```
002                             ORG   :E016
003                     *
004                     *
005                     *
006                     ***************************
007                     * RUN basiccmd WAIT TIME *
008                     ***************************
009                     *
010                     * Timer 01BE/BF is decremented by clock
011                     * interrupt RST7.
012                     *
013 E016 CDF8E6         RWTET   CALL  :E6F8     Get time to wait
014 E019 22BE01                 SHLD  :01BE     Load timer
015 E01C 2ABE01         RWT40   LHLD  :01BE     Get timer
016 E01F 7C                     MOV   A,H
017 E020 B5                     ORA   L
018 E021 C8                     RZ              Abort if (timer)=0
019 E022 CDA5D6                 CALL  :D6A5     Check keyb for new inputs
020 E025 D21CE0                 JNC   :E01C     Again if no break pressed
021
022                     * If suspended:
023
024 E028 C312E0                 JMP   :E012     Abort, 'command broken in'
025                     *
026                     ****************************************
027                     * RUN basiccmd FOR .. TO .. (STEP ..) *
028                     ****************************************
029                     *
030 E02B D1             RFOR    POP   D         Kill return addr
031 E02C CD3CE1                 CALL  :E13C     Save old FRAME on stack
032 E02F CD5AE4                 CALL  :E45A     Init variable
033 E032 220401                 SHLD  :0104     Remember location variable
034 E035 E630                   ANI   :30
035 E037 D610                   SUI   :10       Set flags for var.type
036 E039 CD08E8                 CALL  :E808     Eval TO expr, result in MACC
037 E03C CAA2E0                 JZ    :E0A2     If INT variable
038
039                     * If FPT variable:
040
041 E03F E7                     RST   4         Subtract 'FROM'
042 E040 03                     DATA  :03
043 E041 210601                 LXI   H,:0106   Get addr. LSTPF
044 E044 3600                   MVI   M,:00     Default STEP implicit
045 E046 0A                     LDAX  B         Get evt. STEP val. from text
046 E047 03                     INX   B
047 E048 FEFF                   CPI   :FF
048 E04A CA5FE0                 JZ    :E05F     Jump if no STEP
049
050                     * If STEP:
051
052 E04D CD18C0                 CALL  :C018     Save 'to-from' on stack
053 E050 34                     INR   M         Stepflag explicit
054 E051 0B                     DCX   B
055 E052 CD08E8                 CALL  :E808     Stepvalue in MACC
056 E055 210701                 LXI   H,:0107   Addr. LSTEP
057 E058 E7                     RST   4         Stepvalue in LSTEP
058 E059 0F                     DATA  :0F
059 E05A CD1BC0                 CALL  :C01B     'to-from' range in MACC
060 E05D E7                     RST   4         Find nr of iterations
061 E05E 09                     DATA  :09
062 E05F E7             RFR10   RST   4         Make it INT
063 E060 48                     DATA  :48
```



```
064 E061 210B01         RFR20   LXI   H,:010B   Addr. LCOUNT
065 E064 E7                     RST   4         Iterations in LCOUNT
066 E065 0F                     DATA  :0F
067 E066 7E                     MOV   A,M
068 E067 B7                     ORA   A
069 E068 FC9ECB                 CM    :CB9E     Clear LCOUNT if loop in
070                                               wrong direction
071 E06B 60                     MOV   H,B       ) Current pos in start of
072 E06C 69                     MOV   L,C       ) loop in HL
073 E06D 220F01                 SHLD  :010F     Set pointer to start loop
074
075                     * Now delete any previous use of the loop:
076
077 E070 011000                 LXI   B,:0010   Size of 1 'FOR' stackframe
078 E073 2A0401                 LHLD  :0104     Get current loop variable
079 E076 EB                     XCHG            in DE
080 E077 210000                 LXI   H,:0000
081 E07A 39                     DAD   SP        Stack start
082 E07B C37FE0                 JMP   :E07F     Into loop
083
084                     * Loop:
085
086 E07E 09             RFR30   DAD   B         Up 1 frame
087 E07F 7E             RFR40   MOV   A,M
088 E080 23                     INX   H
089 E081 B6                     ORA   M
090 E082 CA9FE0                 JZ    :E09F     Jump if top of stack
091 E085 7E                     MOV   A,M
092 E086 2B                     DCX   H
093 E087 BA                     CMP   D         Comp top byte variable addr
094 E088 C27EE0                 JNZ   :E07E     Again if not the same
095 E08B 7E                     MOV   A,M
096 E08C 93                     SUB   E
097 E08D C27EE0                 JNZ   :E07E     Cont if bottombyte different
098 E090 E5                     PUSH  H         Frame bottom to be removed
099 E091 210200                 LXI   H,:0002
100 E094 39                     DAD   SP        Update stackpointer
101 E095 54                     MOV   D,H       ) DE is bottom of area to be
102 E096 5D                     MOV   E,L       ) moved
103 E097 09                     DAD   B         Add 1 frame
104 E098 44                     MOV   B,H       ) Place to move area to
105 E099 4D                     MOV   C,L       )
106 E09A E1                     POP   H         Top area to move
107 E09B CD4FDE                 CALL  :DE4F     Remove old frame
108 E09E F9                     SPHL            New stack position
109 E09F C314E1         RFR50   JMP   :E114     Use common mode to re-
110                                               instate textpointer
111
112                     * If INT variable:
113
114 E0A2 E7             RFR70   RST   4         Subtract 'from'
115 E0A3 51                     DATA  :51
116 E0A4 210601                 LXI   H,:0106   Get addr. LSTPF
117 E0A7 3680                   MVI   M,:80     Default step implicit
118 E0A9 0A                     LDAX  B         Get evt STEP value
119 E0AA 03                     INX   B
120 E0AB FEFF                   CPI   :FF
121 E0AD CA61E0                 JZ    :E061     If no step given
122
123                     * If STEP:
124
125 E0B0 CD18C0                 CALL  :C018     Save 'to-from' on stack
```

```
126 E0B3 34              INR   M          Stepflag explicit
127 E0B4 0B              DCX   B
128 E0B5 CD08E8          CALL  :E808      Get stepvalue in MACC
129 E0B8 210701          LXI   H,:0107    Addr. stepvalue if explicit
130 E0BB E7              RST   4          Stepvalue in LSTEP
131 E0BC 0F              DATA  :0F
132 E0BD CD1BC0          CALL  :C01B      'to-from' range in MACC
133 E0C0 E7              RST   4          Find nr of iterations
134 E0C1 57              DATA  :57
135 E0C2 C361E0          JMP   :E061      Handle loop
136                *
137                ****************************************
138                * RUN basiccmd NEXT <named variable> *
139                ****************************************
140                *
141 E0C5 D1        RNEXI POP   D          Returnaddr
142 E0C6 CD63E9          CALL  :E963      Get varptr in HL
143 E0C9 EB        RNX10 XCHG
144 E0CA 2A0401          LHLD  :0104      Get current loop variable
145 E0CD 7D              MOV   A,L
146 E0CE B4              ORA   H          Loopvariable 0?
147 E0CF CA2EDA          JZ    :DA2E      Then run error 'NEXT
148                                       WITHOUT FOR'
149 E0D2 CD14DE          CALL  :DE14      Compare loop and named
150                                       variable pntrs
151 E0D5 CAEEE0          JZ    :E0EE      Perform NEXT if identical
152 E0D8 EB              XCHG
153 E0D9 221901          SHLD  :0119      Store in scratch area
154 E0DC CD67E1          CALL  :E167      Re-instate next loopvariable
155 E0DF 2A1901          LHLD  :0119      Get it back
156 E0E2 C3C9E0          JMP   :E0C9      Try for a match
157                *
158                **********************
159                * RUN basiccmd NEXT *
160                **********************
161                *
162                * No variable name is given.
163                *
164 E0E5 D1        RNEXT POP   D          Returnaddr
165 E0E6 2A0401          LHLD  :0104      Get current loop variable
166 E0E9 7D              MOV   A,L
167 E0EA B4              ORA   H          Loopvar is 0?
168 E0EB CA2EDA          JZ    :DA2E      Then run error 'NEXT
169                                       WITHOUT FOR'
170 E0EE 3A0601    RNX20 LDA   :0106      Get LSTPF
171 E0F1 B7              ORA   A
172 E0F2 FA33E1          JM    :E133      Jump if INT loop variable
173
174                * If FPT loop variable:
175
176 E0F5 1F              RAR
177 E0F6 DA1DE1          JC    :E11D      Jump if explicit step
178 E0F9 CD06C0          CALL  :C006      Incr. variable in memory
179 E0FC 210E01    RNX30 LXI   H,:010E    Addr lobyte LCOUNT
180 E0FF 7E        RNX40 MOV   A,M        Get lobyte
181 E100 D601            SUI   :01
182 E102 77              MOV   M,A        Decr it
183 E103 D214E1          JNC   :E114      Continu if no overflow
184 E106 2B              DCX   H          Pnts to next byte LCOUNT
185 E107 7D              MOV   A,L
186 E108 FE0A            CPI   :0A        Hibyte done?
187 E10A C2FFE0          JNZ   :E0FF      More bytes if not
```



```
188
189                * Loop finished:
190
191 E10D CD67E1          CALL  :E167      Pop frame
192 E110 B7              ORA   A          No special action
193 E111 C38FC8          JMP   :C88F      Exit to Basic monitor
194
195                * More time round (entry from 'FOR'):
196
197 E114 2A0F01    RNX50 LHLD  :010F      Get pntr to start loop
198 E117 44              MOV   B,H        ) in BC
199 E118 4D              MOV   C,L        )
200 E119 B7              ORA   A          No special action
201 E11A C38FC8          JMP   :C88F      Exit to Basic monitor
202
203                * Explicit step:
204
205 E11D E7        RNX60 RST   4          Get value loopvar in MACC
206 E11E 0C              DATA  :0C
207 E11F E5              PUSH  H
208 E120 210701          LXI   H,:0107    Addr. LSTEP
209 E123 D228E1          JNC   :E128      If INT
210 E126 E7              RST   4          FPT: add stepvalue
211 E127 00              DATA  :00
212 E128 DA2DE1    LOE19 JC    :E12D      If FPT
213 E12B E7              RST   4          INT: add stepvalue
214 E12C 4E              DATA  :4E
215 E12D E1        LOE20 POP   H
216 E12E E7              RST   4          Store new value in
217 E12F 0F              DATA  :0F        variable
218 E130 C3FCE0          JMP   :E0FC      Test end of loop
219
220                * If INT loopvariable:
221
222 E133 EA1DE1    RNX70 JPE   :E11D      Jump if explicit step
223 E136 CD0FC0          CALL  :C00F      Incr. variable in memory
224 E139 C3FCE0          JMP   :E0FC      Test end of loop
225                *
226                **************
227                * PUSH FRAME *
228                **************
229                *
230                * Several pointers are save on stack during
231                * execution of FOR-NEXT loops.
232                *
233 E13C D1        PUSHF POP   D          Get addr. where to continue
234 E13D 2A0401          LHLD  :0104      Get current loop variable
235 E140 7C              MOV   A,H
236 E141 B5              ORA   L          Check if 0
237 E142 CA64E1          JZ    :E164      Then abort routine
238 E145 2A0F01          LHLD  :010F
239 E148 E5              PUSH  H          Save pntr to start loop
240 E149 2A1101          LHLD  :0111
241 E14C E5              PUSH  H          Save pntr to start loop line
242 E14D 2A0B01          LHLD  :010B
243 E150 E5              PUSH  H          ) Save loop iteration count
244 E151 2A0D01          LHLD  :010D      ) (4 bytes)
245 E154 E5              PUSH  H          )
246 E155 2A0701          LHLD  :0107
247 E158 E5              PUSH  H          ) Save step value
248 E159 2A0901          LHLD  :0109      ) (4 bytes)
249 E15C E5              PUSH  H          )
```

```
250 E15D 3A0601          LDA   :0106
251 E160 F5              PUSH  PSW        Save LSTPF
252 E161 2A0401          LHLD  :0104
253 E164 E5        PU1   PUSH  H          Save current loop variable
254 E165 EB              XCHG             Addr. to continue in HL
255 E166 E9              PCHL             Set program counter
256                *
257                **************
258                * POP FRAME *
259                **************
260                *
261                * Restores looppointers in RAM.
262                *
263 E167 D1        POPF  POP   D
264 E168 E1              POP   H
265 E169 220401          SHLD  :0104      Restore LOPVAR
266 E16C 7C              MOV   A,H
267 E16D B5              ORA   L          LOPVAR=0?
268 E16E CA8DE1          JZ    :E18D      Then abort routine
269 E171 F1              POP   PSW
270 E172 320601          STA   :0106      Restore LSTPF
271 E175 E1              POP   H          )
272 E176 220901          SHLD  :0109      ) Restore LSTEP
273 E179 E1              POP   H          ) (4 bytes)
274 E17A 220701          SHLD  :0107
275 E17D E1              POP   H          )
276 E17E 220D01          SHLD  :010D      ) Restore LCOUNT
277 E181 E1              POP   H          ) (4 bytes)
278 E182 220B01          SHLD  :010B
279 E185 E1              POP   H
280 E186 221101          SHLD  :0111      Restore LOPLN
281 E189 E1              POP   H
282 E18A 220F01          SHLD  :010F      Restore LOPPT
283 E18D D5        PP1   PUSH  D
284 E18E C9              RET
285                *
286                **********************************
287                * RUN basiccmds DATA - REM - IMP *
288                **********************************
289                *
290                * RREM/RDATA:
291                *    Entry: B: Points to length byte of string.
292                *    Exit:  DEHL preserved. AF corrupted.
293                * RIMP:
294                *    No action.
295                *
296                RREM
297 E18F 0A        RDATA LDAX  B          Get length of string
298
299                * Entry for REXPS:
300
301 E190 03        RRM10 INX   B          BC points to 1st char
302 E191 81              ADD   C
303 E192 4F              MOV   C,A        BC points to end of string
304 E193 D0              RNC
305 E194 04              INR   B          If overflow: correct B
306
307                * Entry for RUN IMP:
308
309 E195 B7        RIMP  ORA   A          No special action
310 E196 C9              RET
311                *
```



```
312                **********************
313                * RUN basiccmd LIST *
314                **********************
315                *
316                * The whole textbuffer contents is listed.
317                *
318                RLIST
319 E197 3EFF      RLIS0 MVI   A,:FF      Init. mode 0
320 E199 EF              RST   5          Change mode
321 E19A 18              DATA  :18
322 E19B 3E0C            MVI   A,:0C
323 E19D EF              RST   5          Clear screen
324 E19E 03              DATA  :03
325 E19F 2A9F02    RLS05 LHLD  :029F      Get startaddr. textbuf
326 E1A2 EB              XCHG             in DE
327 E1A3 2AA102          LHLD  :02A1      Get start symtab
328 E1A6 2B              DCX   H          End textbuf in HL
329 E1A7 C3CFE1          JMP   :E1CF      Perform listing
330                *
331                ************************************
332                * RUN basiccmd LIST <linenumber> *
333                ************************************
334                *
335                * Entry: BC points to linenumber.
336                * Exit:  BC updated. AFDEHL corrupted.
337                *
338 E1AA CDE7E6    RLIS1 CALL  :E6E7      Read linenr and find
339                                       it in textbuf
340 E1AD 00                NOP
341 E1AE 54              MOV   D,H        ) Linenr in DE
342 E1AF 5D              MOV   E,L        )
343 E1B0 DC39DE          CC    :DE39      If linenr found: calc addr
344                                       after string in HL
345 E1B3 C3CFE1          JMP   :E1CF      Perform listing
346                *
347                ******************************
348                * RUN basiccmd LIST <range> *
349                ******************************
350                *
351                * Entry: BC points to 1st linenumber.
352                * Exit:  BC updated. AFDEHL corrupted.
353                *
354 E1B6 2A9F02    RLIS2 LHLD  :029F      Get start textbuf
355 E1B9 CDD9E6          CALL  :E6D9      Read 1st linenr
356 E1BC C4F6CA          CNZ   :CAF6      If given: Find it in textbuf
357 E1BF EB              XCHG             Addr in DE
358 E1C0 2AA102          LHLD  :02A1      Get start symtab
359 E1C3 2B              DCX   H          End textbuf in HL
360 E1C4 CDD9E6          CALL  :E6D9      Read 1st linenr
361 E1C7 37              STC
362 E1C8 3F              CMC
363 E1C9 C4F6CA          CNZ   :CAF6      If 1st nr found: find 2nd
364 E1CC DC39DE          CC    :DE39      If found: Calc addr after
365                                       string in HL
366
367                * Perform listing:
368
369 E1CF C5        RLS10 PUSH  B
370 E1D0 42              MOV   B,D        ) Start listed area in BC
371 E1D1 4B              MOV   C,E        )
372 E1D2 221B01          SHLD  :011B      Store end listed area
373 E1D5 EB              XCHG             also in DE
```

```
374 E1D6 221901              SHLD  :0119     Store start listed area
375 E1D9 60          RLS20   MOV   H,B       ) Startaddr in HL
376 E1DA 69                  MOV   L,C       )
377 E1DB CD14DE              CALL  :DE14     Check if all lines listed
378 E1DE CAF2E1              JZ    :E1F2     Abort if ready
379 E1E1 CD73D8              CALL  :D873     List curent line if linenr
                                             correct
380
381 E1E4 CDBBD6              CALL  :D6BB     Scan keyboard
382 E1E7 DAF2E1              JC    :E1F2     Break pressed: stop listing
383 E1EA 00                  NOP
384 E1EB 00                  NOP
385 E1EC C4DAD6              CNZ   :D6DA     If a key is pressed:
                                             Wait for spacebar
386
387 E1EF D2D9E1              JNC   :E1D9     No break: list further
388
389                 * If ready:
390
391 E1F2 B7          RLS30   ORA   A         No special action
392 E1F3 C1                  POP   B
393 E1F4 C9                  RET
394                 *
395                 *
396                 *
397 E1F5                     END
```

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

L0E19  E128   L0E20  E12D   POPF   E167   PP1    E18D
PU1    E164   PUSHF  E13C   RDATA  E18F   RFOR   E02B
RFR10  E05F   RFR20  E061   RFR30  E07E   RFR40  E07F
RFR50  E09F   RFR70  E0A2   RIMP   E195   RLIS0  E197
RLIS1  E1AA   RLIS2  E1B6   RLIST  E197   RLS05  E19F
RLS10  E1CF   RLS20  E1D9   RLS30  E1F2   RNEXI  E0C5
RNEXT  E0E5   RNX10  E0C9   RNX20  E0EE   RNX30  E0FC
RNX40  E0FF   RNX50  E114   RNX60  E11D   RNX70  E133
RREM   E18F   RRM10  E190   RWT40  E01C   RWTET  E016
```



