect
234                                          tab + space
235 E2E9 CD02DE              CALL  :DE02     Check if upper case char
236 E2EC D20BDA              JNC   :DA0B     Run 'SYNTAX ERROR' if not
237 E2EF 0C                  INR   C         Update textline pntr
238 E2F0 C9                  RET
239                  *
240                  * STRINGS VARIABLE TYPES TABLE:
241                  *
242                  * The first byte is a length byte. The byte
243                  * after the string is the variable type byte.
244                  *
245 E2F1 03          IMPTT   DATA  :03
246 E2F2 46                  DATA  :46       F
247 E2F3 50                  DATA  :50       P
248 E2F4 54                  DATA  :54       T
249 E2F5 00                  DATA  :00       type is #00
```

```
                        *
E2F6 03                         DATA  :03
E2F7 49                         DATA  :49       I
E2F8 4E                         DATA  :4E       N
E2F9 54                         DATA  :54       T
E2FA 10                         DATA  :10       type is #10
                        *
E2FB 03                         DATA  :03
E2FC 53                         DATA  :53       S
E2FD 54                         DATA  :54       T
E2FE 52                         DATA  :52       R
E2FF 20                         DATA  :20       type is #20
                        *
E300 00                         DATA  :00       End of table
E301 80                         DATA  :80
                        *
                        ************************************
                        * ENCODE 'POKE', 'OUT', 'CURSOR' *
                        ************************************
                        *
                        * Also used by: Encode 'DRAW' and 'FILL'.
                        *
                        EPOKE
                        EOUT
                        ECURS
E302 CD6DE3             ENC3    CALL  :E36D     Encode INT expr
E305 CD62E8                     CALL  :E862     Check if next char is ',';
                                                run error if not
E308 C36DE3                     JMP   :E36D     Encode INT expression
                        *
                        ******************************
                        * ENCODE 'COLORG', 'COLORT' *
                        ******************************
                        *
                        * Partly also used to encode 'CLEAR' and 'TALK'.
                        *
                        * Exit: C, HL: updated.
                        *       AFDE preserved, B corrupted.
                        *
                        ECOLT
E30B CD6DE3             ECOLG   CALL  :E36D     Encode INT expr
E30E CD6DE3                     CALL  :E36D     Idem
E311 CD6DE3             ENC6    CALL  :E36D     Idem

                        * Entry for encode CLEAR/TALK:

                        ETALK
E314 C36DE3             ECLEAR  JMP   :E36D     Idem
                        *
                        ****************************************
                        * ENCODE 'SOUND' WITH POSSIBLE 'OFF' *
                        ****************************************
                        *
                        * Encodes SOUND <CHAN><ENV><VOL><TG><FREQ>, or
                        * SOUND <CHAN> OFF or SOUND OFF.
                        *
                        * Exit: C, HL: Updated.
                        *       A preserved, B corrupted, D=0, E=1.
                        *       CY=1: Sound off.
                        *
E317 CD2CE3             ESOUND  CALL  :E32C     Encode a possible 'OFF'
E31A D8                         RC              Ready if 'OFF' given
```

```
312 E31B CD6DE3             CALL  :E36D     Encode <CHAN>
313 E31E CD25E3             CALL  :E325     Encode 'OFF' or <ENV><VOL>
314 E321 D8                 RC              Ready if 'OFF' given
315 E322 C311E3             JMP   :E311     Encode <TG><FREQ>
316                 *
317                 ****************************************
318                 * ENCODE 'NOISE' WITH POSSIBLE 'OFF' *
319                 ****************************************
320                 *
321                 * Encodes NOISE <ENV><VOL> or NOISE OFF.
322                 *
323                 * Exit: C, HL: Updated.
324                 *       CY=1 : Noise off.
325                 *       A preserved, B corrupted, D=0, E=1.
326                 *
327 E325 CD2CE3     ENOISE  CALL  :E32C     Encode  possible 'OFF'
328 E328 D411E3             CNC   :E311     Encode <ENV><VOL> if no
329                                         'OFF' given
330 E32B C9                 RET
331                 *
332                 ****************************
333                 * ENCODE A POSSIBLE 'OFF' *
334                 ****************************
335                 *
336                 * Exit: CY=1:  'OFF' in input; FF in EBUF.
337                 *       CY=0:  No 'OFF' given.
338                 *       C, HL: Updated.
339                 *       AB preserved, D=0, E=1.
340                 *
341 E32C 1132E3     L3E55   LXI   D,:E332   Startaddr table
342 E32F C39AE0             JMP   :E09A     Get addr encoding instr
343                                         and go to it
344
345                 * Table:
346
347 E332 03         L3E404  DATA  :03
348 E333 4F                 DATA  :4F       O
349 E334 46                 DATA  :46       F
350 E335 46                 DATA  :46       F
351 E336 3BE3               DBL   :E33B     addr encode 'OFF'
352                 *
353 E338 00                 DATA  :00
354 E339 42E3               DBL   :E342     encoding addr if no 'OFF'
355
356                 * Encode 'OFF':
357
358 E33B 36FF       L3E365  MVI   M,:FF     FF into EBUF
359 E33D CD18E0             CALL  :E018     Update EBUF pointer
360 E340 37                 STC             CY=1
361 E341 C9                 RET
362
363                 * If no 'OFF':
364
365 E342 B7         L3E366  ORA   A         CY=0
366 E343 C9                 RET
367                 *
368                 ******************
369                 * ENCODE 'CALLM' *
370                 ******************
371                 *
372 E344 CD6DE3     ECALM   CALL  :E36D     Encode memory addr (INT)
373 E347 36FF               MVI   M,:FF     FF into EBUF
```