```
002                          ORG   :E1F5
003                 *
004                 *
005                 *
006                 **********************
007                 * RUN basiccmd EDIT *
008                 **********************
009                 *
010                 * The editbuffer is set up by moving the program
011                 * to the end of the free RAM space. All memory
012                 * between heapstart and textbegin is used as
013                 * editbuffer.
014                 * On break + space: The edited area is deleted
015                 * from the textbuffer and the program is moved
016                 * to just after the end of the edited text by
017                 * changing the heapsize.
018                 * EFSW is set for input from editbuffer.
019                 *
020 E1F5 CD65E2     REDIT   CALL  :E265     Init. edit buffer
021 E1F8 CD9FE1             CALL  :E19F     List into edit buffer
022 E1FB AF         RED05   XRA   A
023 E1FC 323101              STA   :0131     Set output to screen
024 E1FF 32B902              STA   :02B9     Enable complete keyb.scan
025 E202 CD75DD              CALL  :DD75     0 on end of buffer
026 E205 210000              LXI   H,:0000
027 E208 22B400              SHLD  :00B4     Clear tab table pntr
028 E20B EF                  RST   5         Init screen editor
029 E20C 2A                  DATA  :2A
030 E20D CDBED6      RED10   CALL  :D6BE     Get char from keyboard
031 E210 DA1BE2              JC    :E21B     If break pressed
032 E213 CA0DE2              JZ    :E20D     Wait for inputs
033 E216 EF                  RST   5         Obey character
034 E217 2D                  DATA  :2D
035 E218 C30DE2              JMP   :E20D     Get next input
036
037                 * If Break pressed:
038
039 E21B 3E0C      RED19   MVI   A,:0C
040 E21D EF                  RST   5         Clear screen
041 E21E 03                  DATA  :03
042 E21F CDBED6      RED20   CALL  :D6BE     Get char from keyboard
043 E222 DA4DE2              JC    :E24D     If again Break
044
045                 * Break followed by any character:
046
047 E225 CA1FE2              JZ    :E21F     Wait for a char typed in
048 E228 3E02                MVI   A,:02
049 E22A 323501              STA   :0135     EFSW: input from buffer
050 E22D 2A1B01              LHLD  :011B     Get end listed area
051 E230 EB                  XCHG            in DE
052 E231 2A1901              LHLD  :0119     Get start listed area
053 E234 CD1ADE              CALL  :DE1A     Calc. length listed area
054 E237 00                  NOP
055 E238 CDD1C9              CALL  :C9D1     Delete edited area in txtbuf
056 E23B 2A9B02              LHLD  :029B     Get start HEAP
057 E23E EB                  XCHG            in DE
058 E23F 2AA400              LHLD  :00A4     Get input pntr editbuf
059 E242 CD1ADE              CALL  :DE1A     Calc length used edit area
060 E245 23                  INX   H
061 E246 23                  INX   H
062 E247 EB                  XCHG            DE: length edit area +2
063 E248 CD95D1              CALL  :D195     Program to end of editbuf
```

```
064 E24B B7              ORA   A          No special conditions
065 E24C C9              RET
066
067              * Break followed by 2nd break:
068
069 E24D CDCADE  RED30   CALL  :DECA       Restore original Heap +
070                                        program buffers
071 E250 B7              ORA   A          No special conditions
072 E251 C9              RET
073 E252 00              NOP
074              *
075              ******************************
076              * RUN basiccmd EDIT <linenr> *
077              ******************************
078              *
079 E253 CD65E2  REDI1   CALL  :E265       Init edit buffer
080 E256 CDAAE1          CALL  :E1AA       List one line
081 E259 C3FBE1          JMP   :E1FB       Into Run edit
082              *
083              ******************************
084              * RUN basiccmd EDIT <range> *
085              ******************************
086              *
087 E25C CD65E2  REDI2   CALL  :E265       Init edit buffer
088 E25F CDB6E1          CALL  :E1B6       List part of program
089 E262 C3FBE1          JMP   :E1FB       Into Run edit
090              *
091              ******************************
092              * INITIALISE SCREEN EDITOR *
093              ******************************
094              *
095              * Sets up a mode 0 screen, clears all variables
096              * and arrays. Moves Basic program to top of free
097              * memory, initialises edit pointers.
098              *
099              * Exit: BC preserved. AFDEHL corrupted.
100              *
101 E265 3EFF    REDIN   MVI   A,:FF
102 E267 EF              RST   5          Change screen to mode 0
103 E268 18              DATA  :18
104 E269 CD6DD8          CALL  :D86D       Run 'OUT OF MEMORY' error
105                                        if insufficient space. Else
106                                        empty HEAP + variables
107 E26C CD51EB          CALL  :EB51       Calc free RAM space
108 E26F EB              XCHG              in DE
109 E270 2A9D02          LHLD  :029D       Get HEAPsize
110 E273 19              DAD   D
111 E274 EB              XCHG              Total 'free' RAM in DE
112 E275 CD95D1          CALL  :D195       Program to end free RAM
113 E278 2A9F02          LHLD  :029F       Get startaddr. textbuf
114 E27B 2B              DCX   H
115 E27C 2B              DCX   H           Minus 2
116 E27D 22A600          SHLD  :00A6       Store end available space
117 E280 2A9B02          LHLD  :029B       Get startaddr HEAP
118 E283 23              INX   H
119 E284 23              INX   H           Plus 2
120 E285 22A200          SHLD  :00A2       Store startaddr. editbuf
121 E288 22A400          SHLD  :00A4       Set input pntr editbuf
122 E28B 213101          LXI   H,:131
123 E28E 3602            MVI   M,:02       Set output to editbuf
124 E290 C9              RET
125              *
```



```
126              ****************************
127              * INPUT FROM EDIT BUFFER *
128              ****************************
129              *
130              * Entry: A=0, CY=0.
131              *
132 E291 F5      IFBNL   PUSH  PSW
133 E292 2AA200          LHLD  :00A2       Get startaddr. editbuf
134 E295 223201          SHLD  :0132       Store it in EFEPT
135 E298 7E      IFB10   MOV   A,M         Get char from editbuf
136 E299 B7              ORA   A           Char is 0?
137 E29A CAAAE2          JZ    :E2AA       Then editbuf empty
138 E29D 23              INX   H
139 E29E FE0D            CPI   :0D         Car.ret?
140 E2A0 C298E2          JNZ   :E298       Get next char if not
141
142              * If char is car.ret:
143
144 E2A3 22A200          SHLD  :00A2       Update startaddr. editbuf
145 E2A6 0E00            MVI   C,:00       1st pos on line
146 E2A8 F1              POP   PSW         No special action
147 E2A9 C9              RET
148
149              * If buffer empty:
150
151 E2AA 323501  IFB20   STA   :0135       Set input from keyboard
152 E2AD CDCADE          CALL  :DECA       Organise HEAP + buffers
153 E2B0 F1              POP   PSW         special action
154 E2B1 37              STC               CY=1
155 E2B2 C9              RET
156              *
157              ***********************
158              * RUN basiccmd PRINT *
159              ***********************
160              *
161              * Entry: BC: Position in textbuffer.
162              *
163 E2B3 0A      RPRINT  LDAX  B           Get length
164 E2B4 03              INX   B
165 E2B5 57              MOV   D,A         Count of expr in D
166 E2B6 B7              ORA   A
167 E2B7 CAF7E2          JZ    :E2F7       Jump if only car.ret
168 E2BA D5      RPR10   PUSH  D           Save nr of expressions
169 E2BB 0A              LDAX  B           Get expr type
170 E2BC 03              INX   B
171 E2BD FE20            CPI   :20
172 E2BF CAD2E2          JZ    :E2D2       Jump if string
173
174              * If INT/FPT number:
175
176 E2C2 FE00            CPI   :00
177 E2C4 CD08E8          CALL  :E808       Eval expr. Result in MACC
178 E2C7 F5              PUSH  PSW         Save flags on expr.type
179 E2C8 C453DB          CNZ   :DB53       If INT: print INT number
180 E2CB F1              POP   PSW
181 E2CC CC59DB          CZ    :DB59       If FPT: print FPT number
182 E2CF C3E3E2          JMP   :E2E3
183
184              * If string:
185
186 E2D2 CD91E7  RPR40   CALL  :E791       Evaluate string expr
187 E2D5 E5              PUSH  H
```

```
188 E2D6 EF                RST   5          Ask cursor pos and size
189 E2D7 0C                DATA  :0C        char screen
190 E2D8 7B                MOV   A,E        X-size of screen in A
191 E2D9 95                SUB   L          Minus X-coord cursor pos
192 E2DA 3C                INR   A          +1
193 E2DB E1                POP   H
194 E2DC BE                CMP   M          (not used further: off
195 E2DD 00                NOP              screen printing possible)
196 E2DE 00                NOP              (should be:  CC :DD5E)
197 E2DF 00                NOP
198 E2E0 CD32DB            CALL  :DB32      Print string pntd by HL
199                *
200 E2E3 0A        RPR50   LDAX  B          Get byte after string
201 E2E4 03                INX   B
202 E2E5 FEFF              CPI   :FF        End marker?
203 E2E7 CAF6E2            JZ    :E2F6      Then quit with car.ret
204 E2EA FE3B              CPI   :3B        ';'?
205 E2EC C40DDB            CNZ   :DB0D      Cursor to next column if
206                                         not (must be ',')
207 E2EF D1                POP   D          Get nr of expr to print
208 E2F0 15                DCR   D          Update expr count
209 E2F1 C2BAE2            JNZ   :E2BA      Loop if more expressions
210 E2F4 B7                ORA   A          No special action
211 E2F5 C9                RET
212
213                * If end of print statement:
214
215 E2F6 D1        RPR70   POP   D
216 E2F7 CD5EDD    RPR80   CALL  :DD5E      Print 'CR'
217 E2FA B7                ORA   A
218 E2FB C9                RET
219                *
220                **********************
221                * RUN basiccmd INPUT *
222                **********************
223                *
224                * Runs an input statement with a prompt ('?').
225                * RINPQ:  Input with a string.
226                * RINPUT: Input without a string.
227                *
228 E2FC CD91E7    RINPQ   CALL  :E791      Evaluate stringexpression
229 E2FF CD32DB            CALL  :DB32      Print string pntd by HL
230 E302 210200    RINPUT  LXI   H,:0002
231 E305 39                DAD   SP
232 E306 CD47E4            CALL  :E447      Update ERSSP, print '?' and
233                                         ask for inputs
234 E309 D464E3            CNC   :E364      If no break: store inputs
235 E30C F5                PUSH  PSW        Save last input
236 E30D AF                XRA   A
237 E30E 321701            STA   :0117      Reset flag running inputs
238 E311 F1                POP   PSW        Get last input
239 E312 DA20E3            JC    :E320      Abort if break
240 E315 1C                INR   E
241 E316 CA1EE3            JZ    :E31E      Quit if correct nr of inputs
242 E319 CDFFDA            CALL  :DAFF      Else: Print
243 E31C 08D2              DBL   :D208      'SOME INPUT IGNORED'
244 E31E B7        RIP10   ORA   A          No special action
245 E31F C9                RET
246
247                * If suspended:
248
249 E320 3E02      RIP20   MVI   A,:02      Code 'cmd broken in'
```



```
250 E322 C9                RET
251                *
252                **********************
253                * RUN basiccmd READ *
254                **********************
255                *
256 E323 3E01      RREAD   MVI   A,:01
257 E325 323501            STA   :0135      Set input from string
258 E328 3A2301            LDA   :0123      Get offset next char to en-
259 E32B 5F                MOV   E,A        code and store it in E
260 E32C 2A9102            LHLD  :0291      Get DATAQ addr
261 E32F 7E                MOV   A,M        Get length of string
262 E330 323401            STA   :0134      Store it in EFECT
263 E333 23                INX   H          Pnts to 1st char
264 E334 223201            SHLD  :0132      Addr 1st char in EFEPT
265 E337 CD64E3            CALL  :E364      Store data
266 E33A 7B                MOV   A,E        Get offset next char
267 E33B 322301            STA   :0123      Store it
268 E33E 2A3201            LHLD  :0132      Get EFEPT
269 E341 2B                DCX   H          Pnts to next dataline
270 E342 229102            SHLD  :0291      Store it in DATAQ
271 E345 AF                XRA   A
272 E346 323501            STA   :0135      Set input from keyboard
273 E349 C9                RET
274                *
275                **********************
276                * DATA - (not used) *
277                **********************
278                *
279 E34A 21        LOE351  DATA  :21
280 E34B 06                DATA  :06
281 E34C 60                DATA  :60
282 E34D 69                DATA  :69
283 E34E 22                DATA  :22
284 E34F 1F                DATA  :1F
285                *
286                *****************
287                * RESTART INPUT *
288                *****************
289                *
290                * Error handling if running inputs.
291                *
292 E350 2A1D01    INPRS   LHLD  :011D      Get saved stackpntr
293 E353 F9                SPHL             Reload stackpntr
294 E354 2A0201            LHLD  :0102      Get start current command
295 E357 44                MOV   B,H        ) Store it in BC
296 E358 4D                MOV   C,L        )
297 E359 CDFFDA            CALL  :DAFF      Print 'RETYPE LINE'
298 E35C 93DB              DBL   :DB93
299 E35E C37FC8            JMP   :C87F      Re-execute input statement
300                *
301                ***********************************
302                * STORE DATA IN CORRECT LOCATION *
303                ***********************************
304                *
305                * Entry LOE56 not used.
306                *
307                * Stores data read from datastatements or gotten
308                * from an input line on the correct location.
309                *
310                * Entry: E : Offset next character to encode
311                *            (#0291).
```

```
                        *       HL: Startaddress data line.
                        *
E361 CD36E4     LOE56   CALL    :E436       (not used)
                        *
E364 0A         RRDIP   LDAX    B           Get nr of data reqd
E365 03                 INX     B
E366 57                 MOV     D,A         Into D
E367 15         RRI10   DCR     D
E368 FAC1E3             JM      :E3C1       Jump if ready
E36B 1C         RRI20   INR     E           Offset nex char
E36C CAA8E3             JZ      :E3A8       Jump if at end of line
E36F 1D                 DCR     E
E370 CD63E9     RRI30   CALL    :E963       Get varptr in HL
E373 C5                 PUSH    B
E374 D5                 PUSH    D
E375 4B                 MOV     C,E         Offset in C
E376 E5                 PUSH    H           Save varptr
E377 213E01             LXI     H,:013E     Startaddr EBUF
E37A E5                 PUSH    H           Save it on stack
E37B E630               ANI     :30
E37D CF                 RST     1           Encode a constant
E37E 06                 DATA    :06
E37F 59                 MOV     E,C         Offset in E
E380 C1                 POP     B
E381 E1                 POP     H           Get varptr
E382 FE20               CPI     :20         String type ?
E384 7B                 MOV     A,E         Offset in A
E385 CAA2E3             JZ      :E3A2       Jump if string type
E388 CDB4E4             CALL    :E4B4       Perform LET (INT/FPT)

                        * Check separator/terminator:

E38B 4F         RRI40   MOV     C,A         Offset in C
E38C CDD2DD             CALL    :DDD2       Get char from line, neglect
                                            tab and space
E38F 0C                 INR     C           Offset +1
E390 FE2C               CPI     :2C         ',' ?
E392 CA99E3             JZ      :E399
E395 FE0D               CPI     :0D         CR ?
E397 0EFF               MVI     C,:FF       Dummy offset for end of
                                            line
E399 D1         RRI50   POP     D
E39A 59                 MOV     E,C         Offset in E
E39B C1                 POP     B
E39C C2C3E3             JNZ     :E3C3       Error if char not ',' or
                                            car.ret
E39F C367E3             JMP     :E367       Read next data line

                        * If string type:

E3A2 CDB8E4     RRI60   CALL    :E4B8       Perform LET (STR)
E3A5 C38BE3             JMP     :E38B       Check terminator/separator

                        * If end of line reached ('CR'):

E3A8 3A1701     RRI70   LDA     :0117       Get flag for running inputs
E3AB B7                 ORA     A
E3AC CABBE3             JZ      :E3BB       Quit if not running inputs
E3AF CD55DD             CALL    :DD55       Cursor to begin next line
E3B2 CDD0E3             CALL    :E3D0       Print '?', read input line
E3B5 DAC2E3             JC      :E3C2       Abort if break pressed
E3B8 C370E3             JMP     :E370       Cont reading
```



```
                        * If whole line read:

E3BB CDDCE3     RRI80   CALL    :E3DC       Find next dataline in txtbuf
E3BE C370E3             JMP     :E370       Cont reading

                        * If reading done:

E3C1 B7         RRI90   ORA     A           No special action
E3C2 C9         RRI99   RET

                        * If error:

E3C3 2A3201     RRISN   LHLD    :0132       Get EFEPT
                        LXI     D,:FFFC
                        DAD     D           EFEPT-4
E3C9 19                 SHLD    :0100       Store start current line
E3CA 220001             JMP     :DA0B       Run 'SYNTAX ERROR'
E3CD C30BDA             *
                        ****************************
                        * PREPARE GETTING INPUTS *
                        ****************************
                        *
                        * Continuation of 0E447.
                        * Prints a prompt ('?') on the screen and gets a
                        * textline from input.
                        *
E3D0 D5         INPGT   PUSH    D
E3D1 EF                 RST     5           Ask cursor pos. and size
E3D2 0C                 DATA    :0C         char. screen
E3D3 D1                 POP     D
E3D4 3E3F               MVI     A,:3F
E3D6 CDA0C6             CALL    :C6A0       Print '?'; input a textline
E3D9 5D                 MOV     E,L
E3DA 1C                 INR     E           E pnts after last char of
                                            input
E3DB C9                 RET
                        *
                        ************************************
                        * FIND NEXT DATALINE IN TEXTBUFFER *
                        ************************************
                        *
E3DC F5         DATAF   PUSH    PSW
E3DD D5                 PUSH    D
E3DE E5                 PUSH    H
E3DF 2A2401             LHLD    :0124       Get addr current dataline
E3E2 7E         DTF10   MOV     A,M         Get length data string
E3E3 B7                 ORA     A
E3E4 3E02               MVI     A,:02
E3E6 CAF5D9             JZ      :D9F5       Run error 'OUT OF DATA' if
                                            no data available
E3E9 54                 MOV     D,H         ) Addr dataline in DE
E3EA 5D                 MOV     E,L         )
E3EB CD39DE             CALL    :DE39       Calc end dataline
E3EE 222401             SHLD    :0124       Store it in DATAP
E3F1 EB                 XCHG                End in DE, start in HL
E3F2 23                 INX     H
E3F3 23                 INX     H
E3F4 23                 INX     H           Pnts to token
E3F5 7E                 MOV     A,M         Get token
E3F6 FEA2               CPI     :A2         Is it DATA?
E3F8 EB                 XCHG
```

```
436 E3F9 C2E2E3             JNZ   :E3E2     Check next textline if not
437 E3FC EB                 XCHG
438 E3FD C336E4             JMP   :E436     Continu
439                 *
440 E400 FF                 DATA  :FF
441                 *
442                 *
443                 *
444 E401                    END

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

DATAF  E3DC   DTF10  E3E2   IFB10  E298   IFB20  E2AA
IFBNL  E291   INPGT  E3D0   INPRS  E350   L0E351 E34A
L0E56  E361   RED05  E1FB   RED10  E20D   RED19  E21B
RED20  E21F   RED30  E24D   REDI1  E253   REDI2  E25C
REDIN  E265   REDIT  E1F5   RINP0  E2FC   RINPUT E302
RIP10  E31E   RIP20  E320   RPR10  E2BA   RPR40  E2D2
RPR50  E2E3   RPR70  E2F6   RPR80  E2F7   RPRINT E2B3
RRDIP  E364   RREAD  E323   RRI10  E367   RRI20  E36B
RRI30  E370   RRI40  E38B   RRI50  E399   RRI60  E3A2
RRI70  E3A8   RRI80  E3BB   RRI90  E3C1   RRI99  E3C2
RRISN  E3C3
```



```
002                         ORG   :E401
003                 *
004                 *
005                 *
006                 ************************
007                 * RUN basiccmd RESTORE *
008                 ************************
009                 *
010 E401 2A9F02     RREST   LHLD  :029F     Get startaddr. textbuf
011 E404 222401             SHLD  :0124     Store it in DATAP
012 E407 AF                 XRA   A
013 E408 3D                 DCR   A
014 E409 C352E4             JMP   :E452     Set DATA0=FF
015                 *
016                 ******************
017                 * RUN RANDOMISE *
018                 ******************
019                 *
020                 * Returns a hardware random number 0 < R < 1.
021                 *
022                 * Exit: BC preserved. AFDEHL corrupted.
023                 *       Result in MACC.
024                 *
025 E40C C5         RRAND   PUSH  B
026 E40D 2100FD             LXI   H,:FD00   Addr PORI
027 E410 CD5FD6             CALL  :D65F     BC=0, E=0, #40 or #80
028 E413 1601               MVI   D,:01
029                 *
030 E415 3A9302     RMI10   LDA   :0293     Get RNDLY (0 by C72E)
031 E418 CD58D6             CALL  :D658     Delay, A=M XOR E
032 E41B 07                 RLC
033 E41C 07                 RLC
034
035                 * Now CY is 0 if E=#40 and CY = bit 6 of FD00
036                 * (hardware random) if E=0 or E=#80.
037
038 E41D 7A         RMI20   MOV   A,D
039 E41E 17                 RAL             RAL D
040 E41F 57                 MOV   D,A
041 E420 79                 MOV   A,C
042 E421 17                 RAL             RAL C
043 E422 4F                 MOV   C,A
044 E423 78                 MOV   A,B
045 E424 17                 RAL             RAL B
046 E425 47                 MOV   B,A
047 E426 B7                 ORA   A
048 E427 F215E4             JP    :E415     Again if B<#80 (must be
049                                           normalised FPT nr)
050
051                 * Result in MACC:
052
053 E42A 3E01               MVI   A,:01     Set exp.byte for 1 < R < 2
054 E42C E7                 RST   4         Copy A,B,C,D to MACC
055 E42D 12                 DATA  :12
056 E42E 21F1D0             LXI   H,:D0F1   Addr FPT(-1)
057 E431 E7                 RST   4         Add -1 to MACC
058 E432 00                 DATA  :00       Now: 0 < R < 1
059 E433 C1                 POP   B
060 E434 B7                 ORA   A         No special action
061 E435 C9                 RET
062                 *
063                 *
```

```
                        ******************
                        * cont. of OE3DC *
                        ******************
                        *
E436 23         MPT36   INX   H
E437 7E                 MOV   A,M       Get length dataline
E438 323401             STA   :0134     Store it in EFECT
E43B 23                 INX   H         Pnts to start data bytes
E43C 223201             SHLD  :0132     Startaddr string in EFEPT
E43F 00                 NOP
E440 00                 NOP
E441 E1                 POP   H
E442 D1                 POP   D
E443 1E00               MVI   E,:00
E445 F1                 POP   PSW
E446 C9                 RET
                        *
                        ***************************
                        * PREPARE GETTING INPUTS *
                        ***************************
                        *
E447 221D01     MPT5A   SHLD  :011D     Store SP+2 in ERSSP
E44A 3EFF               MVI   A,:FF
E44C 321701             STA   :0117     Set flag for running inputs
E44F C3D0E3             JMP   :E3D0     Print '?', input a textline
                        *
                        ******************
                        * cont. of OE401 *
                        ******************
                        *
E452 322301     MPT32   STA   :0123     Save offset next char to
E455 C9                 RET             encode
                        *
E456 FF                 DATA  :FF
E457 FF                 DATA  :FF
E458 FF                 DATA  :FF
E459 FF                 DATA  :FF
                        *
                        ********************
                        * RUN basiccmd LET *
                        ********************
                        *
                        * Valid as direct command or in program. Computes
                        * variable pointer or variable reference and
                        * checks type. 'LET' can be explicitly or
                        * implicitly given.
                        *
                        * Entry: BC: Points to program line.
                        * Exit:  BC: Updated.
                        *        A:  Type of variable.
                        *        FDEHL corrupted.
                        *
                        RLETX
E45A CD63E9     RLETI   CALL  :E963     Get varptr in HL, T/L in A
E45D E630               ANI   :30       Type only
E45F F5                 PUSH  PSW       Save type
E460 FE20               CPI   :20       String type?
E462 C294E4             JNZ   :E494     Jump if INT/FPT

                        * If string type:

E465 E5         RLT10   PUSH  H         Save varptr
```



```
E466 7E                 MOV   A,M       )
E467 23                 INX   H         )
E468 66                 MOV   H,M       ) Addr string in HL
E469 6F                 MOV   L,A
E46A E3                 XTHL            ) Save stringaddr
E46B E5                 PUSH  H         ) and varptr
E46C CD9DE7             CALL  :E79D     Get value being assigned
E46F 7B                 MOV   A,E       Get status
E470 FE02               CPI   :02
E472 CA7FE4             JZ    :E47F     Jump if temp on heap
E475 7E                 MOV   A,M       Get string length
E476 EB                 XCHG
E477 CD8BD1             CALL  :D18B     Get place in heap for string
E47A E5                 PUSH  H
E47B CD72D1             CALL  :D172     Transfer string into heap
E47E E1                 POP   H
E47F EB         RLT20   XCHG            Pntr to string in DE
E480 E1                 POP   H         Pntr to variable in HL
E481 73                 MOV   M,E       ) Stringpntr in variable
E482 23                 INX   H         )
E483 72                 MOV   M,D       )

                        * Entry from scratch:

E484 2A9F02     RLT25   LHLD  :029F     Get startaddr. textbuf
E487 EB                 XCHG            in DE
E488 E1                 POP   H         Get pntr old value
E489 7C                 MOV   A,H
E48A B5                 ORA   L
E48B C414DE             CNZ   :DE14     If <>0: Test if on heap
E48E DC87D1             CC    :D187     Then clear string in heap
E491 C3B2E4             JMP   :E4B2     Ready

                        * If FPT/INT type:

E494 E5         RLT30   PUSH  H
E495 CD19E8             CALL  :E819     Eval numeric arguments
E498 7C                 MOV   A,H
E499 B5                 ORA   L
E49A C2A3E4             JNZ   :E4A3     Copy num expr if not in MACC
E49D E1                 POP   H
E49E E7                 RST   4         Copy MACC to variable
E49F 0F                 DATA  :0F
E4A0 C3B2E4             JMP   :E4B2     Ready

                        * If just constant or variable:

E4A3 D1         RLT50   POP   D         Get pntr to variable
E4A4 D5                 PUSH  D
E4A5 C5                 PUSH  B
E4A6 0604               MVI   B,:04     Nr of bytes
E4A8 7E         RLT60   MOV   A,M       Get byte
E4A9 12                 STAX  D         Store it in variable
E4AA 23                 INX   H
E4AB 13                 INX   D
E4AC 05                 DCR   B         Decr byte count
E4AD C2A8E4             JNZ   :E4A8     Next byte if not ready
E4B0 C1                 POP   B
E4B1 E1                 POP   H

                        * Ready:
```

```
E4B2 F1          RLT90   POP   PSW
E4B3 C9                  RET

                 * Entry from READ/INPUT for FPT/INT:

E4B4 F5          RLT70   PUSH  PSW
E4B5 C394E4              JMP   :E494      Value to variable

                 * Entry from READ/INPUT for STR:

E4B8 F5          RLT80   PUSH  PSW
E4B9 C365E4              JMP   :E465      Stringvalue to variable
                 *
                 ***********************
                 * RUN basiccmd SOUND *
                 ***********************
                 *
                 * Formats: SOUND <CHAN><ENV><VOL><TG><FREQ>.
                 *          SOUND <CHAN> OFF.
                 *          SOUND OFF.
                 *
                 * Entry: BC: Points to program line.
                 * Exit:  BC updated, AFDEHL corrupted.
                 *
E4BC 0A          RSOUND  LDAX  B          Get 1st expr.
E4BD FEFF                CPI   :FF       Is it sound OFF?
E4BF CA01E5              JZ    :E501     Then turn all sound off
E4C2 3E02                MVI   A,:02
E4C4 CD43E7              CALL  :E743     Get channelnr (0,1,2)
E4C7 21C201              LXI   H,:01C2   Startaddr SCB0
E4CA 110E00              LXI   D,:000E   Length SCB
E4CD F5                  PUSH  PSW       Save channelnr
E4CE 3D          SND10   DCR   A
E4CF FAD6E4              JM    :E4D6     If chan.0 or ready
E4D2 19                  DAD   D         Absolute addr SCB in HL
E4D3 C3CEE4              JMP   :E4CE     If chan.2
E4D6 D1          SND20   POP   D         Channelnr in D
E4D7 CD1FE5              CALL  :E51F     Set up SCB <ENV><VOL>
E4DA DAFCE4              JC    :E4FC     If channel to be OFF
E4DD 3E03                MVI   A,:03
E4DF CD43E7              CALL  :E743     Get <TG>
E4E2 F5                  PUSH  PSW
E4E3 E601                ANI   :01       Tremoloflag only
E4E5 77                  MOV   M,A       Into SCB
E4E6 23                  INX   H
E4E7 00                  NOP
E4E8 00                  NOP
E4E9 F1                  POP   PSW       Restore <TG>
E4EA E602                ANI   :02       Glissandoflag only
E4EC 1F                  RAR
E4ED 3C                  INR   A
E4EE 23                  INX   H         1 free byte
E4EF 77                  MOV   M,A       Glissandoflag in SCB
E4F0 23                  INX   H         )
E4F1 23                  INX   H         ) Ignore current period
E4F2 23                  INX   H         )
E4F3 E5                  PUSH  H         Save pntr to reqd period
E4F4 CDF8E6              CALL  :E6F8     Get <period>
E4F7 EB                  XCHG            in DE
E4F8 E1                  POP   H
E4F9 73                  MOV   M,E       ) Reqd period in SCB
E4FA 23                  INX   H         )
```



```
E4FB 72                  MOV   M,D       )
E4FC CD96D9      SND70   CALL  :D996     Enable sound interrupts
E4FF B7                  ORA   A         No special action
E500 C9                  RET

                 * If SOUND OFF:

E501 03          SND80   INX   B
E502 CD8FD9              CALL  :D98F     Disable sound interrupts
E505 C5                  PUSH  B
E506 CDA6D8              CALL  :D8A6     Stop oscillators
E509 C1                  POP   B
E50A B7                  ORA   A         No special action
E50B C9                  RET
                 *
                 ***********************
                 * RUN basiccmd NOISE *
                 ***********************
                 *
                 * Sets up a noise control block.
                 * Formats: NOISE <ENV><VOL>.
                 *          NOISE OFF.
                 *
                 * Entry: BC: Points to expression.
                 * Exit:  Noise off: BC updated, DE preserved,
                 *                   AFHL corrupted.
                 *        Noise on:  BC updated, AFDEHL corrupted.
                 *
E50C 21EC01      RNOISE  LXI   H,:01EC   Startaddr NCB
E50F 1603                MVI   D,:03     Channelnr.3
E511 CD1FE5              CALL  :E51F     Set up NCB
E514 DAFCE4              JC    :E4FC     If channel to be off
E517 3600                MVI   M,:00     No tremolo
E519 23                  INX   H
E51A 3600                MVI   M,:00     Current volume = 0
E51C C3FCE4              JMP   :E4FC     Enable sound interrupts
                 *
                 ****************
                 * SET ENVELOPE *
                 ****************
                 *
                 * Sets up a sound or noise control block for
                 * a channel.
                 *
                 * Entry: D:  Channelnumber (0,1,2 or 3).
                 *        BC: Points to programline.
                 *        HL: Points to startaddr SCB/NCB.
                 * Exit:  If sound/noise off (CY=1):
                 *        FF in 1st byte of SCB/NCB.
                 *            HL: points to 1st byte SCB/NCB.
                 *            BC: points to 2nd byte SCB/NCB.
                 *            DE preserved, AF corrupted.
                 *        If sound/noise on (CY=0):
                 *        00 in 1st byte SCB/NCB.
                 *            HL: Points after free byte.
                 *            DE: Envelopeaddr +1.
                 *            BC: Points beyond volume.
                 *            AF: corrupted.
                 *
E51F CD8FD9      SNGEV   CALL  :D98F     Disable sound interrupts
E522 3600                MVI   M,:00     00 in 1st byte SCB/NCB
E524 0A                  LDAX  B         Get 1st byte from progr.line
```

```
E525 FEFF            CPI   :FF
E527 CA52E5          JZ    :E552      Jump if channel to be off

                   * If channel on:

E52A 23              INX   H          Pntr to next byte of block
E52B E5              PUSH  H          Save pntr
E52C 3E01            MVI   A,:01
E52E CD43E7          CALL  :E743      Get envelopenr in A (0,1)
E531 21F501          LXI   H,:01F5    Addr envelope area
E534 0F              RRC
E535 0F              RRC
E536 5F              MOV   E,A        DE=64*A
E537 1600            MVI   D,:00
E539 19              DAD   D          Add offset for env area
E53A EB              XCHG             Pntr to envelope in DE
E53B E1              POP   H          Get input pntr SCB/NCB
E53C 73              MOV   M,E        ) Envelope addr in block
E53D 23              INX   H          )
E53E 72              MOV   M,D        )
E53F 23              INX   H
E540 13              INX   D
E541 73              MOV   M,E        ) Env addr +1 in block
E542 23              INX   H          )
E543 72              MOV   M,D        )
E544 23              INX   H
E545 3E0F            MVI   A,:0F
E547 CD43E7          CALL  :E743      Get volume multiplier in
                                      A (0-15)
E54A 07              RLC              )
E54B 07              RLC              ) *8
E54C 07              RLC              )
E54D 77              MOV   M,A        Vol.*8 in block
E54E 23              INX   H          Pntr to basic volume
E54F 23              INX   H          Pntr to tremoloflag
E550 B7              ORA   A          Return 'channel on'
E551 C9              RET

                   * If channel to be off:

E552 03            SNG10   INX   B
E553 35                    DCR   M          FF in 1st byte SCB/NCB
E554 7A                    MOV   A,D        Get channelnr
E555 FE03                  CPI   :03
E557 CA63E5                JZ    :E563      Jump if noisechannel
E55A 0F                    RRC              ) Find disable data for
E55B 0F                    RRC              ) chan. 0-2
E55C F636                  ORI   :36        )
E55E 3206FC                STA   :FC06      Load sound cmd word
E561 37                    STC              Return 'channel off'
E562 C9                    RET

                   * If noise to be off:

E563 3A9502        SNG20   LDA   :0295      Get volume osc.2 + noise
E566 E60F                  ANI   :0F        Vol. noise = 0
E568 329502                STA   :0295      Update POR1M
E56B 3205FD                STA   :FD05      and POR1
E56E 37                    STC              Return 'channel off'
E56F C9                    RET
                   *
                   *
```



```
                   ************************
                   * RUN basiccmd ENVELOPE *
                   ************************
                   *
                   * Load envelope table 0 or 1 with data.
                   * Formats: ENVELOPE <ENV> (<V>,<T>;) <V>,<T> FF.
                   *          ENVELOPE <ENV> (<V>,<T>;) <V>.
                   *
E570 3E01          RENV    MVI   A,:01
E572 CD43E7                CALL  :E743      Get envelope number (0,1)
E575 0F                    RRC
E576 0F                    RRC
E577 5F                    MOV   E,A        ) Offset for env addr
E578 1600                  MVI   D,:00      )
E57A 21F501                LXI   H,:01F5    Addr env storage area
E57D 19                    DAD   D          Startaddr table in HL
E57E 3600                  MVI   M,:00      00 in 1st field
E580 23                    INX   H
E581 0A                    LDAX  B          Get length of expr
E582 03                    INX   B
E583 3C                    INR   A
E584 57                    MOV   D,A        Nr of complete entries in D
E585 15            REN10   DCR   D
E586 CA9DE5                JZ    :E59D      If all entries done
E589 3E10                  MVI   A,:10
E58B CD43E7                CALL  :E743      Get volume (0-16)
E58E 77                    MOV   M,A        Into env table
E58F 23                    INX   H
E590 CD1DE7                CALL  :E71D      Get time
E593 D601                  SUI   :01        Min. value is 1
E595 DA15DA                JC    :DA15      Run error 'NUMBER OUT OF
                                            RANGE' if time=0
E598 77                    MOV   M,A        Time into env table
E599 23                    INX   H
E59A C385E5                JMP   :E585      Next <V>,<T>
E59D 36FF          REN15   MVI   M,:FF      FF as last in env table
E59F 0A                    LDAX  B          Get last expr
E5A0 03                    INX   B
E5A1 FEFF                  CPI   :FF        End marker?
E5A3 CA80E4                JZ    :E480      Then quit
E5A6 0B                    DCX   B
E5A7 3E10                  MVI   A,:10
E5A9 CD43E7                CALL  :E743      Get final volume <V>
E5AC 77                    MOV   M,A        <V> into env table
E5AD 23                    INX   H
E5AE 36FF                  MVI   M,:FF      Time = forever
E5B0 B7            REN20   ORA   A          No special action
E5B1 C9                    RET
                   *
                   ************************
                   * RUN basiccmd CURSOR *
                   ************************
                   *
E5B2 CDF3E5        RCURS   CALL  :E5F3      Evaluate coordinate
E5B5 67                    MOV   H,A        Y-coord in H
E5B6 EF                    RST   5          Set cursor position
E5B7 09                    DATA  :09
E5B8 C3C9E5                JMP   :E5C9      Evt run screen error
                   *
                   **********************
                   * RUN basiccmd MODE *
                   **********************
```

```
                        *
                        * Entry: BC: Points to program line.
                        *
E5BB 0A         RMODE   LDAX  B         Get reqd mode in A
E5BC 03                 INX   B
E5BD C3B5CE             JMP   :CEB5     Change screen mode
                        *
E5C0 C9         LOE99   RET
                        *
                        *********************
                        * RUN basiccmd DOT *
                        *********************
                        *
E5C1 CDF9E5     RDOT    CALL  :E5F9     Eval dot addr + colour
E5C4 C5                 PUSH  B
E5C5 4B                 MOV   C,E       Colour in C
E5C6 EF                 RST   5         Draw dot on screen
E5C7 1E                 DATA  :1E
E5C8 C1         RDT10   POP   B
E5C9 DA02E6     RDT20   JC    :E602     Jump if screen error
E5CC B7                 ORA   A         No special action
E5CD C9                 RET
                        *
                        **********************
                        * RUN basiccmd DRAW *
                        **********************
                        *
E5CE CDE0E5     RDRAW   CALL  :E5E0     Eval begin/end addr +
                                        colour
E5D1 E3                 XTHL
E5D2 EF                 RST   5         Draw a line on screen
E5D3 21                 DATA  :21
E5D4 C3C8E5             JMP   :E5C8     Evt. run screen error
                        *
                        **********************
                        * RUN basiccmd FILL *
                        **********************
                        *
E5D7 CDE0E5     RFILL   CALL  :E5E0     Eval coordinates, colour
E5DA E3                 XTHL
E5DB EF                 RST   5         Fill a rectangular area
E5DC 24                 DATA  :24
E5DD C3C8E5             JMP   :E5C8     Evt run screen error
                        *
                        ************************************
                        * EVALUATE 2 COORDINATES + COLOUR *
                        ************************************
                        *
E5E0 CDF3E5     R2COC   CALL  :E5F3     Get 1st coordinate
E5E3 E3                 XTHL            X-coord on stack
E5E4 E5                 PUSH  H
E5E5 5F                 MOV   E,A       Y-coord in E
E5E6 CDF3E5             CALL  :E5F3     Get 2nd coordinate,
                                        x-coord in HL
E5E9 57                 MOV   D,A       Y-coord in D
E5EA CDFDE5             CALL  :E5FD     Get colour in A
E5ED C5                 PUSH  B
E5EE D5                 PUSH  D
E5EF EB                 XCHG
E5F0 C1                 POP   B
E5F1 E1                 POP   H
E5F2 C9                 RET
```