```
374 E349 23                 INX   H
375 E34A CD59E8             CALL  :E859     Next char ':' or 'CR' ?
376 E34D C8                 RZ              Then ready
377 E34E 2B                 DCX   H
378 E34F CD62E8             CALL  :E862     Check if next char is ',';
379                                         run error if not
380 E352 C3BCE5             JMP   :E5BC     Encode var.pntr
381                 *
382                 ***************************
383                 * ENCODE 'SAVE', 'LOAD' *
384                 ***************************
385                 *
386                 * Checks if a name is given after 'SAVE/LOAD',
387                 * then encodes a string. Else #19,#00 (empty
388                 * unquoted string) is added to code.
389                 *
390                 * Exit: C,HL updated, AFB corrupted, DE preserved.
391                 *
392                 ELOAD
393 E355 CD59E8     ESAVE   CALL  :E859     Next char ':' or 'CR' ?
394 E358 C2A1E3             JNZ   :E3A1     Encode string if not
395 E35B 3619               MVI   M,:19     #19 in next loc EBUF
396 E35D CD18E0             CALL  :E018     Update EBUF pointer
397 E360 3600               MVI   M,:00     #00 in next loc EBUF
398 E362 CD18E0             CALL  :E018     Update EBUF pointer
399 E365 C9                 RET
400                 *
401                 ************************************
402                 * ENCODE 'REM' AND '***' (ERROR) *
403                 ************************************
404                 *
405                 EERR
406 E366 C3B0E6     EREM    JMP   :E6B0     Encode text
407                 *
408                 ***************************
409                 * ENCODE SINGLE ROUTINES *
410                 ***************************
411                 *
412                 EREST
413                 EEND
414                 ENEW
415                 ERET
416                 ECHECK
417                 ECONT
418                 ESTEP
419                 ETROF
420                 ESTOP
421                 ETRON
422 E369 C9         EUT     RET             No further handling
423                 *
424                 ***************************
425                 * ENCODE 'GOTO', 'GOSUB' *
426                 ***************************
427                 *
428                 EGOTO
429 E36A C32AE7     EGOSUB  JMP   :E72A     Get linenr
430                 *
431                 ************************************************
432                 * ENCODE AN EXPRESSION IF VARIABLE TYPE IS INT *
433                 ************************************************
434                 *
435 E36D D5         L3E371  PUSH  D
```