```
                        *
                        ************************
                        * EVALUATE COORDINATE *
                        ************************
                        *
E5F3 CDF8E6     RCOOR   CALL  :E6F8     Get x-coord in HL
E5F6 C31DE7             JMP   :E71D     Get y-coord in A
                        *
                        ********************************
                        * EVALUATE COORDINATE + COLOUR *
                        ********************************
                        *
E5F9 CDF3E5     RCOCO   CALL  :E5F3     Get x-coord in HL
E5FC 5F                 MOV   E,A       Y-coord in E
E5FD 3E17       RCOL    MVI   A,:17
E5FF C343E7             JMP   :E743     Get colour (0-23) in A
                        *
                        *
                        *
E602                    END
```

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

LOE99   E5C0    MPT32   E452    MPT36   E436    MPT5A   E447
R2COC   E5E0    RCOCO   E5F9    RCOL    E5FD    RCOOR   E5F3
RCURS   E5B2    RDOT    E5C1    RDRAW   E5CE    RDT10   E5C8
RDT20   E5C9    REN10   E585    REN15   E59D    REN20   E5B0
RENV    E570    RFILL   E5D7    RLETI   E45A    RLETX   E45A
RLT10   E465    RLT20   E47F    RLT25   E484    RLT30   E494
RLT50   E4A3    RLT60   E4A8    RLT70   E4B4    RLT80   E4B8
RLT90   E4B2    RMI10   E415    RMI20   E41D    RMODE   E5BB
RNOISE  E50C    RRAND   E40C    RREST   E401    RSOUND  E4BC
SND10   E4CE    SND20   E4D6    SND70   E4FC    SND80   E501
SNG10   E552    SNG20   E563    SNGEV   E51F
```

```
002                     ORG   :E602
003                 *
004                 *
005                 *
006                 ********************
007                 * RUN SCREEN ERROR *
008                 ********************
009                 *
010                 * Entry: A: Error code: 01: Off screen.
011                 *                       02: Colour not available.
012 E602 FE01       SCRER   CPI   :01       Code is 1?
013 E604 3E11               MVI   A,:11
014 E606 CAF5D9             JZ    :D9F5     Then run error 'OFF SCREEN'
015 E609 3E10               MVI   A,:10     Else: run error
016 E60B C3F5D9             JMP   :D9F5     'COLOUR NOT AVAILABLE'
017                 *
018                 ************************
019                 * RUN basiccmd COLORT *
020                 ************************
021                 *
022 E60E CD1CE6     RCOLT   CALL  :E61C     Get colours in scratch area
023 E611 EF                 RST   5         Set text colours
024 E612 06                 DATA  :06
025 E613 B7                 ORA   A         No special action
026 E614 C9                 RET
027                 *
028                 ************************
029                 * RUN basiccmd COLORG *
030                 ************************
031                 *
032 E615 CD1CE6     RCOLG   CALL  :E61C     Get colours in scratch area
033 E618 EF                 RST   5         Set graphic colours
034 E619 1B                 DATA  :1B
035 E61A B7                 ORA   A         No special action
036 E61B C9                 RET
037                 *
038                 ************************************
039                 * GET 4 COLOURS INTO SCRATCH AREA *
040                 ************************************
041                 *
042                 * Colour data from a program line are stored in
043                 * scratch area SCOLT/SCOLG (#0119-011C).
044                 *
045                 * Exit: HL: Points to start scratch area.
046                 *
047 E61C 211901      R4COL   LXI   H,:0119   Startaddr SCOLT/SCOLG area
048 E61F E5                 PUSH  H
049 E620 3E0F       R4C10   MVI   A,:0F
050 E622 CD43E7             CALL  :E743     Get one colour (0-15)
051 E625 77                 MOV   M,A       Store it in scratch area
052 E626 23                 INX   H
053 E627 7D                 MOV   A,L
054 E628 FE1D               CPI   :1D       4 colours done?
055 E62A C220E6             JNZ   :E620     Next if not
056 E62D E1                 POP   H
057 E62E C9                 RET
058                 *
059                 ********************
060                 * RUN basiccmd DIM *
061                 ********************
062                 *
063                 * Entry: BC: points to program line.
```



```
064                 *
065 E62F 0A         RDIM    LDAX  B         Get nr of items
066 E630 03                 INX   B
067 E631 B7         RDM05   ORA   A
068 E632 C8                 RZ              Abort if no items or ready
069 E633 3D                 DCR   A         Item count
070 E634 F5                 PUSH  PSW       Preserve count
071 E635 CD5AE9             CALL  :E95A     Get pnr to array in HL;
072                                         type in A
073 E638 E5                 PUSH  H         Preserve pntr
074 E639 CD5CCE             CALL  :CE5C     Erase array if existing
075 E63C E630               ANI   :30       Get type only
076 E63E 110400             LXI   D,:0004   Length array element if
077                                         FPT/INT
078 E641 FE20               CPI   :20       String type?
079 E643 C248E6             JNZ   :E648     Jump if not
080 E646 1E02               MVI   E,:02     Length STR array element
081 E648 0A         RDM10   LDAX  B         Get number of elements
082 E649 03                 INX   B
083 E64A 67                 MOV   H,A       ) In H and in L
084 E64B 6F                 MOV   L,A       )
085 E64C EB                 XCHG
086
087                 * Calculate total required length:
088
089 E64D CD4FE7      RDM20   CALL  :E74F     Get length next dimension
090 E650 F5                 PUSH  PSW       Remember it
091 E651 3C                 INR   A         Length +1
092 E652 CDF0ED             CALL  :EDF0     Calc reqd space
093 E655 DA15DA             JC    :DA15     Run error 'NUMBER OUT
094                                         OF RANGE' if total space
095                                         > 64K
096 E658 15                 DCR   D         nr of elements -1
097 E659 C24DE6             JNZ   :E64D     Next element if not ready
098
099                 * Find space in heap:
100
101 E65C 25                 DCR   H
102 E65D 24                 INR   H
103 E65E FA15DA             JM    :DA15     Run error 'NUMBER OUT OF
104                                         RANGE' if > 32K reserved
105 E661 19                 DAD   D
106 E662 23                 INX   H         Size of space reqd in HL
107 E663 D5                 PUSH  D
108 E664 EB                 XCHG
109 E665 CD8BE6             CALL  :E68B     Get space of size needed
110 E668 D1                 POP   D
111 E669 73                 MOV   M,E       Store nr of elements
112 E66A 19                 DAD   D         Last element
113
114                 * Elements into heap:
115
116 E66B F1         RDM30   POP   PSW       Get length 1 element
117 E66C 77                 MOV   M,A       Store it in memory
118 E66D 2B                 DCX   H
119 E66E 1D                 DCR   E
120 E66F C26BE6             JNZ   :E66B     Next element to memory
121                 *
122 E672 EB                 XCHG
123 E673 E1                 POP   H         Get pntr to array
124 E674 73                 MOV   M,E       )
125 E675 23                 INX   H         ) Set pointer
```

```
126 E676 72              MOV   M,D       )
127 E677 F1              POP   PSW       Get item count in A
128 E678 C331E6          JMP   :E631     Next item
129                   *
130                   ****************************
131                   * part of RUN TALK (0EE94) *
132                   ****************************
133                   *
134                   * Entry: A: Code for osc.channel SHR 1.
135                   *
136 E67B 29           MPT47   DAD   H
137 E67C 119402       RTK50   LXI   D,:0294   Addr volumes osc. 0,1
138 E67F E601                 ANI   :01       Code SHR 1 only
139 E681 83                   ADD   E
140 E682 5F                   MOV   E,A       DE=#0294 for osc.0,1;
141                                           DE=#0295 for osc.2,N
142 E683 7C                   MOV   A,H       Get mask
143 E684 2F                   CMA             Complement it
144 E685 EB                   XCHG            Mask + vol in DE, addr
145                                           POROM/POR1M in HL
146 E686 A6                   ANA   M         Part to be preserved from
147                                           old POROM/POR1M
148 E687 B3                   ORA   E         Add new volume
149 E688 C340EA               JMP   :EA40     Continu
150                   *
151                   ***********************
152                   * REQUEST HEAP SPACE *
153                   ***********************
154                   *
155                   * Part of Run 'DIM' (0E665).
156                   * Requests space from Heap and fills it with
157                   * zeroes.
158                   *
159                   * Entry: DE: Size needed.
160                   * Exit:  HL: Points to data area (after length
161                   *            bytes).
162                   *        AFDE corrupted.
163                   *
164 E68B 15           ZHREQ   DCR   D
165 E68C 14                   INR   D
166 E68D FA15DA               JM    :DA15     Run error 'NUMBER OUT OF
167                                           RANGE' if >32K reqd
168 E690 CDC5D1               CALL  :D1C5     Run Heap request
169 E693 23                   INX   H
170 E694 23                   INX   H         HL pnts after length byte
171 E695 E5                   PUSH  H
172 E696 EB                   XCHG            Start data area in DE
173 E697 19                   DAD   D         End area in HL
174 E698 AF                   XRA   A
175 E699 CD7CDE               CALL  :DE7C     Load bank with '0'
176 E69C E1                   POP   H
177 E69D C9                   RET
178                   *
179                   ********************
180                   * RUN basiccmd UT *
181                   ********************
182                   *
183                   * Valid as direct command only.
184                   *
185 E69E AF           RUT     XRA   A
186 E69F 32B902               STA   :02B9     Enable complete keyb scan
187 E6A2 CF                   RST   1         Go to utility
```



```
188 E6A3 09                   DATA  :09
189                   *
190                   ************************
191                   * RUN basiccmd CALLM *
192                   ************************
193                   *
194 E6A4 21B3E6       RCALM   LXI   H,:E6B3   Returnaddr from Utility
195 E6A7 E5                   PUSH  H         on stack
196 E6A8 CDF8E6               CALL  :E6F8     Get UT addr in HL
197 E6AB E5                   PUSH  H         UT addr on stack
198 E6AC 0A                   LDAX  B         Get next expr
199 E6AD FEFF                 CPI   :FF       End marker?
200 E6AF C263E9               JNZ   :E963     If not: Get varptr of given
201                                           variable in HL, T/L in A
202 E6B2 03                   INX   B
203 E6B3 B7           RCM10   ORA   A         No special action
204 E6B4 C9                   RET             On entry: Goto UT addr
205                                           On exit: Back to Basic
206                                           monitor
207                   *
208                   ************************
209                   * RUN basiccmd CLEAR *
210                   ************************
211                   *
212 E6B5 CD7FD8       RCLEAR  CALL  :D87F     Get space reqd in HL
213                                           (CY=1 if > 32K)
214 E6B8 CDBBCE               CALL  :CEBB     Must be >=4 bytes, else run
215                                           error 'NUMBER OUT OF RANGE'
216 E6BB EB                   XCHG
217 E6BC 229D02               SHLD  :029D     Set Heap size
218 E6BF 2A9F02               LHLD  :029F     Get startaddr. textbuf
219 E6C2 E5                   PUSH  H
220 E6C3 CD23CB               CALL  :CB23     Empty Heap + symtab
221 E6C6 D5                   PUSH  D
222 E6C7 CD95D1               CALL  :D195     Set Heap to all available
223 E6CA E1                   POP   H
224 E6CB C314D2               JMP   :D214     Continu
225                   *
226                   ******************************
227                   * Run basiccmds TRON - TROFF *
228                   ******************************
229                   *
230                   * Sets or resets the trace flag.
231                   *
232                   * RTRON: Set trace flag.
233                   * RTROF: Reset trace flag.
234                   *
235                   * Entry: none.
236                   * Exit:  Z=1: Flag reset.
237                   *        Z=0: Flag set.
238                   *
239 E6CE 3EFF           RTRON   MVI   A,:FF
240 E6D0 321501       RTR10   STA   :0115     Set trace flag
241 E6D3 B7                   ORA   A         No special action
242 E6D4 C9                   RET
243                   *
244 E6D5 AF           RTROF   XRA   A
245 E6D6 C3D0E6               JMP   :E6D0     Reset trace flag
246                   *
247                   ********************
248                   * READ LINENUMBER *
249                   ********************
```

```
                        *
                        * Entry: BC: Points to linenumber.
                        * Exit:  Z=0: Linenumber in HL.
                        *        Z=1: HL preserved.
                        *        BC updated, DE preserved, AF corrupted.
                        *
E6D9 E5         RLN     PUSH  H
E6DA 0A                 LDAX  B
E6DB 03                 INX   B
E6DC 67                 MOV   H,A       )
E6DD 0A                 LDAX  B         ) Get linenr in HL
E6DE 03                 INX   B         )
E6DF 6F                 MOV   L,A       )
E6E0 B4                 ORA   H
E6E1 CAE5E6             JZ    :E6E5     Abort if linenr is 0
E6E4 E3                 XTHL            Linenr on stack
E6E5 E1         RLN10   POP   H         Old HL or linenr in HL
E6E6 C9                 RET
                        *
                        ******************************************
                        * READ LINENUMBER AND FIND IT IN TEXTBUFFER *
                        ******************************************
                        *
                        * Entry: BC: Points to linenumber.
                        * Exit:  BC updated, DE preserved, AF corrupted
                        *        (RLNF) or preserved (RLNFI).
                        *        HL: Points to 1st linenr >= reqd. number.
                        *        CY=1: Linenumber found.
                        *        CY=0: Not found.
                        *
E6E7 CDD9E6     RLNF    CALL  :E6D9     Get linenr in HL
E6EA C3F6CA             JMP   :CAF6     Find it in textbuffer

                        * Idem as RLNF, but with error reporting:

E6ED F5         RLNFI   PUSH  PSW
E6EE CDE7E6             CALL  :E6E7     Read linenr and find it
E6F1 3E04               MVI   A,:04
E6F3 D2F5D9             JNC   :D9F5     Run error 'UNDEFINED NUMBER'
                                        if not found
E6F6 F1                 POP   PSW
E6F7 C9                 RET
                        *
                        ******************************************
                        * RUN A INT EXPRESSION WITH 2-BYTE RESULT *
                        ******************************************
                        *
                        * Evaluates a 16-bit INT expression (in range 0-
                        * FFFF). The result is in HL.
                        *
                        * Entry: BC: Points to expression.
                        * Exit:  HL: Result.
                        *        BC updated, AFDE corrupted.
                        *
E6F8 F5         REXI2   PUSH  PSW
E6F9 D5                 PUSH  D
E6FA CD19E8             CALL  :E819     Eval arguments in num expr
                                        Result in MACC or in WORKE
E6FD 7C                 MOV   A,H
E6FE B5                 ORA   L
E6FF CA10E7             JZ    :E710     Jump if result in MACC
```



```
                        * If result in WORKE:

E702 7E                 MOV   A,M       )
E703 23                 INX   H         ) Check if > 2 bytes
E704 B6                 ORA   M         )
E705 C215DA             JNZ   :DA15     Then run error 'NUMBER OUT
                                        OF RANGE'
E708 23                 INX   H
E709 7E                 MOV   A,M       )
E70A 23                 INX   H         ) Get result in HL
E70B 6E                 MOV   L,M       )
E70C 67                 MOV   H,A       )
E70D D1                 POP   D
E70E F1                 POP   PSW
E70F C9                 RET

                        * If result in MACC:

E710 C5         RX210   PUSH  B
E711 E7                 RST   4         Copy MACC to reg A,B,C,D
E712 15                 DATA  :15
E713 B0                 ORA   B         Check if > 2 bytes
E714 C215DA             JNZ   :DA15     Then run error 'NUMBER OUT
                                        OF RANGE'
E717 6A                 MOV   L,D       ) Result in HL
E718 61                 MOV   H,C       )
E719 C1                 POP   B
E71A D1                 POP   D
E71B F1                 POP   PSW
E71C C9                 RET
                        *
                        ********************************
                        * RUN A 1-BYTE INT EXPRESSION *
                        ********************************
                        *
                        * Evaluates a 8-bit INT expression (range 0-
                        * FF). Result in A.
                        *
                        * Entry: BC: Points to expression.
                        * Exit:  A:  Result.
                        *        BC updated, DEHL preserved.
                        *
E71D D5         REXI1   PUSH  D
E71E E5                 PUSH  H
E71F CD19E8             CALL  :E819     Eval arguments in num expr
                                        Result in MACC or WORKE
E722 7C                 MOV   A,H
E723 B5                 ORA   L
E724 CA36E7             JZ    :E736     If HL=0: Get result frm MACC

                        * Result in WORKE:

E727 7E                 MOV   A,M       )
E728 23                 INX   H         )
E729 B6                 ORA   M         ) Check if > 1 byte
E72A 23                 INX   H         )
E72B B6                 ORA   M         )
E72C C215DA             JNZ   :DA15     Then run error 'NUMBER OUT
                                        OF RANGE'
E72F 23                 INX   H
E730 7E                 MOV   A,M       Get result in A
E731 E1                 POP   H
```

```
E732 D1              POP   D
E733 C9              RET

                   * If result in MACC (also entry from REXF1):

E734 D5            RX110   PUSH  D
E735 E5                    PUSH  H
E736 E1            RX120   POP   H
E737 C5                    PUSH  B
E738 E7                    RST   4          Copy MACC to reg A,B,C,D
E739 15                    DATA  :15
E73A B0                    ORA   B          ) Check if > 1 byte
E73B B1                    ORA   C          )
E73C C215DA                JNZ   :DA15      Then run error 'NUMBER OUT
                                            OF RANGE'
E73F 7A                    MOV   A,D        Get result in A
E740 C1                    POP   B
E741 D1                    POP   D
E742 C9                    RET
                   *
                   ************************************************
                   * RUN 1-BYTE INT EXPRESSION WITH LIMITED RANGE *
                   ************************************************
                   *
                   * Entry: BC: Points to expression.
                   *        A:  Range of arguments (<=FE).
                   * Exit:  BC updated, DEHL preserved, F corrupted.
                   *        A:  Result.
                   *
E743 D5            REXIL   PUSH  D
E744 57                    MOV   D,A        Argument range in D
E745 CD1DE7                CALL  :E71D      Get value of argument in A
E748 14                    INR   D
E749 BA                    CMP   D          Out of range ? Then run
E74A D215DA                JNC   :DA15      error 'NUMBER OUT OF RANGE'
E74D D1                    POP   D
E74E C9                    RET
                   *
                   ********************************************
                   * CHECK VARIABLE TYPE AND GET ITS INT VALUE *
                   ********************************************
                   *
                   * Entry: BC: Points to expression.
                   * Exit:  Error: If string type.
                   *        If OK: Value in A (FPT: converted to INT).
                   *               BC updated, DEHL preserved.
                   *
E74F 0A            REX1    LDAX  B          Get var. type byte
E750 03                    INX   B
E751 FE20                  CPI   :20        String type?
E753 CA1ADA                JZ    :DA1A      Then run error 'TYPE
                                            MISMATCH'
E756 FE10                  CPI   :10        INT type?
E758 CA1DE7                JZ    :E71D      Then get value in A

                   * If FPT:

E75B CD08E8        REXF1   CALL  :E808      Get value in MACC
E75E E7                    RST   4          Change it to INT
E75F 48                    DATA  :48
E760 C334E7                JMP   :E734      Get value in A
                   *
```



```
                   *====================================*
                   * RUN EXPRESSIONS WITH OPERATOR PREFIX *
                   *====================================*
                   *
                   * #E763-E8ED evaluate logical, FPT, INT or STR
                   * expressions in 'operator prefix' format.
                   *
                   * Register allocation during operation:
                   *    INT/FPT: D=0:   MACC empty.
                   *             E:     Operator.
                   *             HL=0:  Result in MACC.
                   *             HL<>0: HL points to result.
                   *    STR:     HL:    Points to string.
                   *             E:     Type of string (constant [0],
                   *                    variable [1], temporary [2]).
                   *
                   **********************************
                   * EVALUATE A LOGICAL EXPRESSION *
                   **********************************
                   *
E763 1600          REXPL   MVI   D,:00      MACC free
E765 0A                    LDAX  B          Get byte
E766 E660                  ANI   :60
E768 FE40                  CPI   :40        String ?
E76A CABDE7                JZ    :E7BD      Then jump
E76D 0A                    LDAX  B          Get byte
E76E E61F                  ANI   :1F
E770 FE18                  CPI   :18        Relational operator?
E772 DA50E8                JC    :E850      If not: eval expr which
                                            begins with num operator
                           INX   B
E775 03                    CPI   :1A        Bracket?
E776 FE1A                  JZ    :E763      Then ignore it
E778 CA63E7

                   * Logical AND or OR:

                           PUSH  PSW        Preserve type of operation
E77B F5                    CALL  :E763      Get 1st operand
E77C CD63E7                PUSH  PSW        Preserve it
E77F F5                    CALL  :E763      Get 2nd operand
E780 CD63E7                POP   D          1st operand in D
E783 D1                    PUSH  PSW        Preserve 2nd operand
E784 F5                    ANA   D          AND operation
E785 A2                    MOV   E,A        Result in E
E786 5F                    POP   PSW        2nd operand in A
E787 F1                    ORA   D          OR operation
E788 B2                    MOV   D,A        Result in D
E789 57                    POP   PSW        Type of operation in F
E78A F1                    MOV   A,D        Result OR in A
E78B 7A                    JPE   :E790      Quit if OR
E78C EA90E7                MOV   A,E        Result AND in A
E78F 7B            RXL10   RET
E790 C9                    *
                   ******************************
                   * EVALUATE STRING EXPRESSION *
                   ******************************
                   *
                   * This routine returns temporary strings before
                   * they are really free.
                   * The heap is cleared if it is a temporary string.
                   *
                   * Entry: BC: Points to expression.
```

```
498                     * Exit:  BC updated, AFD corrupted.
499                     *        HL: Points to string.
500                     *         E:  Status.
501                     *
502 E791 CD9DE7          REXSR   CALL   :E79D      Evaluate string expr
503 E794 7B                    MOV    A,E        Get status
504 E795 FE02                  CPI    :02        Temporary ?
505 E797 E5                    PUSH   H
506 E798 CC87D1                CZ     :D187      Then clear heap entry
507 E79B E1                    POP    H
508 E79C C9                    RET
509                     *
510                     *
511                     *
512 E79D                       END
```