```
436 E36E 110001          LXI   D,:0100   Set D=#01 (conversion) and
437                                      E=#00 (FPT var type)
438 E371 0610            MVI   B,:10     Reqd type is INT
439 E373 C37CE3          JMP   :E37C     Encode expression
440                *
441                ************************************************
442                * ENCODE AN EXPRESSION IF VARIABLE TYPE IS FPT *
443                ************************************************
444                *
445 E376 D5        L3E372  PUSH  D
446 E377 111002          LXI   D,:0210   Set D for evt conversion
447                                      and E=#10 (INT var type)
448 E37A 0600            MVI   B,:00     Reqd type is FPT
449 E37C F5        L3E373  PUSH  PSW
450 E37D 78              MOV   A,B
451 E37E 329002          STA   :0290     Set reqd number type
452 E381 E5              PUSH  H
453 E382 D5              PUSH  D
454 E383 CDC9E3          CALL  :E3C9     Encode expr
455 E386 D1              POP   D
456 E387 3A3601          LDA   :0136     Get type latest expression
457 E38A B8              CMP   B         Was it as reqd ?
458 E38B CA98E3          JZ    :E398     Then ready
459
460                * Type not as expected:
461
462 E38E BB              CMP   E         Was it alternative type
463 E38F C21ADA          JNZ   :DA1A     Run error 'TYPE MISMATCH'
464                                      if not
465 E392 7A              MOV   A,D       Get conversion byte in A
466 E393 D1              POP   D
467 E394 D5              PUSH  D
468 E395 CD70E7          CALL  :E770     Add conversion byte to expr
469 E398 D1        L3E374  POP   D
470 E399 F1        L3E375  POP   PSW
471 E39A D1              POP   D
472 E39B C9              RET
473                *
474                ********************************
475                * ENCODE A BOOLEAN EXPRESSION *
476                ********************************
477                *
478                * Variable type is #30.
479                *
480 E39C 0630      L3E376  MVI   B,:30     Var type is #30
481 E39E C3A3E3          JMP   :E3A3     Encode expression
482                *
483                ********************************
484                * ENCODE A STRING EXPRESSION *
485                ********************************
486                *
487                * Variable type is #20.
488                *
489 E3A1 0620      L3E377  MVI   B,:20     Var type is #20
490
491                * Entry for 'encode Boolean expression':
492
493 E3A3 D5        L3E378  PUSH  D
494 E3A4 F5                PUSH  PSW
495 E3A5 CDC9E3            CALL  :E3C9     Encode expr
496 E3A8 3A3601            LDA   :0136     Get type latest expression
497 E3AB B8                CMP   B         Compare with reqd type
```

```
498 E3AC C21ADA             JNZ   :DA1A     Run error 'TYPE MISMATCH'
499                                          if not identical
500 E3AF C399E3             JMP   :E399     Quit
501                 *
502                 ************************************
503                 * ENCODE A NON-BOOLEAN EXPRESSION *
504                 ************************************
505                 *
506                 * Encodes an entire expression, preceeded by its
507                 * type, into the EBUF. 'TYPE MISMATCH' error occurs
508                 * if expression is boolean.
509                 *
510                 * Exit: C,HL updated; B preserved.
511                 *       A: Type:      Type in TYPE.
512                 *       D: Orig. B    OLDOP,RGTPT,HOPPT preserved.
513                 *       E: OLDOP.
514                 *
515 E3B2 E5         EEXPI   PUSH  H
516 E3B3 CD18E0             CALL  :E018     Update EBUF pointer
517 E3B6 AF                 XRA   A
518 E3B7 329002             STA   :0290     Req. number type is #00
519 E3BA CDC9E3             CALL  :E3C9     Encode expr
520 E3BD 3A3601             LDA   :0136     Get type latest expression
521 E3C0 FE30               CPI   :30       Boolean ?
522 E3C2 CA1ADA             JZ    :DA1A     Then run error 'TYPE
523                                          MISMATCH'
524 E3C5 E3                 XTHL            Get old EBUF pntr
525 E3C6 77                 MOV   M,A       Type into EBUF
526 E3C7 E1                 POP   H         New EBUF pntr
527 E3C8 C9                 RET
528                 *
529                 ************************
530                 * ENCODE AN EXPRESSION *
531                 ************************
532                 *
533                 * Routine encodes an entire expression until no
534                 * operator (or INOT) is found. The expression may
535                 * begin with an unitary operator (highest priority).
536                 *
537                 * Exit: C,HL updated, B preserved.
538                 *       A,E: OLDOP;   D: Entry B.
539                 *       OLDOP, RGTPT, HOPPT preserved.
540                 *       Type of expr in TYPE.
541                 *       RGTOP: #00 or #1E.
542                 *
543 E3C9 EB         L3E380  XCHG            Save HL in DE
544 E3CA 60                 MOV   H,B       Var type in H
545 E3CB 3A3801             LDA   :0138     Get old priority operator
546 E3CE 6F                 MOV   L,A       in L
547 E3CF E5                 PUSH  H         Save it on stack
548 E3D0 2A3901             LHLD  :0139     Get pntr place for operator
549 E3D3 E5                 PUSH  H         Save it on stack
550 E3D4 2A3B01             LHLD  :013B     Get pntr to RGT operand of
551                                          last operator
552 E3D7 E5                 PUSH  H         Save it on stack
553 E3D8 EB                 XCHG
554 E3D9 AF                 XRA   A
555 E3DA 323801             STA   :0138     Reset old prio operator
556 E3DD CDF1E3             CALL  :E3F1     Encode a term with possible
557                                          unitary operator
558 E3E0 EB                 XCHG
559 E3E1 E1                 POP   H
```

```
560 E3E2 223B01              SHLD  :013B     Restore RGTPT
561 E3E5 E1                  POP   H
562 E3E6 223901              SHLD  :0139     Restore HOPPT
563 E3E9 E1                  POP   H
564 E3EA 44                  MOV   B,H       Restore B
565 E3EB 7D                  MOV   A,L
566 E3EC 323801              STA   :0138     Restore OLDOP
567 E3EF EB                  XCHG
568 E3F0 C9                  RET
569                  *
570                  *
571                  *
572 E3F1                     END
```

****
* SYMBOL TABLE *
****

EALPHA	E2E6	ECALM	E344	ECHECK	E369	ECLEAR	E314
ECOLG	E30B	ECOLT	E30B	ECONT	E369	ECURS	E302
EDOT	E28F	EDRAW	E28C	EEDIT	E22B	EEND	E369
EENV	E1F6	EERR	E366	EEXPI	E3B2	EFILL	E28C
EGOSUB	E36A	EGOTO	E36A	EIM10	E2B9	EIM20	E2D4
EIMP	E29F	ELIST	E228	ELOAD	E355	ENC3	E302
ENC6	E311	ENEW	E369	ENOISE	E325	EOUT	E302
EPOKE	E302	EREM	E366	EREST	E369	ERET	E369
ERUN	E295	ESAVE	E355	ESOUND	E317	ESTEP	E369
ESTOP	E369	ETALK	E314	ETROF	E369	ETRON	E369
EUT	E369	EWAIT	E259	IMPTT	E2F1	L3E33	E1FF
L3E34	E21E	L3E35	E222	L3E365	E33B	L3E366	E342
L3E37	E244	L3E371	E36D	L3E372	E376	L3E373	E37C
L3E374	E398	L3E375	E399	L3E376	E39C	L3E377	E3A1
L3E378	E3A3	L3E38	E248	L3E380	E3C9	L3E40	E26F
L3E401	E25F	L3E404	E332	L3E41	E285	L3E412	E272
L3E47	E2D9	L3E55	E32C				

```
                        ORG   :E3F1
                *
                *
                *
                ******************************************************
                * ENCODE A TERM WITH A POSSIBLE UNITARY OPERATOR *
                ******************************************************
                *
                * L3E381: First operand may be preceeded by unitary
                *         operator +, - or INOT.
                *         Then code byte preceeding 1st operand is:
                *                          +    -  INOT
                *                INT:     BC   BD   BE
                *                FPT:     9C   9D    *
                *
                * L3E382: Encodes a sequence of higher priority
                *         operations and their operands.
                *         Encodes all terms after OLDOP in input
                *         which form a succession of higher priority
                *         operators until next lower or equal
                *         operator is found. This will be in RGTOP.
                *         The type of this collection will be in
                *         TYPE.
                *
                * Exit: C,HL updated. BE corrupted. CY=0.
                *       A: bits 5,6,7 OLDOP; D: bits 5,6,7 new RGTOP
                *
E3F1 CDF6E6     L3E381  CALL  :E6F6     Find unitary operator
                                        in table
E3F4 B7                 ORA   A
E3F5 C244E4             JNZ   :E444     Jump if found
                *
E3F8 223901     L3E382  SHLD  :0139     Set pntr place for operator
E3FB CD55E4             CALL  :E455     Encode 1st operand
E3FE CDEAE6     L3E383  CALL  :E6EA     Find binary or unitary
                                        operator in table
E401 323701             STA   :0137     Store latest prio operator
E404 3A3701     L3E384  LDA   :0137     Get latest prio operator
E407 E6E0               ANI   :E0       Prio in bits 5,6,7
E409 57                 MOV   D,A       RGTOP in D
E40A 3A3801             LDA   :0138     Get old prio operator
E40D E6E0               ANI   :E0       Prio only
E40F BA                 CMP   D         Compare both operators
E410 D0                 RNC             Ready if RGTOP <= OLDOP

                * RGTOP > OLDOP:

E411 EB                 XCHG
E412 2A3601             LHLD  :0136     Get type latest expression
E415 E5                 PUSH  H         Preserve type left operand
                                        and RGTOP
E416 3A3801             LDA   :0138     Get old prio operator
E419 F5                 PUSH  PSW       Preserve it
E41A D5                 PUSH  D         Preserve EBUF pntr
E41B 2A3901             LHLD  :0139     Get pntr place for operator
E41E E5                 PUSH  H         Preserve HOPPT
E41F EB                 XCHG            New EBUF pntr in HL
E420 3A3701             LDA   :0137     Get latest prio operator
E423 323801             STA   :0138     and store it as old one
E426 CDF8E3             CALL  :E3F8     Encode right hand operand
                                        until higher prio operators
E429 EB                 XCHG
```

```
064 E42A E1                 POP   H
065 E42B 223901             SHLD  :0139     Restore HOPPT
066 E42E E1                 POP   H
067 E42F 223B01             SHLD  :013B     Restore RGTPT
068 E432 F1                 POP   PSW
069 E433 323801             STA   :0138     Restore OLDOP
070 E436 EB                 XCHG            New EBUF pntr in HL
071 E437 D1                 POP   D         Restore type left operand
072                                         (E) and orig RGTOP (D)
073 E438 7A                 MOV   A,D       Old RGTOP in A
074 E439 E61F               ANI   :1F       Op.code only
075 E43B CDCFE7             CALL  :E7CF     Obtain type info for
076                                         binary operation
077 E43E CD57E7             CALL  :E757     Encode binary operation
078                                         into EBUF
079 E441 C304E4             JMP   :E404     Check again if prios correct
080
081                 * Unitary operator:
082
083 E444 223901      L3E56   SHLD  :0139     Set pntr place for operator
084 E447 E5                 PUSH  H
085 E448 CD55E4             CALL  :E455     Encode a term
086 E44B CD97E7             CALL  :E797     Encode unitary operator
087                                         for a term
088 E44E D1                 POP   D
089 E44F CD83E7             CALL  :E783     Byte in A into EBUF
090 E452 C3FEE3             JMP   :E3FE     Encode sequence with prio's
091                 *
092                 * ENCODE A TERM:
093                 *
094                 * Non-error exit: C,HL: updated.
095                 *                 BCDE corrupted, AF preserved.
096                 *
097 E455 F5          L3E57   PUSH  PSW
098 E456 CDD2DD             CALL  :DDD2     Get char from line, neglect
099                                         tab + space
100 E459 FE22               CPI   :22
101 E45B CA9BE4             JZ    :E49B     Jump if char is '"'
102 E45E FE28               CPI   :28
103 E460 CA86E4             JZ    :E486     Jump if char is '('
104 E463 CD02DE             CALL  :DE02     Check if char is upper case
105 E466 DA77E4             JC    :E477     Then jump
106 E469 FE2D               CPI   :2D
107 E46B CA0BDA             JZ    :DA0B     Run 'SYNTAX ERROR' if '-'
108 E46E CDC8E4             CALL  :E4C8     Encode a number
109 E471 D20BDA             JNC   :DA0B     Evt run 'SYNTAX ERROR'
110 E474 C3A4E4             JMP   .:E4A4    Store type and quit
111
112                 * If upper case character:
113
114 E477 CD22E5      L3E58   CALL  :E522     Encode function
115 E47A DAA4E4             JC    :E4A4     If ready: store type (#30)
116                                         and quit
117 E47D 48                 MOV   C,B
118 E47E 1600               MVI   D,:00
119 E480 CDBCE5             CALL  :E5BC     Encode var/array reference
120 E483 C3A7E4             JMP   :E4A7     Quit
121
122                 * If opening bracket:
123
124 E486 369A        L3E59   MVI   M,:9A     Load #9A in EBUF
125 E488 CD18E0             CALL  :E018     Update EBUF pointer
```