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

MPT47   E67B    R4C10   E620    R4COL   E61C    RCALM   E6A4
RCLEAR  E6B5    RCM10   E6B3    RCOLG   E615    RCOLT   E60E
RDIM    E62F    RDM05   E631    RDM10   E648    RDM20   E64D
RDM30   E66B    REX1    E74F    REXF1   E75B    REXI1   E71D
REXI2   E6F8    REXIL   E743    REXPL   E763    REXSR   E791
RLN     E6D9    RLN10   E6E5    RLNF    E6E7    RLNFI   E6ED
RTK50   E67C    RTR10   E6D0    RTROF   E6D5    RTRON   E6CE
RUT     E69E    RX110   E734    RX120   E736    RX210   E710
RXL10   E790    SCRER   E602    ZHREQ   E68B
```



```
002                             ORG    :E79D
003                     *
004                     *
005                     *
006                     ********************************************
007                     * EVALUATE ARGUMENTS IN STRING EXPRESSION *
008                     ********************************************
009                     *
010                     * Only '+' or compare with logical result is
011                     * allowed. The right-hand side of a string expres-
012                     * sion is evaluated. If it is not status 02, it is
013                     * moved into the Heap. The stringpointer is saved
014                     * at the varptr location.
015                     * If the variable had already an old value on the
016                     * heap, it is cleared, see further exit conditions.
017                     *
018                     * Entry: (BC): 1.....  Expr. begins with operator.
019                     *              01....  Variable reference.
020                     *              001...  Function call.
021                     *              else    Constant.
022                     * Exit:  BC updated, DEHL corrupted.
023                     *         A: Type (#20).
024                     *
025 E79D 0A             REXPS   LDAX   B          get 1st byte
026 E79E 07                     RLC
027 E79F DABDE7                 JC     :E7BD      Jump if 1st byte is operator
028 E7A2 07                     RLC
029 E7A3 DAB3E7                 JC     :E7B3      Jump if string variable
030 E7A6 07                     RLC
031 E7A7 DAD9E9                 JC     :E9D9      Jump if string function
032
033                     * If string constant:
034
035 E7AA 1E00                   MVI    E,:00      Status: constant
036 E7AC 03                     INX    B
037 E7AD 0A                     LDAX   B          Get string length
038 E7AE 60                     MOV    H,B        ) Stringpntr in HL
039 E7AF 69                     MOV    L,C        )
040 E7B0 C390E1                 JMP    :E190      Abort with BC pnts after STR
041
042                     * If string variable:
043
044 E7B3 CD63E9         RXS10   CALL   :E963      Get varptr in HL, T/L in A
045 E7B6 5E                     MOV    E,M        )
046 E7B7 23                     INX    H          ) Stringaddr in DE
047 E7B8 56                     MOV    D,M        )
048 E7B9 EB                     XCHG              and in HL
049 E7BA 1E01                   MVI    E,:01      Status: variable
050 E7BC C9                     RET
051
052                     * If string operation:
053
054 E7BD 0A             ROSTR   LDAX   B          Get 1st byte
055 E7BE 03                     INX    B
056 E7BF F5                     PUSH   PSW
057 E7C0 D5                     PUSH   D
058 E7C1 CD9DE7                 CALL   :E79D      Evaluate string expression
059 E7C4 F1                     POP    PSW
060 E7C5 57                     MOV    D,A
061 E7C6 F1                     POP    PSW
062 E7C7 E5                     PUSH   H          Remember it
063 E7C8 53                     MOV    D,E        Type in D
```

```
064 E7C9 F5              PUSH  PSW
065 E7CA D5              PUSH  D
066 E7CB CD9DE7          CALL  :E79D     Eval 2nd string expr
067 E7CE F1              POP   PSW
068 E7CF 57              MOV   D,A
069 E7D0 F1              POP   PSW
070 E7D1 C5              PUSH  B         Save it
071 E7D2 44              MOV   B,H
072 E7D3 4D              MOV   C,L
073 E7D4 E1              POP   H
074 E7D5 E3              XTHL            Save program pointer
075 E7D6 C5              PUSH  B         )
076 E7D7 42              MOV   B,D       )
077 E7D8 4B              MOV   C,E       ) Re-arrange registers
078 E7D9 EB              XCHG            )
079 E7DA E1              POP   H         )
080 E7DB FECO            CPI   :CO       Operator is '+'?
081 E7DD CAEBE7          JZ    :E7EB     Then append 2 strings
082
083                  * If string compare:
084
085 E7E0 CD21D1          CALL  :D121     Compare 2 strings
086 E7E3 CDF8E7          CALL  :E7F8     Returns operands if temp
087 E7E6 C1              POP   B         restore
088 E7E7 5F              MOV   E,A       Opcode in E
089 E7E8 C333E9          JMP   :E933     Evaluate the compare
090
091                  * If operator is '+':
092
093 E7EB E5        ROS10 PUSH  H
094 E7EC CD06D1          CALL  :D106     Make 1 string out of 2
095 E7EF E3              XTHL            Save pntr to result/store
096                                      pntr to operand
097 E7F0 CDF8E7          CALL  :E7F8     Clean up heap
098 E7F3 E1              POP   H         Pntr to result in HL
099 E7F4 C1              POP   B         Program pntr in BC
100 E7F5 1E02            MVI   E,:02     Status: temporary
101 E7F7 C9              RET
102                  *
103                  * CLEAR UP HEAP AFTER STRING OPERATION:
104                  *
105                  * Entry: B,C: Code for 1st resp. 2nd operand.
106                  *             (0=const, 1=var, 2=temp).
107                  *        DE:  Points to 1st operand.
108                  *        HL:  Points to 2nd operand.
109                  * Exit:  DEHL corrupted, AFBC preserved.
110                  *
111 E7F8 F5        DROPS PUSH  PSW
112 E7F9 79              MOV   A,C       Get code 2nd operand
113 E7FA FE02            CPI   :02       Temporary?
114 E7FC CC87D1          CZ    :D187     Then clear string in heap
115 E7FF EB              XCHG
116 E800 78              MOV   A,B       Get code 1st operand
117 E801 FE02            CPI   :02       Temporary?
118 E803 CC87D1          CZ    :D187     Then clear string in heap
119 E806 F1              POP   PSW
120 E807 C9              RET
121                  *
122                  **********************************
123                  * EVALUATE A NUMERIC EXPRESSION *
124                  **********************************
125                  *
```



```
126                  * Entry: BC: Points to a numeric function argument
127                  *             or a numeric expression in program.
128                  * Exit:  BC updated, AFDEHL preserved.
129                  *        Result in MACC.
130                  *
131 E808 F5        REXNA PUSH  PSW
132 E809 D5              PUSH  D
133 E80A E5              PUSH  H
134 E80B CD19E8          CALL  :E819     Eval arguments in num expr
135 E80E 7C              MOV   A,H
136 E80F B5              ORA   L
137 E810 CA15E8          JZ    :E815     Abort if result in MACC
138 E813 E7              RST   4         Else: copy operand to MACC
139 E814 0C              DATA  :0C
140 E815 E1        LOE146 POP   H
141 E816 D1              POP   D
142 E817 F1              POP   PSW
143 E818 C9              RET
144                  *
145                  **********************************************
146                  * EVALUATE ARGUMENTS IN NUMERIC EXPRESSION *
147                  **********************************************
148                  *
149                  * Checks for constants, functions, variables and
150                  * operators. The right-hand side of the expression
151                  * is therefore evaluated. The value of the variable
152                  * is stored at its varptr location.
153                  *
154                  * Entry: BC:   Points to expression in program.
155                  *        (BC): 1.... Expr begins with operator.
156                  *              01... Variable reference.
157                  *              001.. Function calll.
158                  *              Else  Constant.
159                  *        D<>0: MACC must be preserved.
160                  *
161 E819 1600      REXPN MVI   D,:00     Set MACC free
162
163                  * Called by lower levels:
164
165 E81B 0A        RXN10 LDAX  B         Get 1st byte
166 E81C 07              RLC
167 E81D DA50E8          JC    :E850     Jump if expr starts with
168                                      operator
169 E820 07              RLC
170 E821 DA6BE9          JC    :E96B     Jump if var reference
171 E824 07              RLC
172 E825 DA30E8          JC    :E830     Jump if function call
173
174                  * If numeric constant:
175
176 E828 03              INX   B         Past flag byte
177 E829 60              MOV   H,B       ) HL pnts to constant
178 E82A 69              MOV   L,C       )
179 E82B 03              INX   B
180 E82C 03              INX   B
181 E82D 03              INX   B
182 E82E 03              INX   B         Program pntr pnts beyond
183 E82F C9              RET
184
185                  * If numeric function:
186
187 E830 D5        RFUNN PUSH  D
```

```
188 E831 7A              MOV   A,D
189 E832 B7              ORA   A
190 E833 CA46E8          JZ    :E846      Jump if MACC free
191 E836 CD18C0          CALL  :C018      Save MACC on stack
192 E839 CDD9E9          CALL  :E9D9      Evaluate function call
193                                       result in MACC
194 E83C 212901          LXI   H,:0129    Addr WORKE
195 E83F E7              RST   4          Copy result to WORKE
196 E840 0F              DATA  :0F
197 E841 CD1BC0          CALL  :C01B      Restore MACC from stack
198 E844 D1              POP   D
199 E845 C9              RET
200 E846 CDD9E9    RFN10 CALL  :E9D9      Evaluate function call,
201                                       result in MACC
202 E849 D1              POP   D
203 E84A 16FF            MVI   D,:FF      Set MACC to be preserved
204 E84C 210000          LXI   H,:00      Flag 'result in MACC'
205 E84F C9              RET
206
207                * If expr begins with numeric operator:
208
209 E850 0A        RONUM LDAX  B          Get operator
210 E851 E67F            ANI   :7F        Clip operator bit
211 E853 03              INX   B
212 E854 FE1A            CPI   :1A        Bracket?
213 E856 CA1BE8          JZ    :E81B      Then ignore it
214 E859 15              DCR   D
215 E85A 14              INR   D          Check if D<>0
216 E85B C418C0          CNZ   :C018      Then save MACC on stack
217 E85E D5              PUSH  D
218 E85F 5F              MOV   E,A        Opcode in E
219 E860 21DCE8          LXI   H,:E8DC
220 E863 E5              PUSH  H          Returnaddr on stack
221 E864 CD19E8          CALL  :E819      Get 1st operand
222 E867 7B              MOV   A,E        Opcode in A
223 E868 E61F            ANI   :1F
224 E86A FE1C            CPI   :1C
225 E86C D29FE8          JNC   :E89F      Jump if unitary operation
226
227                * If boolean operator:
228
229 E86F E5              PUSH  H          Save pntr to 1st operand
230 E870 CD1BE8          CALL  :E81B      Get 2nd operand
231 E873 7C              MOV   A,H
232 E874 B5              ORA   L
233 E875 C27DE8          JNZ   :E87D      Jump if HL pnts to WORKE
234 E878 212901          LXI   H,:0129    Addr WORKE
235 E87B E7              RST   4          Copy 2nd operand to WORKE
236 E87C 0F              DATA  :0F
237 E87D E3        RON10 XTHL
238 E87E 7C              MOV   A,H
239 E87F B5              ORA   L
240 E880 CA85E8          JZ    :E885      Jump if 1st operand in MACC
241 E883 E7              RST   4          Else copy it from WORKE
242 E884 0C              DATA  :0C        to MACC
243 E885 7B        LOE153 MOV   A,E        Get opcode
244 E886 E61F            ANI   :1F
245 E888 FE10            CPI   :10
246 E88A D2C9E8          JNC   :E8C9      If relational operation
247
248                * If arithmetic operation:
```



```
250 E88D BB              CMP   E
251 E88E 21FDE8          LXI   H,:E8FD    Addr table INT routines
252 E891 C297E8          JNZ   :E897      Jump if INT
253 E894 21EEE8          LXI   H,:E8EE    Addr table FPT routines
254 E897 1600      RON20 MVI   D,:00      Set MACC free
255 E899 5F              MOV   E,A        Opcode in E
256 E89A 19              DAD   D
257 E89B 19              DAD   D
258 E89C 19              DAD   D          Find routine in table
259 E89D E3              XTHL             Addr routine on stack; pntr
260                                       to 2nd operand in HL
261 E89E C9              RET              Perform routine
262                *
263                * If an unitary operation:
264                *
265                * Entry: HL: Points to operand (0 if in MACC).
266                *         E: Full opcode.
267                *         A: Lower 5 bits opcode.
268                *         Returnaddr on stack (E8DC).
269                * Exit:  Result in MACC.
270                *         ABCDEHL preserved
271                *
272 E89F F5        RON40 PUSH  PSW
273 E8A0 7C              MOV   A,H
274 E8A1 B5              ORA   L
275 E8A2 CAA7E8          JZ    :E8A7      If operand in MACC
276 E8A5 E7              RST   4          Else: operand in MACC
277 E8A6 0C              DATA  :0C
278 E8A7 F1        LOE156 POP  PSW
279 E8A8 C8              RZ               Ready if unitary '+'
280 E8A9 BB              CMP   E          Bits 6,7 opcode 0?
281 E8AA CABEE8          JZ    :E8BE      Then change MACC to INT
282 E8AD FE1E            CPI   :1E        INOT?
283 E8AF DABBE8          JC    :E8BB      Then change sign MACC (INT)
284 E8B2 C2B8E8          JNZ   :E8B8      Then convert MACC to FPT
285 E8B5 E7              RST   4          Perform INOT
286 E8B6 6C              DATA  :6C
287 E8B7 C9              RET
288                *
289 E8B8 E7        LOE157 RST  4          Convert MACC to FPT
290 E8B9 4B              DATA  :4B
291 E8BA C9              RET
292                *
293 E8BB E7        RON44 RST   4          Change sign MACC (INT)
294 E8BC 60              DATA  :60
295 E8BD C9              RET
296                *
297 E8BE FE1D      RON45 CPI   :1D
298 E8C0 CAC6E8          JZ    :E8C6      Then change sign MACC (FPT)
299 E8C3 E7              RST   4          Convert MACC to INT
300 E8C4 48              DATA  :48
301 E8C5 C9              RET
302                *
303 E8C6 E7        RON49 RST   4          Change sign MACC (FPT)
304 E8C7 1B              DATA  :1B
305 E8C8 C9              RET
306                *
307                * If relational numeric operation:
308                *
309                * Entry: 1st operand in MACC, 2nd operand on stack.
310                *         E: Full opcode.
311                *         A: Lowest 5 bits opcode.
```

```
                        * Exit:  BC preserved, DEHL corrupted.
                        *
E8C9 E1         RON50   POP   H         Get pntr 2nd operand
E8CA BB                 CMP   E
E8CB CAD6E8             JZ    :E8D6     Jump if FPT
E8CE CD15C0             CALL  :C015     Compare 2 INT numbers
E8D1 E1         RON55   POP   H         Kill returnaddr
E8D2 E1                 POP   H         Kill saved DE
E8D3 C333E9             JMP   :E933     Return logical result
E8D6 CD0CC0     RON60   CALL  :C00C     Compare 2 FPT numbers
E8D9 C3D1E8             JMP   :E8D1
                        *
                        * MOVE OPERAND:
                        *
                        * REX.. routines return here after operation.
                        * Moves operand to proper location after computing.
                        *
                        * Entry: DE and returnaddress on stack.
                        * Exit:  ABC preserved.
                        *
E8DC D1         RON30   POP   D
E8DD 15                 DCR   D
E8DE 14                 INR   D         Check D=0 (MACC free)
E8DF 16FF               MVI   D,:FF
E8E1 210000             LXI   H,:0000   Flag 'result in MACC'
E8E4 C8                 RZ              Abort if operand in MACC
E8E5 212901             LXI   H,:0129   Addr WORKE
E8E8 E7                 RST   4         Copy MACC to WORKE
E8E9 0F                 DATA  :0F
E8EA CD1BC0             CALL  :C01B     Restore old MACC from stack
E8ED C9                 RET
                        *
                        * TABLE OF JUMPS TO INT/FPT OPERATOR ROUTINES:
                        *
E8EE E7         ROFTAB  RST   4
E8EF 00                 DATA  :00       MFADD; +
E8F0 C9                 RET
                        *
E8F1 E7                 RST   4
E8F2 03                 DATA  :03       MFSUB; -
E8F3 C9                 RET
                        *
E8F4 E7                 RST   4
E8F5 09                 DATA  :09       MFDIV; /
E8F6 C9                 RET
                        *
E8F7 E7                 RST   4
E8F8 06                 DATA  :06       MFMUL; *
E8F9 C9                 RET
                        *
E8FA E7                 RST   4
E8FB 24                 DATA  :24       MPWR ; ^
E8FC C9                 RET
                        *
E8FD E7         ROITAB  RST   4
E8FE 4E                 DATA  :4E       MIADD; +
E8FF C9                 RET
                        *
E900 E7                 RST   4
E901 51                 DATA  :51       MISUB; -
E902 C9                 RET
                        *
```



```
E903 E7                 RST   4
E904 57                 DATA  :57       MIDIV; /
E905 C9                 RET
                        *
E906 E7                 RST   4
E907 54                 DATA  :54       MIMUL; *
E908 C9                 RET
                        *
E909 00                 DATA  :00
E90A 00                 DATA  :00
E90B 00                 DATA  :00
E90C 00                 DATA  :00
E90D 00                 DATA  :00
E90E 00                 DATA  :00
E90F 00                 DATA  :00
E910 00                 DATA  :00
E911 00                 DATA  :00
E912 00                 DATA  :00
E913 00                 DATA  :00
E914 00                 DATA  :00
E915 00                 DATA  :00
E916 00                 DATA  :00
E917 00                 DATA  :00
                        *
E918 E7                 RST   4         MIAND
E919 63                 DATA  :63
E91A C9                 RET
                        *
E91B E7                 RST   4         MIOR
E91C 66                 DATA  :66
E91D C9                 RET
                        *
E91E 00                 DATA  :00
E91F 00                 DATA  :00
E920 00                 DATA  :00
                        *
E921 E7                 RST   4         MIXOR
E922 69                 DATA  :69
E923 C9                 RET
                        *
E924 E7                 RST   4         MSHL
E925 6F                 DATA  :6F
E926 C9                 RET
                        *
E927 E7                 RST   4         MSHR
E928 72                 DATA  :72
E929 C9                 RET
                        *
E92A E7                 RST   4         MIREM
E92B 5A                 DATA  :5A
E92C C9                 RET
                        *
                        *
                        *
E92D                    END
```

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

DROPS  E7F8   L0E146 E815   L0E153 E885   L0E156 E8A7
L0E157 E8B8   REXNA  E808   REXPN  E819   REXPS  E79D
```