```
126 E48B 0C             INR   C         Next pos in textline
127 E48C CDC9E3         CALL  :E3C9     Encode expression
128 E48F CDE0DD         CALL  :DDE0     Get char from line
129 E492 FE29           CPI   :29
130 E494 C20BDA         JNZ   :DA0B     'SYNTAX ERROR' if not ')'
131 E497 0C             INR   C         Next pos in textline
132 E498 C3A7E4         JMP   :E4A7     Quit
133
134                 * If opening '"':
135
136 E49B 00         L3E60   NOP
137 E49C CD80E8             CALL  :E880     Store quoted text in EBUF
138 E49F 3E20               MVI   A,:20     Type is STR
139 E4A1 C3A4E4             JMP   :E4A4     Store type and quit
140
141                 * Ready:
142
143 E4A4 323601     L3E61   STA   :0136     Set type latest expression
144 E4A7 F1         L3E62   POP   PSW
145 E4A8 C9                 RET
146                 *
147                 ****************************
148                 * ENCODE 'SAVEA', 'LOADA' *
149                 ****************************
150                 *
151                 ELODA
152 E4A9 CD78E6     ESAVA   CALL  :E678     Enc. array without arguments
153 E4AC C355E3             JMP   :E355     Into encode 'SAVE/LOAD'
154                 *
155 E4AF FF                 DATA  :FF
156 E4B0 FF                 DATA  :FF
157 E4B1 FF                 DATA  :FF
158 E4B2 FF                 DATA  :FF
159 E4B3 FF                 DATA  :FF
160 E4B4 FF                 DATA  :FF
161 E4B5 FF                 DATA  :FF
162 E4B6 FF                 DATA  :FF
163 E4B7 FF                 DATA  :FF
164 E4B8 FF                 DATA  :FF
165 E4B9 FF                 DATA  :FF
166 E4BA FF                 DATA  :FF
167 E4BB FF                 DATA  :FF
168 E4BC FF                 DATA  :FF
169 E4BD FF                 DATA  :FF
170 E4BE FF                 DATA  :FF
171 E4BF FF                 DATA  :FF
172 E4C0 FF                 DATA  :FF
173 E4C1 FF                 DATA  :FF
174 E4C2 FF                 DATA  :FF
175 E4C3 FF                 DATA  :FF
176 E4C4 FF                 DATA  :FF
177 E4C5 FF                 DATA  :FF
178 E4C6 FF                 DATA  :FF
179 E4C7 FF                 DATA  :FF
180                 *
181                 ********************
182                 * ENCODE A NUMBER *
183                 ********************
184                 *
185                 * Encodes a INT or a FPT number.
186                 *
187                 * Entry: C : Offset of start of number.
```

```
                        *        HL: Points to EBUF.
                        *
                        * On exit: In A:     #00 FPT;  #10 INT;  #10 HEX.
                        *          In EBUF:  #10 FPT;  #14 INT;  #15 HEX.
                        *
                        * On exit, non-hex numbers will only be INT if reqd
                        * type or IMP type is INT, and there is no '.' or
                        * 'E' in the input string.
                        *
                        * Non-error exit: CY=1: C,HL updated, B preserved.
                        *                       A: Type-code, DE: entry BC.
                        * Error exit:     CY=0, BCDEHL preserved.
                        *
E4C8 C5         ENUM    PUSH  B
E4C9 E5                 PUSH  H
E4CA CDD2DD             CALL  :DDD2     Get char from line, neglect
                                        tab + space
E4CD FE23               CPI   :23
E4CF CA13E5             JZ    :E513     Hex if char is '#'
E4D2 3A9002             LDA   :0290     Get required number type
E4D5 B7                 ORA   A
E4D6 C2DCE4             JNZ   :E4DC     Jump if reqd type is not FPT
E4D9 3A8F02             LDA   :028F     Get default number type
E4DC FE00       ENM03   CPI   :00
E4DE CAFFE4             JZ    :E4FF     Jump if type is not FPT

                        * If INT:

E4E1 3614               MVI   M,:14     Load #14 in EBUF
E4E3 CD18E0             CALL  :E018     Update EBUF pointer
E4E6 CD7BE5             CALL  :E57B     INT nr into EBUF
E4E9 D2FFE4             JNC   :E4FF     Then handle as FPT
E4EC CDE0DD             CALL  :DDE0     Get char from line
E4EF FE2E               CPI   :2E
E4F1 CAFFE4             JZ    :E4FF     Try FPT if no digits before
                                        the '.'
E4F4 FE45               CPI   :45
E4F6 CAFFE4             JZ    :E4FF     Handle as FPT if 'E'
E4F9 3E10       ENM05   MVI   A,:10     Type is INT
E4FB D1         ENM10   POP   D
E4FC D1                 POP   D
E4FD 37                 STC             CY=1
E4FE C9                 RET

                        * If FPT:

E4FF E1         ENM20   POP   H
E500 C1                 POP   B

                        * Entry from ETCON:

E501 C5         ENM25   PUSH  B         ) Save pointers
E502 E5                 PUSH  H         )
E503 3610               MVI   M,:10     Load #10 in EBUF
E505 CD18E0             CALL  :E018     Update EBUF pointer
E508 CD96E5             CALL  :E596     FPT nr into EBUF
E50B 3E00               MVI   A,:00     Type is FPT
E50D DAFBE4             JC    :E4FB     Jump if no errors
E510 E1                 POP   H
E511 C1                 POP   B
E512 C9                 RET             Error exit (no number)
```

```
250                     * If HEX:
251
252 E513 3615           ENM30   MVI   M,:15      Load #15 in EBUF
253 E515 CD18E0                 CALL  :E018      Update EBUF pointer
254 E518 00                     NOP
255 E519 CD84E8                 CALL  :E884      Hex nr into EBUF
256 E51C D20BDA                 JNC   :DA0B      Run 'SYNTAX ERROR' if no
257                                              digits
258 E51F C3F9E4                 JMP   :E4F9      Quit
259                     *
260                     **********************
261                     * ENCODE A FUNCTION *
262                     **********************
263                     *
264                     * Reads a system function (both function and
265                     * arguments) into EBUF. Error exit if syntax or
266                     * type mismatch errors are found.
267                     *
268                     * Entry:   HL: Points to 1st free pos. in EBUF.
269                     *          C : Points to input.
270                     * Exit: If found: CY=1:
271                     *          A:  Type info of result.
272                     *          C,HL updated: BDE corrupted.
273                     *       If not found: CY=0:
274                     *          A:  End of table count.
275                     *          B:  Start of name in input.
276                     *          C:  Points beyond.
277                     *          DE: Points to 0 T/L byte at table end.
278                     *          HL: Points to EBUF.
279                     *
280 E522 E5             EFUN    PUSH  H
281 E523 37                     STC                Set 'include type letter'
282 E524 CDFDE6                 CALL  :E6FD        Find variable name in input,
283                                                 allow !,%,$.
284 E527 21E6CF                 LXI   H,:CFE6      Addr table BASIC functions
285 E52A CD5ACA                 CALL  :CA5A        Find function in table
286 E52D 7B                     MOV   A,E          Get serialnr of entry in
287                                                 table in A
288 E52E EB                     XCHG
289 E52F E1                     POP   H
290 E530 D27AE5                 JNC   :E57A        Abort if not found (CY=0)
291
292                     * If found in table:
293
294 E533 3620                   MVI   M,:20        Load fn.code (#20) in EBUF
295 E535 CD18E0                 CALL  :E018        Update EBUF pointer
296 E538 77                     MOV   M,A          Load how manyth function
297                                                 in EBUF
298 E539 CD18E0                 CALL  :E018        Update EBUF pointer
299 E53C 1A                     LDAX  D            Get T/L byte of function
300 E53D 13                     INX   D
301 E53E F5                     PUSH  PSW          Preserve it
302 E53F E60F                   ANI   :0F          Le ngth only
303 E541 CA76E5                 JZ    :E576        Jump if no arguments reqd
304
305                     * If arguments required:
306
307 E544 F5                     PUSH  PSW          Preserve length fuction
308 E545 CD67E8                 CALL  :E867        Check if next char is '(',
309 E548 28                     DATA  :28          run 'SYNTAX ERROR' if not
310 E549 EB             L3E72   XCHG
311 E54A 7E                     MOV   A,M          Get T/L byte
```