RFN10	E846	RFUNN	E830	ROFTAB	E8EE	ROITAB	E8FD
RON10	E87D	RON20	E897	RON30	E8DC	RON40	E89F
RON44	E8BB	RON45	E8BE	RON49	E8C6	RON50	E8C9
RON55	E8D1	RON60	E8D6	RONUM	E850	ROS10	E7EB
ROSTR	E7BD	RXN10	E81B	RXS10	E7B3		


```
002                     ORG   :E92D
003                *
004                *
005                *
006                *************************
007                * LENGTH OF BLOCK IN BC *
008                *************************
009                *
010                * Part of Run 'CLEAR' (D214).
011                *
012 E92D CD1ADE    MPT45  CALL  :DE1A      Calc. length of block
013 E930 44               MOV   B,H       )
014 E931 4D               MOV   C,L       ) Length in BC
015 E932 C9               RET
016                *
017                ***********************
018                * EVALUATE A COMPARE *
019                ***********************
020                *
021                * Decodes flags and opcode to a truthtable.
022                * Following a XFCOMP or a XICOMP by a jump to
023                * E933 leaves FF (true) or 00 (false) in A as
024                * result.
025                *
026                * Entry: E: Opcode.
027                *        F: Flags.
028                * Exit:  BCDEF preserved, HL corrupted.
029                *        A: Truth value.
030                *
031 E933 F5        ROREL  PUSH  PSW       Save flags
032 E934 7B               MOV   A,E       )
033 E935 E60F             ANI   :0F       ) Calc offset
034 E937 87               ADD   A         )
035 E938 87               ADD   A         )
036 E939 2143E9           LXI   H,:E943   Base addr
037 E93C CD30DE           CALL  :DE30     Add offset to base
038 E93F F1               POP   PSW       Restore flags
039 E940 3EFF             MVI   A,:FF     Init truth value
040 E942 E9               PCHL            Goto routine
041                *
042 E943 F0        ROGEQ  RP              FF if MACC >= M
043 E944 C358E9           JMP   :E958     (S=0)
044                *
045 E947 FA58E9    ROGT   JM    :E958     FF if MACC > M
046 E94A 00               NOP             (S=0 and Z=0)
047                *
048 E94B C0        RONEQ  RNZ             FF if MACC <> M
049 E94C C358E9           JMP   :E958     (Z=0)
050                *
051 E94F C8        ROLEQ  RZ              FF if MACC <= M
052 E950 00               NOP             (S=1 or Z=1)
053 E951 00               NOP
054 E952 00               NOP
055                *
056 E953 F8        ROLT   RM              FF if MACC < M
057 E954 C358E9           JMP   :E958     (S=1)
058                *
059 E957 C8        ROEQ   RZ              FF if MACC = M
060                                       (Z=1)
061                *
062 E958 2F        RFALSE CMA             00 if condition false
063 E959 C9               RET
```

```
                        *
                        ****************************
                        * RUN A VARIABLE REFERENCE *
                        ****************************
                        *
                        * Produces a pointer to the value of a variable.
                        * The variable may be simple or subscripted and
                        * of any type. If subscripted, subscripts are
                        * evaluated and checked for range.
                        *
                        * Entry: BC: Points to encoded variable.
                        *        D:  0 if MACC free.
                        * Exit:  BC updated, DE preserved, F corrupted.
                        *        HL: Points to variable storage.
                        *        A:  Type of variable from symbol table.
                        *
E95A D5                 RARRN   PUSH  D
E95B AF                         XRA   A          Array name only
                        *
E95C 1600               RVREN   MVI   D,:00
E95E C33DD7                     JMP   :D73D      Run varptr
                        *
E961 FF                         DATA  :FF
E962 FF                         DATA  :FF
                        *
                        ************************************
                        * RUN VARPTR (ARRAY WITH ARGUMENTS) *
                        ************************************
                        *
                        * Runs a varptr with A=FF, D=0.
                        *
                        * Exit: BC updated, DE preserved, MACC corrupted.
                        *       HL: Varptr.
                        *       A:  T/L byte.
                        *
E963 D5                 RVAR    PUSH  D
E964 3EFF                       MVI   A,:FF      Set mask for arrays (not
                                                 name only)
E966 C35CE9                     JMP   :E95C      Run varptr
                        *
E969 FF                         DATA  :FF
E96A FF                         DATA  :FF
                        *
                        ************************
                        * RUN A VARIABLE POINTER *
                        ************************
                        *
                        * RVARE: Entry for arrays.
                        * RVR05: 'Normal' entry.
                        *
                        * Entry: BC: Points to var.reference in program.
                        *        A:  Mask.
                        *        D:  0 if MACC free.
                        * Exit:  HL: Varptr (for strings: stringpntr).
                        *        A:  T/L of symtab.
                        *        BC updated, DE preserved.
                        *
E96B 3EFF               RVARE   MVI   A,:FF      Not array name only
E96D F5                 RVR05   PUSH  PSW
E96E D5                         PUSH  D
E96F 0A                         LDAX  B          )
E970 03                         INX   B          )
```



```
E971 E63F                       ANI   :3F        ) Get offset in symtab
E973 57                         MOV   D,A        ) in DE
E974 0A                         LDAX  B          )
E975 03                         INX   B          )
E976 5F                         MOV   E,A        )
E977 2AA102                     LHLD  :02A1      Get start symtab
E97A 19                         DAD   D          HL pnts to actual addr
                                                 in symtab
E97B D1                         POP   D
E97C F1                         POP   PSW        Restore mask
E97D A6                         ANA   M          And with T/L byte
E97E E640                       ANI   :40        Bit 6 only
E980 7E                         MOV   A,M
E981 23                         INX   H
E982 C8                         RZ               Abort if simple variable or
                                                 array name only

                        * Compute actual position in array:

E983 15                         DCR   D
E984 14                         INR   D          Check D=0: MACC free
E985 C418C0                     CNZ   :C018      Save MACC on stack if not
E988 D5                         PUSH  D
E989 F5                         PUSH  PSW
E98A 5E                         MOV   E,M        ) Get pntr from symtab
E98B 23                         INX   H          ) in DE
E98C 56                         MOV   D,M        )
E98D 23                         INX   H
E98E 7A                         MOV   A,D        ) Check if undimensioned
E98F B3                         ORA   E          )
E990 3E0F               ERRUA   MVI   A,:0F      Then run error
E992 CAF5D9                     JZ    :D9F5      'UNDEFINED ARRAY'
E995 D5                         PUSH  D
E996 210000                     LXI   H,:0000    Init index
E999 E3                         XTHL             Pntr to array in HL
E99A 0A                         LDAX  B          Get nr of subscripts
E99B 03                         INX   B
E99C 57                         MOV   D,A        in D
E99D BE                         CMP   M          Comp. with nr of dimensions
E99E 23                         INX   H
E99F C229DA                     JNZ   :DA29      Run 'SUBSCRIPT ERROR' if
                                                 not identical
E9A2 CD4FE7             RVR10   CALL  :E74F      Get variable type in A
E9A5 BE                 RVR15   CMP   M
E9A6 DAB1E9                     JC    :E9B1      If value in A is offset
E9A9 CAB1E9                     JZ    :E9B1
E9AC 3E05                       MVI   A,:05      If subscript <0 or >FF:
E9AE C3F5D9                     JMP   :D9F5      Run 'SUBSCRIPT ERROR'

                        * Calc reference to Nth array element
                        * via (a1*(d2+1)+a2*(d3+1)+..+aN). aN
                        * is argument, dN is dimension:

E9B1 E3                 RVR20   XTHL             Restore HL=00
E9B2 CD30DE                     CALL  :DE30      Add offset to index
E9B5 15                         DCR   D          decr nr of arguments
E9B6 E3                         XTHL
E9B7 23                         INX   H
E9B8 CAC5E9                     JZ    :E9C5      If no more subscipts
E9BB 7E                         MOV   A,M        Next dimension
E9BC 3C                         INR   A
E9BD E3                         XTHL
```

```
E9BE CD8FDE          CALL  :DE8F     Multiply index by it
E9C1 E3              XTHL
E9C2 C3A2E9          JMP   :E9A2     Process next subscript
                *
E9C5 EB         RVR30  XCHG
E9C6 E1              POP   H         Get index to array from
                                     symtab
E9C7 F1              POP   PSW       Get type byte
E9C8 F5              PUSH  PSW
E9C9 E630            ANI   :30       Bits 5,6 only
E9CB FE20            CPI   :20
E9CD 29              DAD   H         Add offset to element
E9CE CAD2E9          JZ    :E9D2
E9D1 29              DAD   H
E9D2 19         RVR40  DAD   D         Abs addr of element in HL
                                     (STR: pntr to string)
E9D3 F1              POP   PSW
E9D4 D1              POP   D
E9D5 C41BC0          CNZ   :C01B     Evt retrieve MACC from stack
E9D8 C9              RET
                *
                ****************************
                * EVALUATE A FUNCTION CALL *
                ****************************
                *
                * Finds address of function routine in
                * indirection table (#E9F0) and performs the
                * function routine.
                *
                * Entry: BC: Points to function label (#20)
                *            in program line.
                * Exit:  MACC: Numeric result.
                *        BC updated, AFDEHL corrupted.
                *
E9D9 03         RFUN   INX   B         Pnts past label
E9DA 0A              LDAX  B         Get function code
E9DB 03              INX   B
E9DC 6F              MOV   L,A       Code in HL
E9DD 2600            MVI   H,:00
E9DF 11F0E9          LXI   D,:E9F0   Get startaddr. table
E9E2 29              DAD   H         Code *2
E9E3 19              DAD   D         Add offset to startaddr
E9E4 5E              MOV   E,M       lobyte addr in E
E9E5 23              INX   H
E9E6 7E              MOV   A,M       hibyte addr in A
E9E7 F680            ORI   :80       Set msb = 1
E9E9 57              MOV   D,A       hibyte in D
E9EA BE              CMP   M         Was it already E...?
E9EB D5              PUSH  D         Addr function on stack
E9EC CC08E8          CZ    :E808     Then evaluate 1st num
                                     expr, result in MACC
E9EF C9              RET             Go to function routine
                *
                ******************************
                * FUNCTION INDIRECTION TABLE *
                ******************************
                *
                * Startaddress table is E9F0. The function code
                * is given between brackets.
                * If the msb of the address is set, the first
                * argument is a numeric one.
                *
```



```
                * In the textbuffer, the Basic fuctions are
                * indicated by 20 xx, (xx is function code).
                *
E9F0 50EA       FUNIT  DBL   :EA50    (00)  ABS
E9F2 53EA              DBL   :EA53    (01)  ALOG
E9F4 CB6A              DBL   :6ACB    (02)  ASC
E9F6 D26A              DBL   :6AD2    (03)  CHR$
E9F8 B86A              DBL   :6AB8    (04)  CURY
E9FA BE6A              DBL   :6ABE    (05)  CURX
E9FC 56EA              DBL   :EA56    (06)  EXP
E9FE 59EA              DBL   :EA59    (07)  FRAC
EA00 436B              DBL   :6B43    (08)  FRE
EA02 5CEB              DBL   :EB5C    (09)  FREQ
EA04 796B              DBL   :6B79    (0A)  GETC
EA06 836A              DBL   :6A83    (0B)  HEX$
EA08 82EB              DBL   :EB82    (0C)  INP
EA0A 8DEB              DBL   :EB8D    (0D)  INT
EA0C E26A              DBL   :6AE2    (0E)  LEFT$
EA0E C46A              DBL   :6AC4    (0F)  LEN
EA10 A16B              DBL   :6BA1    (10)  VARPTR
EA12 5CEA              DBL   :EA5C    (11)  LOG
EA14 5FEA              DBL   :EA5F    (12)  LOGT
EA16 A76B              DBL   :6BA7    (13)  XMAX
EA18 AE6B              DBL   :6BAE    (14)  YMAX
EA1A 0E6B              DBL   :6B0E    (15)  MID$
EA1C C16B              DBL   :6BC1    (16)  PDL
EA1E 166C              DBL   :6C16    (17)  PEEK
EA20 1D6C              DBL   :6C1D    (18)  PI
EA22 FF6A              DBL   :6AFF    (19)  RIGHT$
EA24 27EC              DBL   :EC27    (1A)  RND
EA26 9D6C              DBL   :6C9D    (1B)  SCRN
EA28 7BEC              DBL   :EC7B    (1C)  SGN
EA2A 8C6A              DBL   :6A8C    (1D)  SPC
EA2C 62EA              DBL   :EA62    (1E)  SQR
EA2E 77EA              DBL   :EA77    (1F)  STR$
EA30 A26A              DBL   :6AA2    (20)  TAB
EA32 256B              DBL   :6B25    (21)  VAL
EA34 65EA              DBL   :EA65    (22)  SIN
EA36 68EA              DBL   :EA68    (23)  COS
EA38 6BEA              DBL   :EA6B    (24)  TAN
EA3A 6EEA              DBL   :EA6E    (25)  ASIN
EA3C 71EA              DBL   :EA71    (26)  ACOS
EA3E 74EA              DBL   :EA74    (27)  ATN
                *
                *
                *
EA40                   END

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

ERRUA  E990   FUNIT  E9F0   MPT45  E92D   RARRN  E95A
RFALSE E958   RFUN   E9D9   ROEQ   E957   ROGEQ  E943
ROGT   E947   ROLEQ  E94F   ROLT   E953   RONEQ  E94B
OREL   E933   RVAR   E963   RVARE  E96B   RVR05  E96D
RVR10  E9A2   RVR15  E9A5   RVR20  E9B1   RVR30  E9C5
RVR40  E9D2   RVREN  E95C
```

```
002                             ORG   :EA40
003                     *
004                     *
005                     *
006                     ****************************
007                     * part of RUN TALK (OE67B) *
008                     ****************************
009                     *
010                     * Sets oscillator volumes.
011                     *
012                     * Entry: A:  New volume.
013                     *        HL: Address POROM/POR1M.
014                     *
015 EA40 77             MPT48   MOV   M,A       Update POROM/POR1M
016 EA41 1170FA                 LXI   D,:FA70
017 EA44 19                     DAD   D         HL= POR0/POR1
018 EA45 77                     MOV   M,A       Update osc.volume
019 EA46 E1                     POP   H         Get parameter pntr
020 EA47 23             RTK55   INX   H         Pnts to next code
021 EA48 C367CD                 JMP   :CD67     Handle next code
022                     *
023 EA4B FF                     DATA  :FF
024 EA4C FF                     DATA  :FF
025 EA4D FF                     DATA  :FF
026 EA4E FF                     DATA  :FF
027 EA4F FF                     DATA  :FF
028                     *
029                     ************************
030                     * RUN basicfunction ABS *
031                     ************************
032                     *
033 EA50 E7             RABS    RST   4         MFABS
034 EA51 18                     DATA  :18
035 EA52 C9                     RET
036                     *
037                     ************************
038                     * RUN basicfunction ALOG *
039                     ************************
040                     *
041 EA53 E7             RALOG   RST   4         MALOG
042 EA54 30                     DATA  :30
043 EA55 C9                     RET
044                     *
045                     ************************
046                     * RUN basicfunction EXP *
047                     ************************
048                     *
049 EA56 E7             REXP    RST   4         MEXP
050 EA57 2A                     DATA  :2A
051 EA58 C9                     RET
052                     *
053                     ************************
054                     * RUN basicfunction FRAC *
055                     ************************
056                     *
057 EA59 E7             RFRAC   RST   4         MFRAC
058 EA5A 21                     DATA  :21
059 EA5B C9                     RET
060                     *
061                     ************************
062                     * RUN basicfunction LOG *
063                     ************************
```



```
064                     *
065 EA5C E7             RLOG    RST   4         MLOG
066 EA5D 27                     DATA  :27
067 EA5E C9                     RET
068                     *
069                     ************************
070                     * RUN basicfunction LOGT *
071                     ************************
072                     *
073 EA5F E7             RLOGT   RST   4         MLOGT
074 EA60 2D                     DATA  :2D
075 EA61 C9                     RET
076                     *
077                     ************************
078                     * RUN basicfunction SQR *
079                     ************************
080                     *
081 EA62 E7             RSQR    RST   4         MSQR
082 EA63 33                     DATA  :33
083 EA64 C9                     RET
084                     *
085                     ************************
086                     * RUN basicfunction SIN *
087                     ************************
088                     *
089 EA65 E7             RSIN    RST   4         MSIN
090 EA66 36                     DATA  :36
091 EA67 C9                     RET
092                     *
093                     ************************
094                     * RUN basicfunction COS *
095                     ************************
096                     *
097 EA68 E7             RCOS    RST   4         MCOS
098 EA69 39                     DATA  :39
099 EA6A C9                     RET
100                     *
101                     ************************
102                     * RUN basicfunction TAN *
103                     ************************
104                     *
105 EA6B E7             RTAN    RST   4         MTAN
106 EA6C 3C                     DATA  :3C
107 EA6D C9                     RET
108                     *
109                     ************************
110                     * RUN basicfunction ASIN *
111                     ************************
112                     *
113 EA6E E7             RASIN   RST   4         MASIN
114 EA6F 3F                     DATA  :3F
115 EA70 C9                     RET
116                     *
117                     ************************
118                     * RUN basicfunction ACOS *
119                     ************************
120                     *
121 EA71 E7             RACOS   RST   4         MACOS
122 EA72 42                     DATA  :42
123 EA73 C9                     RET
124                     *
125                     *
```

```
                        **************************
                        * RUN basicfunction ATAN *
                        **************************
                        *
EA74 E7         RATN    RST   4          MATAN
EA75 45                 DATA  :45
EA76 C9                 RET
                        *
                        **************************
                        * RUN basicfunction STR$ *
                        **************************
                        *
                        * Converts a FPT number into a string.
                        *
EA77 CD9BCE     RSTR    CALL  :CE9B      Convert MACC for FPT output
                                         string in DECBUF
EA7A 00                 NOP
EA7B 00                 NOP
EA7C 00                 NOP
EA7D 2A33C0     RST20   LHLD  :C033      Get addr DECBUF
EA80 1E01               MVI   E,:01      Pretend it is a variable
EA82 C9                 RET
                        *
                        **************************
                        * RUN basicfunction HEX$ *
                        **************************
                        *
                        * Converts a INT number into an equivalent string.
                        *
EA83 CD08E8     RHEX    CALL  :E808      Eval expr, result in MACC
EA86 CD2DC0             CALL  :C02D      Conv. MACC to HEX for output
EA89 C37DEA             JMP   :EA7D      Get addr DECBUF, pretend it
                                         is a variable
                        *
                        **************************
                        * RUN basicfunction SPC *
                        **************************
                        *
                        * Returns a string of a number of spaces.
                        * From SPC10 used by TAB if DOUTC<>0.
                        *
EA8C CD1DE7     RSPC    CALL  :E71D      Get nr of spaces in A

                        * Entry from RTAB:

EA8F CD8BD1     SPC10   CALL  :D18B      Get place in heap for string
                                         of spaces
EA92 E5                 PUSH  H          Save pntr to heap
EA93 B7         SPC20   ORA   A
EA94 CA9EEA             JZ    :EA9E      Jump if ready
EA97 23                 INX   H
EA98 3620               MVI   M,:20      Space into heap
EA9A 3D                 DCR   A
EA9B C393EA             JMP   :EA93      Next space
EA9E E1         SPC30   POP   H          HL pnts to start string
EA9F C3DFEA             JMP   :EADF      Pretend it is a temp string
                        *
                        **************************
                        * RUN basicfunction TAB *
                        **************************
                        *
                        * Returns a string of spaces to move cursor to a
```



```
                        * given character position (only to the right).
                        * Works only if output switch DOUTC=0, else it
                        * returns one space only.
                        *
EAA2 CD60CE     RTAB    CALL  :CE60      Get nr of tabs in L,
                                         DOUTC in A
EAA5 B7                 ORA   A          Check output direction
EAA6 3E01               MVI   A,:01      Init 1 space
EAA8 C28FEA             JNZ   :EA8F      Jump if DOUTC<>0
EAAB 7D                 MOV   A,L        Get nr of tabs
EAAC 00                 NOP
EAAD 00                 NOP
EAAE EF                 RST   5          Ask cursor pos and size
EAAF 0C                 DATA  :0C        char screen
EAB0 95                 SUB   L          Calc nr of spaces reqd
EAB1 D28FEA             JNC   :EA8F      Run SPC if not past tab pos
EAB4 AF                 XRA   A          If past TAB:
EAB5 C38FEA             JMP   :EA8F      Run SPC for no spaces
                        *
                        **************************
                        * RUN basicfunction CURX *
                        **************************
                        *
EAB8 EF         RCURX   RST   5          Ask cursor pos and size
EAB9 0C                 DATA  :0C        char screen
EABA 7D                 MOV   A,L        X-coord in A
EABB C37CEB             JMP   :EB7C      and into MACC
                        *
                        **************************
                        * RUN basicfunction CURY *
                        **************************
                        *
EABE EF         RCURY   RST   5          Ask cursor pos and size
EABF 0C                 DATA  :0C        char screen
EAC0 7C                 MOV   A,H        Y-coord in A
EAC1 C37CEB             JMP   :EB7C      and in MACC
                        *
                        **************************
                        * RUN basicfunction LEN *
                        **************************
                        *
                        * Given a string, returns length of the string.
                        *
EAC4 CD91E7     RLEN    CALL  :E791      Eval string expr
EAC7 7E         RLE10   MOV   A,M        Get length in A
EAC8 C37CEB             JMP   :EB7C      and into MACC
                        *
                        **************************
                        * RUN basicfunction ASC *
                        **************************
                        *
                        * Given a string, returns ASCII value of 1st char.
                        *
EACB CD91E7     RASC    CALL  :E791      Eval string expr
EACE 7E                 MOV   A,M        Get length in A
EACF C37ECF             JMP   :CF7E      Check if length is 0; get
                                         1st char in MACC if not
                        *
                        **************************
                        * RUN basicfunction CHR$ *
                        **************************
                        *
```

```
EAD2 CD1DE7     RCHR    CALL  :E71D     Get argument value in A
EAD5 F5                 PUSH  PSW       Save it
EAD6 3E01               MVI   A,:01
EAD8 CD8BD1             CALL  :D18B     Find place in heap for
                                        a 1-byte string
EADB F1                 POP   PSW       Get argument
EADC 23                 INX   H
EADD 77                 MOV   M,A       Store it in Heap
EADE 2B                 DCX   H         Pnts to length byte

                * Entry from 'SPC':

EADF 1E02       RCR10   MVI   E,:02     Status: temporary
EAE1 C9                 RET
                *
                ****************************
                * RUN basicfunction LEFT$ *
                ****************************
                *
                * Given a string, returns a number of characters
                * from the left end.
                *
EAE2 CD9DE7     RLEFT   CALL  :E79D     Eval string expr
EAE5 E5                 PUSH  H         Save string pntr
EAE6 D5                 PUSH  D
EAE7 CD1DEB             CALL  :EB1D     Reqd length in A
EAEA 1600               MVI   D,:00
EAEC 5F         RLF10   MOV   E,A       Length in DE
EAED CD4FD1     RLF20   CALL  :D14F     Extract substring
EAF0 D215DA             JNC   :DA15     Evt. run error 'NUMBER OUT
                                        OF RANGE'
EAF3 D1                 POP   D
EAF4 E3                 XTHL
EAF5 7B                 MOV   A,E       Get status
EAF6 FE02               CPI   :02       Temporary?
EAF8 CC87D1             CZ    :D187     Then clear heap entry
EAFB E1                 POP   H
EAFC 1E02               MVI   E,:02     Status temporary
EAFE C9                 RET
                *
                ****************************
                * RUN basicfunction RIGHT$ *
                ****************************
                *
                * Extracts a number of characters from the
                * right end of a given string.
                *
EAFF CD9DE7     RRIGHT  CALL  :E79D     Eval string expr
EB02 E5                 PUSH  H         Save string pntr
EB03 D5                 PUSH  D
EB04 CD1DEB             CALL  :EB1D     Get length substring
EB07 5F                 MOV   E,A       in E
EB08 7E                 MOV   A,M       Get total string length
EB09 93                 SUB   E
EB0A 57                 MOV   D,A       Startposition in D
EB0B C3EDEA             JMP   :EAED     Extract substring
                *
                ****************************
                * RUN basicfunction MID$ *
                ****************************
                *
EB0E CD9DE7     RMID    CALL  :E79D     Eval string expr
```



```
EB11 E5                 PUSH  H         Save string pntr
EB12 D5                 PUSH  D
EB13 CD1DEB             CALL  :EB1D     Get startposition
EB16 57                 MOV   D,A       in D
EB17 CD1DE7             CALL  :E71D     Get length in A
EB1A C3ECEA             JMP   :EAEC     Extract substring
                *
                ******************************
                * GET VALUE OF ARGUMENT IN A *
                ******************************
                *
                * Exit: DEHL preserved.
                *
EB1D E5         REXIK   PUSH  H
EB1E D5                 PUSH  D
EB1F CD1DE7             CALL  :E71D     Get value of argument in A
EB22 D1                 POP   D
EB23 E1                 POP   H
EB24 C9                 RET
                *
                ************************
                * RUN basicfunction VAL *
                ************************
                *
                * Takes a string and converts it to a FPT number.
                *
EB25 CD91E7     RVAL    CALL  :E791     Eval string expr
EB28 C5                 PUSH  B
EB29 7E         SUEPT   MOV   A,M       Length of string in A
EB2A 323401             STA   :0134     and in EFECT
EB2D 23                 INX   H         HL pnts to 1st string byte
EB2E 223201             SHLD  :0132     Addr into EFEPT
EB31 0E00               MVI   C,:00     Char count
EB33 213501             LXI   H,:0135
EB36 3601               MVI   M,:01     Input from string
EB38 CD1EC0             CALL  :C01E     encode FPT number into MACC
EB3B 35                 DCR   M         Input from keyboard
EB3C 3E0A               MVI   A,:0A     If over/underflow: run
EB3E D2F5D9             JNC   :D9F5     error 'INVALID NUMBER'
EB41 C1                 POP   B
EB42 C9                 RET
                *
                ************************
                * RUN basicfunction FRE *
                ************************
                *
                * Returns a INT given size of free RAM space.
                * Result in MACC.
                * FR2BY: Also used to copy HL into MACC.
                *
                * Exit: BC preserved, AFDEHL corrupted.
                *
EB43 CD51EB     RFRE    CALL  :EB51     Calc free RAM space in HL
EB46 AF         FR2BY   XRA   A
EB47 C5                 PUSH  B
EB48 D5                 PUSH  D
EB49 4C                 MOV   C,H       ) Free space in CD
EB4A 55                 MOV   D,L       )
EB4B 47                 MOV   B,A       A,B=0
EB4C E7                 RST   4         Copy reg A,B,C,D into MACC
EB4D 12                 DATA  :12
EB4E D1                 POP   D
```

```
374 EB4F C1                 POP   B
375 EB50 C9                 RET
376                 *
377                 ****************************
378                 * CALCULATE FREE RAM SPACE *
379                 ****************************
380                 *
381                 * Exit: HL: Free RAM space.
382                 *       DE: STBUSE.
383                 *       ABC preserved, F corrupted.
384                 *
385 EB51 2AA302    SIZE    LHLD  :02A3     Get end symtab
386 EB54 EB               XCHG              in DE
387 EB55 2AA502           LHLD  :02A5     Get bottom screen RAM
388 EB58 CD1ADE           CALL  :DE1A     Calc. free space in HL
389 EB5B C9               RET
390                 *
391                 ************************
392                 * RUN basicfunction FREQ *
393                 ************************
394                 *
395                 * Given a frequency in Hz, returns a period in
396                 * 'oscillator cycles' (INT).
397                 *
398                 * Entry: MACC: Value for freq.
399                 * Exit:  BC preserved, AFDEHL corrupted.
400                 *
401 EB5C 212901    RFREQ   LXI   H,:0129   Startaddr scratch area for
402                                         expression evaluation
403 EB5F E7                 RST   4         Copy reqd freq to scratch
404 EB60 0F                 DATA  :0F       area
405 EB61 E5                 PUSH  H         Save startaddr scratch area
406 EB62 21EDD0             LXI   H,:D0ED   Addr sound constant
407 EB65 E7                 RST   4         Sound constant into MACC
408 EB66 0C                 DATA  :0C
409 EB67 E1                 POP   H         Get start scratch area
410 EB68 E7                 RST   4         Calc sound const/reqd freq
411 EB69 09                 DATA  :09
412 EB6A E7                 RST   4         Change it to INT
413 EB6B 48                 DATA  :48
414 EB6C C5                 PUSH  B
415 EB6D E7                 RST   4         Copy result to reg A,B,C,D
416 EB6E 15                 DATA  :15
417 EB6F B0                 ORA   B         > 64K ? Then run error
418 EB70 C215DA             JNZ   :DA15     'NUMBER OUT OF RANGE'
419 EB73 C1                 POP   B
420 EB74 C9                 RET
421                 *
422                 **********************
423                 * DATA - (not used) *
424                 **********************
425                 *
426 EB75 15         LOE235  DATA  :15       Sound constant
427 EB76 F4                 DATA  :F4
428 EB77 24                 DATA  :24
429 EB78 00                 DATA  :00
430                 *
431                 ************************
432                 * RUN basicfunction GETC *
433                 ************************
434                 *
435                 * Gets one character from keyboard. Returns its
```



```
436                 * ASCII value in MACC; 0 if no inputs.
437                 * FR1BY: Also used to copy 1 byte into MACC.
438                 *
439 EB79 CDBBD6    RGETC   CALL  :D6BB     Scan keyboard, result in A
440 EB7C 6F        FR1BY   MOV   L,A       Result in L
441 EB7D 2600              MVI   H,:00
442 EB7F C346EB            JMP   :EB46     Result into MACC
443                 *
444                 ************************
445                 * RUN basicfunction INP *
446                 ************************
447                 *
448                 * Reads a byte from a Real World address (DCE-bus).
449                 *
450 EB82 CD1DE7    RINP    CALL  :E71D     Get RW addr in A
451 EB85 57                MOV   D,A       and in D
452 EB86 CDE0D8            CALL  :D8E0     Get input from DCE-bus
453 EB89 7B                MOV   A,E       Result in A
454 EB8A C37CEB            JMP   :EB7C     Result into MACC
455                 *
456                 ************************
457                 * RUN basicfunction INT *
458                 ************************
459                 *
460                 * Returns a integer FPT value, just less than the
461                 * FPT argument given.
462                 * REMARK: Routine is wrong if -1 < nr < 0. Then
463                 *         the result is -1 !
464                 *
465 EB8D C5         RINT    PUSH  B
466 EB8E E7                 RST   4         Copy MACC to reg A,B,C,D
467 EB8F 15                 DATA  :15
468 EB90 C1                 POP   B
469 EB91 E7                 RST   4         Change MACC to INT, and then
470 EB92 1E                 DATA  :1E       to FPT
471 EB93 21F1D0             LXI   H,:D0F1   Addr FPT(-1)
472 EB96 B7                 ORA   A
473 EB97 F29CEB             JP    :EB9C     Abort if positive
474 EB9A E7                 RST   4         Add -1 if MACC negative
475 EB9B 00                 DATA  :00
476 EB9C C9         LOE239  RET
477                 *
478                 **********************
479                 * DATA - (not used) *
480                 **********************
481                 *
482 EB9D 81         LOE240  DATA  :81       FPT (-1)
483 EB9E 80                 DATA  :80
484 EB9F 00                 DATA  :00
485 EBA0 00                 DATA  :00
486                 *
487                 ****************************
488                 * RUN basicfunction VARPTR *
489                 ****************************
490                 *
491 EBA1 CD63E9    RVPT    CALL  :E963     Get varptr in HL, T/L in A
492 EBA4 C346EB            JMP   :EB46     Varptr into MACC
493                 *
494                 ************************
495                 * RUN basicfunction XMAX *
496                 ************************
497                 *
```

```
EBA7 CDB4EB     RXMAX   CALL  :EBB4     Get max Y,X-coord graph area
EBAA EB                 XCHG            Max X-coord in HL
EBAB C346EB             JMP   :EB46     and into MACC
                *
                ************************
                * RUN basicfunction YMAX *
                ************************
                *
EBAE CDB4EB     RYMAX   CALL  :EBB4     Get max Y,X-coord graph area
EBB1 C37CEB             JMP   :EB7C     Max Y-coord into MACC
                *
                ****************************************
                * GET MAX. Y,X-COORDINATES GRAPHICS AREA *
                ****************************************
                *
                * Exit: DE: Max. X-coordinate.
                *       A:  Max. Y-coordinate.
                *       BC preserved.
                *
EBB4 210000     L0E245  LXI   H,:0000   ) Coord dot 0,0
EBB7 C5                 PUSH  B         )
EBB8 4C                 MOV   C,H       )
EBB9 EF                 RST   5         Ask colour of point and
EBBA 27                 DATA  :27       size graphics screen
EBBB DA02E6             JC    :E602     Evt run screen error
EBBE 78                 MOV   A,B       Max Y-coord in A
EBBF C1                 POP   B
EBC0 C9                 RET
                *
                ************************
                * RUN basicfunction PDL *
                ************************
                *
                * A given paddle channel is enabled. Counter 0 is
                * set to FFFF. The counter is read over and over
                * until it is counted out.
                *
                * Exit: BC updated, AFDEHL corrupted.
                *
EBC1 3E05       RPDL    MVI   A,:05
EBC3 CD43E7             CALL  :E743     Get paddle select (0-5)
EBC6 57                 MOV   D,A       into D
EBC7 3A4000             LDA   :0040     Get POROM
EBCA E6F8               ANI   :F8       ROM/cass.select only
EBCC B2                 ORA   D         OR with paddle select
EBCD F608               ORI   :08       Paddle enable
EBCF CD08D8             CALL  :D808     Load PORO/POROM
EBD2 C5                 PUSH  B
EBD3 3E30               MVI   A,:30
EBD5 0106FC             LXI   B,:FC06   Addr 8253 cmd word
EBD8 02                 STAX  B         Select ch.0, mode 0, 2 byte
EBD9 21FFFF             LXI   H,:FFFF
EBDC 2200FC             SHLD  :FC00     Load counter ch.0
EBDF 3A01FD             LDA   :FD01     Get pdl timer trig impulse
                                        (Useless: A is cleared in
                                        OEBE3)
EBE2 EB         PDL10   XCHG            DE = FFFF
EBE3 3E00               MVI   A,:00
EBE5 02                 STAX  B         (FC06)=00: counter 0, latch
                                        operation
EBE6 2A00FC             LHLD  :FC00     Get contents counter 0
EBE9 CD14DE             CALL  :DE14     Compare HL-DE
```



```
EBEC DAE2EB             JC    :EBE2     Again if DE > HL
EBEF CD26DE             CALL  :DE26     HL = 2-compl. of HL
EBF2 11CEFF             LXI   D,:FFCE   ) Substract 49
EBF5 19                 DAD   D         )
EBF6 DAFCEB             JC    :EBFC     If result negative
EBF9 210000             LXI   H,:0000
EBFC 7C         PDL20   MOV   A,H
EBFD B7                 ORA   A
EBFE CA06EC             JZ    :EC06
EC01 2EFF               MVI   L,:FF
EC03 00                 NOP
EC04 00                 NOP
EC05 00                 NOP
EC06 3E36       PDL30   MVI   A,:36
EC08 02                 STAX  B         (FC06)=#36: Chan 0, mode 3
EC09 3A4000             LDA   :0040     Get PORO/POROM
EC0C E6F0               ANI   :F0       Disable paddle operation
EC0E CD06D8             CALL  :D806     Load PORO/POROM
EC11 C1                 POP   B
EC12 7D                 MOV   A,L       A=0 if result negative,
                                        else FF
EC13 C37CEB             JMP   :EB7C     Move A into MACC
                *
                *
                *
EC16                    END
```

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

FR1BY   EB7C    FR2BY   EB46    L0E235  EB75    L0E239  EB9C
L0E240  EB9D    L0E245  EBB4    MPT48   EA40    PDL10   EBE2
PDL20   EBFC    PDL30   EC06    RABS    EA50    RACOS   EA71
RALOG   EA53    RASC    EACB    RASIN   EA6E    RATN    EA74
RCHR    EAD2    RCOS    EA68    RCR10   EADF    RCURX   EAB8
RCURY   EABE    REXIK   EB1D    REXP    EA56    RFRAC   EA59
RFRE    EB43    RFREQ   EB5C    RGETC   EB79    RHEX    EA83
RINP    EB82    RINT    EB8D    RLE10   EAC7    RLEFT   EAE2
RLEN    EAC4    RLF10   EAEC    RLF20   EAED    RLOG    EA5C
RLOGT   EA5F    RMID    EB0E    RPDL    EBC1    RRIGHT  EAFF
RSIN    EA65    RSPC    EA8C    RSQR    EA62    RST20   EA7D
RSTR    EA77    RTAB    EAA2    RTAN    EA6B    RTK55   EA47
RVAL    EB25    RVPT    EBA1    RXMAX   EBA7    RYMAX   EBAE
SIZE    EB51    SPC10   EA8F    SPC20   EA93    SPC30   EA9E
SUEPT   EB29
```