```
312 E54B 23                 INX   H
313 E54C EB                 XCHG
314 E54D FE30               CPI   :30       Boolean type ?
315 E54F CCBCE5             CZ    :E5BC     Then encode var/array ref.
316 E552 CA65E5             JZ    :E565     and jump
317 E555 FE20               CPI   :20       STR type ?
318 E557 CCA1E3             CZ    :E3A1     Then encode STR expr
319 E55A CA65E5             JZ    :E565     and jump
320 E55D FE10               CPI   :10       INT type ?
321 E55F CC6DE3             CZ    :E36D     Then encode INT expr
322 E562 C476E3             CNZ   :E376     Else: encode FPT expr
323                 *
324 E565 F1         L3E73   POP   PSW       Get length function
325 E566 3D                 DCR   A         Check if all arguments done
326 E567 CA72E5             JZ    :E572     Jump if ready
327 E56A F5                 PUSH  PSW       Preserve T/L function
328 E56B CD67E8             CALL  :E867     Check if next char is ','
329 E56E 2C                 DATA  :2C       run 'SYNTAX ERROR' if not
330 E56F C349E5             JMP   :E549     Encode next argument
331 E572 CD67E8     L3E74   CALL  :E867     Check if next char is ')',
332 E575 29                 DATA  :29       run 'SYNTAX ERROR' if not
333 E576 F1         L3E75   POP   PSW       Get T/L function
334 E577 E630               ANI   :30       Type only
335 E579 37                 STC             CY=1
336 E57A C9         L3E76   RET
337                 *
338                 ************************
339                 * INT NUMBER INTO EBUF *
340                 ************************
341                 *
342                 * Entry: C : Points to input.
343                 *         HL: Points to EBUF.
344                 *
345 E57B CDD2DD      EINT    CALL  :DDD2     Get char from line, neglect
346                                         tab + space
347 E57E FE2D                 CPI   :2D
348 E580 C285E5               JNZ   :E585     Jump if char is not '-'
349 E583 0C                 INR   C
350 E584 AF                 XRA   A         Else: clear sign bit
351 E585 CD24C0     L3E78   CALL  :C024     Input INT number to MACC
352 E588 C28DE5             JNZ   :E58D     Jump if nr was >= 0
353 E58B E7                 RST   4         Change sign MACC
354 E58C 60                 DATA  :60
355 E58D C3A3E5     L3E79   JMP   :E5A3     Move MACC into EBUF
356                 *
357                 ************************
358                 * HEX NUMBER INTO EBUF *
359                 ************************
360                 *
361 E590 CD2AC0     L3E80   CALL  :C02A     Input HEX number to MACC
362 E593 C3A3E5             JMP   :E5A3     Move MACC into EBUF
363                 *
364                 ************************
365                 * FPT NUMBER INTO EBUF *
366                 ************************
367                 *
368                 * Entry: C : Points to input.
369                 *         HL: Points to EBUF.
370                 * Non-error exit:  CY=1:
371                 *         C,HL updated. B preserved. A corrupted.
372                 *         DE: Location of number in EBUF.
373                 * Error exit: CY=0:
```

```
374                     *       BDEHL preserved, A corrupted, C updated.
375                     *
376 E596 CDD2DD         EFPT    CALL    :DDD2       Get char from line, neglect
377                                                 tab + space
378 E599 FE2D                   CPI     :2D
379 E59B C2A0E5                 JNZ     :E5A0       Jump if char is not '-'
380 E59E 0C                     INR     C
381 E59F AF                     XRA     A           Else: clear sign bit
382 E5A0 CD79E8         L3E82   CALL    :E879       FPT nr into MACC, evt
383                                                 change sign
384
385                     * Entry for HEX/INT numbers:
386
387 E5A3 D2BBE5         L3E83   JNC     :E5BB       Abort if there are no digits
388 E5A6 C5                     PUSH    B
389 E5A7 54                     MOV     D,H         Old EBUF pntr in DE
390 E5A8 5D                     MOV     E,L
391 E5A9 CD18E0                 CALL    :E018       ) Update EBUF pointer
392 E5AC CD18E0                 CALL    :E018       ) to after new input
393 E5AF CD18E0                 CALL    :E018       )
394 E5B2 CD18E0                 CALL    :E018       )
395 E5B5 EB                     XCHG
396 E5B6 E7                     RST     4           Copy MACC into EBUF
397 E5B7 0F                     DATA    :0F
398 E5B8 EB                     XCHG
399 E5B9 C1                     POP     B
400 E5BA 37                     STC                 CY=1
401 E5BB C9             L3E84   RET
402                     *
403                     *
404                     *
405 E5BC                        END
```

```
*****************************
* S Y M B O L   T A B L E *
*****************************
```

EFPT	E596	EFUN	E522	EINT	E57B	ELODA	E4A9
ENM03	E4DC	ENM05	E4F9	ENM10	E4FB	ENM20	E4FF
ENM25	E501	ENM30	E513	ENUM	E4C8	ESAVA	E4A9
L3E381	E3F1	L3E382	E3F8	L3E383	E3FE	L3E384	E404
L3E56	E444	L3E57	E455	L3E58	E477	L3E59	E486
L3E60	E49B	L3E61	E4A4	L3E62	E4A7	L3E72	E549
L3E73	E565	L3E74	E572	L3E75	E576	L3E76	E57A
L3E78	E585	L3E79	E58D	L3E80	E590	L3E82	E5A0
L3E83	E5A3	L3E84	E5BB				