```
002                             ORG   :EC16
003                     *
004                     *
005                     *
006                     ****************************
007                     * RUN basicfunction PEEK *
008                     ****************************
009                     *
010                     * Returns the contents of a memory location
011                     * with an address given as INT argument.
012                     *
013 EC16 CDF8E6         RPEEK   CALL  :E6F8     Get addr in HL
014 EC19 7E                     MOV   A,M       Get its contents
015 EC1A C37CEB                 JMP   :EB7C     Move it into MACC
016                     *
017                     ************************
018                     * RUN basicfunction PI *
019                     ************************
020                     *
021                     * Returns a value of 3.14159.
022                     *
023 EC1D 21F5D0         RPI     LXI   H,:D0F5   Addr FPT (PI)
024 EC20 E7                     RST   4         FPT (PI) into MACC
025 EC21 0C                     DATA  :0C
026 EC22 C9                     RET
027                     *
028                     **********************
029                     * DATA - (not used) *
030                     **********************
031                     *
032 EC23 02             L0E252  DATA  :02       FPT (PI)
033 EC24 C9                     DATA  :C9
034 EC25 0F                     DATA  :0F
035 EC26 DB                     DATA  :DB
036                     *
037                     ****************************
038                     * RUN basicfunction RND *
039                     ****************************
040                     *
041                     * Returns a random or pseudo-random number.
042                     * If argument > 0: Returns a pseudo-random number
043                     *                  in the range 0 <= R < argument.
044                     * If argument = 0: Returns a hardware random number
045                     *                  0 < R < 1.
046                     * If argument < 0: Number replaces the kernel for
047                     *                  calculating further pseudo-
048                     *                  random numbers.
049                     *
050 EC27 CD8AEC         RRND    CALL  :EC8A     Test if arg is 0
051 EC2A CA0CE4                 JZ    :E40C     Then hardware random
052 EC2D 212D01                 LXI   H,:012D   Addr randdom number kernel
053 EC30 F235EC                 JP    :EC35     If pseudo random number
054 EC33 E7                     RST   4         Copy MACC to kernel
055 EC34 0F                     DATA  :0F
056 EC35 EB                     XCHG
057 EC36 212901                 LXI   H,:0129   Addr scratch area WORKE
058 EC39 E7                     RST   4         Copy MACC to WORKE
059 EC3A 0F                     DATA  :0F
060 EC3B E5                     PUSH  H         Saveaddr WORKE
061 EC3C EB                     XCHG
062 EC3D E5                     PUSH  H         Save addr RNUM
063 EC3E 3601                   MVI   M,:01     Limit range 1-2
```



```
064 EC40 E7                     RST   4         Copy last nr from RNUM
065 EC41 0C                     DATA  :0C       into MACC
066 EC42 1605                   MVI   D,:05
067 EC44 21A8C6         RRD10   LXI   H,:C6A8   Addr RNDA
068 EC47 E7                     RST   4         Multiply R0*RNDA
069 EC48 54                     DATA  :54
070 EC49 21ACC6                 LXI   H,:C6AC   Addr RNDB
071 EC4C E7                     RST   4         Add RNDB to R0*RNDA
072 EC4D 4E                     DATA  :4E
073 EC4E 00                     NOP
074 EC4F 00                     NOP
075 EC50 00                     NOP
076 EC51 00                     NOP
077 EC52 00                     NOP
078 EC53 21FDD0                 LXI   H,:D0FD   Addr AND mask
079 EC56 E7                     RST   4         IAND: pick out mantissa
080 EC57 63                     DATA  :63
081 EC58 21B0C6                 LXI   H,:C6B0   Addr OR mask
082 EC5B E7                     RST   4         IOR: set mantissa top
083 EC5C 66                     DATA  :66       bit, + range 1-2
084 EC5D 15                     DCR   D
085 EC5E C244EC                 JNZ   :EC44     Again if D<>0
086 EC61 E1                     POP   H         Get addr RNUM
087 EC62 E7                     RST   4         Copy MACC to RNUM
088 EC63 0F                     DATA  :0F
089 EC64 21F1D0                 LXI   H,:D0F1   Addr FPT (-1)
090 EC67 E7                     RST   4         Add -1 to MACC (range
091 EC68 00                     DATA  :00       0-1)
092 EC69 E1                     POP   H         Get addr WORKE
093 EC6A E7                     RST   4         Frig range: multiply
094 EC6B 06                     DATA  :06       MACC*(WORKE)
095 EC6C C9                     RET
096                     *
097                     ****************************
098                     * part of RUN TALK (CD64) *
099                     ****************************
100                     *
101                     * Entry: Z=1: Code = 0C (delay).
102                     *        Z=0: Code = 0D (ML call)
103                     *
104 EC6D 5E             MPT46   MOV   E,M       )
105 EC6E 23                     INX   H         ) Wait-time/ML address
106 EC6F 56                     MOV   D,M       ) in HL
107 EC70 23                     INX   H         )
108 EC71 EB                     XCHG            )
109 EC72 CCCCDA                 CZ    :DACC     If to be waited
110 EC75 C4A9C8                 CNZ   :C8A9     Else: Run ML routine
111 EC78 C367CD                 JMP   :CD67     Return: Handle next code
112                     *
113                     ************************
114                     * RUN basicfunction SGN *
115                     ************************
116                     *
117                     * Takes a FPT value and returns:
118                     *                +1  if value is positive.
119                     *                 0  if value is zero.
120                     *                -1  if value is negative.
121                     *
122 EC7B CD8AEC         RSGN    CALL  :EC8A     Test if variable is zero
123 EC7E C8                     RZ              Then ready
124 EC7F 21F1D0                 LXI   H,:D0F1   Addr FPT(-1)
125 EC82 E7                     RST   4         Copy -1 into MACC
```

```
EC83 0C                 DATA  :0C
EC84 FA89EC             JM    :EC89     Ready if already negative
EC87 E7                 RST   4         Else change sign MACC
EC88 1B                 DATA  :1B       (make MACC  +1)
EC89 C9           LOE257  RET
                  *
                  ************************
                  * TEST A FPT VARIABLE *
                  ************************
                  *
                  * Entry: Variable in MACC.
                  * Exit:  Z=1: Variable is zero.
                  *        Z=0: Other flags set on exponent byte
                  *             of variable.
                  *        ABCDEHL preserved.
                  *
EC8A C5           FTEST   PUSH  B
EC8B D5                   PUSH  D
EC8C F5                   PUSH  PSW
EC8D E7                   RST   4         Copy MACC to reg A,B,C,D
EC8E 15                   DATA  :15
EC8F 5F                   MOV   E,A       Exp byte in E
EC90 B0                   ORA   B         )
EC91 B1                   ORA   C         ) Check if nr is zero
EC92 B2                   ORA   D         )
EC93 CA98EC               JZ    :EC98     Then quit
EC96 7B                   MOV   A,E       Get exp byte
EC97 B7                   ORA   A         Set flags on it
EC98 D1           FTS10   POP   D
EC99 7A                   MOV   A,D
EC9A D1                   POP   D
EC9B C1                   POP   B
EC9C C9                   RET
                  *
                  ***************************
                  * RUN basicfunction SCRN *
                  ***************************
                  *
EC9D CDF3E5       RSCRN   CALL  :E5F3     Eval given coord
ECA0 C5                   PUSH  B
ECA1 4F                   MOV   C,A       Y-coord in C
ECA2 EF                   RST   5         Ask colour of dot on screen
ECA3 27                   DATA  :27       + size graphics screen
ECA4 C1                   POP   B
ECA5 DA02E6               JC    :E602     Evt run screen error
ECA8 C37CEB               JMP   :EB7C     Contents screen loc in MACC
                  *
                  *
                  *  ==============
                  *** LIST HANDLER ***
                  *  ==============
                  *
                  * This module lists a program from the textbuffer
                  * onto the screen (or into other required direction)
                  *
                  ************************
                  * LIST A PROGRAM LINE *
                  ************************
                  *
                  * Entry: BC: Points to start of textline.
                  * Exit:  BC: Points to start of next line.
                  *        DEHL preserved, AF corrupted.
```



```
                  *
ECAB D5           SLINE   PUSH  D
ECAC E5                   PUSH  H
ECAD 03                   INX   B         Pnts to line nr
ECAE CDAEEF               CALL  :EFAE     List line nr
ECB1 3E08                 MVI   A,:08
ECB3 CD2ADB               CALL  :DB2A     Cursor to tab 8
ECB6 CDCCEC       LOE262  CALL  :ECCC     List statement
ECB9 0A                   LDAX  B         Get next byte
ECBA B7                   ORA   A
ECBB F2C5EC               JP    :ECC5     If no more statements
ECBE CDF5EF               CALL  :EFF5     Else: print ':'
ECC1 3A                   DATA  :3A
ECC2 C3B6EC               JMP   :ECB6     List next statement
ECC5 CDF5EF       LOE263  CALL  :EFF5     print car.ret
ECC8 0D                   DATA  :0D
ECC9 E1                   POP   H
ECCA D1                   POP   D
ECCB C9                   RET
                  *
                  ********************
                  * LIST A STATEMENT *
                  ********************
                  *
                  * Based on the token in the textbuffer, a particular
                  * statement will be printed.
                  * At first, the Basiccommand will be printed. The
                  * pointers to the particular strings are in a table
                  * starting at CD8B. The base for the table is CC08;
                  * the offset is calculated by TOKEN *3.
                  *
                  * The databyte after the stringaddress pointer
                  * indicates which list-routine has to be used for
                  * the rest of the statement. This byte is a offset
                  * for table ECF8.
                  *
                  * Entry: BC: Points to token.
                  * Exit:  BC: Points to next statement.
                  *        AFDEHL corrupted.
                  *        On stack: Returnaddress from this sub-
                  *        routine.
                  *
ECCC 0A           SCOM    LDAX  B         Get token
ECCD 03                   INX   B         Update pointer
ECCE 5F                   MOV   E,A       token in E
ECCF 1600                 MVI   D,:00
ECD1 2108CC               LXI   H,:CC08   Startaddr stringtable
ECD4 19                   DAD   D         ) Add 3* token
ECD5 19                   DAD   D         )
ECD6 19                   DAD   D         )
ECD7 5E                   MOV   E,M       Get lobyte stringaddr
ECD8 23                   INX   H
ECD9 56                   MOV   D,M       Get hibyte stringaddr
ECDA 23                   INX   H         Point to data after addr
ECDB EB                   XCHG            Stringaddr in HL
ECDC 7E                   MOV   A,M       Get length byte of string
ECDD B7                   ORA   A         Length=0?
ECDE CD32DB               CALL  :DB32     List Basiccmd string
ECE1 1A                   LDAX  D         Get data byte after string-
                                          address
ECE2 CAEAEC               JZ    :ECEA     If length string =0
ECE5 FE00                 CPI   :00
```

```
ECE7 C46BCE             CNZ   :CE6B     Print space if byte after
                                        stringaddr <>0
ECEA 1A         SCM10   LDAX  D         Get data byte (= offset)
ECEB 87                 ADD   A         Offset *2
ECEC 5F                 MOV   E,A       in E
ECED 1600               MVI   D,:00
ECEF 21F8EC             LXI   H,:ECF8   Startaddr table Listroutines
ECF2 19                 DAD   D         Add offset
ECF3 5E                 MOV   E,M       Get addr in DE
ECF4 23                 INX   H
ECF5 56                 MOV   D,M
ECF6 EB                 XCHG            Addr routine in HL
ECF7 E9                 PCHL            Go to this adress
                *
                ************************************
                * POINTERS LIST HANDLING ROUTINES *
                ************************************
                *
                * Table with addresses of listroutines for the
                * part of a statement after a token.
                *
                * Startaddress table is ECF8. The offset (given
                * between brackets) is identical to the data
                * byte after the addresses in the table on CD8B.
                *
ECF8 3AED       L0E355  DBL   :ED3A     (00)  nothing more
ECFA 41ED               DBL   :ED41     (01)  linenr
ECFC 3BED               DBL   :ED3B     (02)  linenr linenr
                                        .     (not used)
ECFE 44ED               DBL   :ED44     (03)  unquoted string
ED00 4DED               DBL   :ED4D     (04)  E (E=expr)
ED02 47ED               DBL   :ED47     (05)  E,E
ED04 56ED               DBL   :ED56     (06)  E E
ED06 62ED               DBL   :ED62     (07)  E,E E
ED08 5CED               DBL   :ED5C     (08)  E,E E,E E
ED0A 50ED               DBL   :ED50     (09)  E E E E
ED0C 68ED               DBL   :ED68     (0A)  E
ED0E 7AED               DBL   :ED7A     (0B)  liner-linenr
ED10 84ED               DBL   :ED84     (0C)  sound
ED12 9BED               DBL   :ED9B     (0D)  noise
ED14 A4ED               DBL   :EDA4     (0E)  envelope
ED16 C1ED               DBL   :EDC1     (0F)  mode
ED18 D9ED               DBL   :EDD9     (10)  input <string>
ED1A E0ED               DBL   :EDE0     (11)  input/read/dim
ED1C F0ED               DBL   :EDF0     (12)  (not used [*])
ED1E FFED               DBL   :EDFF     (13)  let
ED20 09EE               DBL   :EE09     (14)  if then <E>
ED22 1AEE               DBL   :EE1A     (15)  if goto <linenr>
ED24 25EE               DBL   :EE25     (16)  if then <linenr>
ED26 30EE               DBL   :EE30     (17)  for to step
ED28 4FEE               DBL   :EE4F     (18)  next
ED2A 52EE               DBL   :EE52     (19)  print
ED2C 66EE               DBL   :EE66     (1A)  on goto
ED2E 71EE               DBL   :EE71     (1B)  on gosub
ED30 87EE               DBL   :EE87     (1C)  callm
ED32 94EE               DBL   :EE94     (1D)  (not used [*])
ED34 9ED8               DBL   :D89E     (1E)  savea/loada
                *
                * The vectors marked with [*] are no pointers to
                * LIST routines.
                *
ED36 FF                 DATA  :FF
```



```
ED37 FF                 DATA  :FF
ED38 FF                 DATA  :FF
ED39 FF                 DATA  :FF
                *
                ********************************
                * LIST NO FURTHER EXPRESSIONS *
                ********************************
                *
ED3A C9         SCN1    RET
                *
                ****************************
                * LIST 1 OR 2 LINENUMBERS *
                ****************************
                *
                * SCN2: List 1 line number.
                * SCN3: List 2 line numbers, separated by space.
                *       (This last entry is not used).
                *
                * Exit: BC updated, DE preserved, AFHL corrupted.
                *
ED3B CDAEEF     SCN3    CALL  :EFAE     List linenr
ED3E CD6BCE             CALL  :CE6B     Print space
ED41 C3AEEF     SCN2    JMP   :EFAE     List linenr
                *
                ************************
                * LIST UNQUOTED STRING *
                ************************
                *
ED44 C3EDEF     SCN5    JMP   :EFED     List unquoted string
                *
                **********************
                * LIST <EXPR>,<EXPR> *
                **********************
                *
                * Exit: BC updated, DE preserved, AFHL corrupted.
                *
ED47 CDA2EE     SCN7    CALL  :EEA2     List <expr>
ED4A CD70CE     SCOEX   CALL  :CE70     Print ','
ED4D C3A2EE     SCN6    JMP   :EEA2     List <expr>
                *
                ************************************
                * LIST <EXPR> <EXPR> <EXPR> <EXPR> *
                ************************************
                *
                * Exit: BC updated, DE preserved, AFHL corrupted.
                *
ED50 CDFCEF     SCN11   CALL  :EFFC     List <expr>; print space
ED53 CDFCEF     S3EXP   CALL  :EFFC     Idem
ED56 CDFCEF     SCN8    CALL  :EFFC     Idem
ED59 C3A2EE             JMP   :EEA2     List <expr>
                *
                ********************************************
                * LIST <EXPR>,<EXPR> <EXPR>,<EXPR> <EXPR> *
                ********************************************
                *
                * Exit: BC updated, DE preserved, AFHL corrupted.
                *
ED5C CD47ED     SCN10   CALL  :ED47     List <expr>,<expr>
ED5F CD6BCE             CALL  :CE6B     Print space
ED62 CD47ED     SCN9    CALL  :ED47     List <expr>,<expr>
ED65 CD6BCE     SCSEX   CALL  :CE6B     Print space
ED68 C3A2EE     L0E343  JMP   :EEA2     List <expr>
```

```
                        *
                        ********************************
                        * LIST <EXPR>,<EXPR>(,<EXPR>) *
                        ********************************
                        *
ED6B CD47ED     SCN12   CALL  :ED47     List <expr>,<expr>
ED6E 0A         S1210   LDAX  B        Get next byte
ED6F FEFF               CPI   :FF       Terminator?
ED71 03                 INX   B
ED72 C8                 RZ              Then abort
ED73 0B                 DCX   B
ED74 CD70CE             CALL  :CE70     Print ','
ED77 C3A2EE             JMP   :EEA2     List <expr>
                        *
                        ***************************
                        * LIST <LINENR>-<LINENR> *
                        ***************************
                        *
                        * Exit: BC updated, DE preserved, AFHL corrupted.
                        *
ED7A CDAEEF     SCN13   CALL  :EFAE     List linenr
ED7D CDF5EF             CALL  :EFF5     Print '-'
ED80 2D                 DATA  :2D
ED81 C3AEEF             JMP   :EFAE     List linenr
                        *
                        *********************************
                        * LIST EXPRESSION AFTER 'SOUND' *
                        *********************************
                        *
                        * Exit: BC updated, AFHL corrupted.
                        *       DE: preserved if ON, corrupted if OFF.
                        *
ED84 0A         SCN14   LDAX  B         Get byte
ED85 FEFF               CPI   :FF       OFF sign?
ED87 C4FCEF             CNZ   :EFFC     If not: List <expr>, print
                                        space
ED8A 0A                 LDAX  B         Get next byte
ED8B FEFF               CPI   :FF       OFF sign?
ED8D C250ED             JNZ   :ED50     If not: List <expr> <expr>
                                        <expr> <expr>; abort
ED90 CD78CE     S1410   CALL  :CE78     Else: print 'OFF'
ED93 97ED               DBL   :ED97
ED95 03                 INX   B
ED96 C9                 RET
                        *
                        * DATA:
                        *
ED97 03         L0E354  DATA  :03
ED98 4F                 DATA  :4F       O
ED99 46                 DATA  :46       F
ED9A 46                 DATA  :46       F
                        *
                        *********************************
                        * LIST EXPRESSION AFTER 'NOISE' *
                        *********************************
                        *
                        * Exit: BC updated, E preserved, AFDHL corrupted.
                        *
ED9B 0A         SC14A   LDAX  B         Get 1st byte NCB
ED9C FEFF               CPI   :FF       OFF sign?
ED9E CA90ED             JZ    :ED90     Then print 'OFF', abort
EDA1 C356ED             JMP   :ED56     Else: List <expr> <expr>
```



```
                        *
                        ************************************
                        * LIST EXPRESSION AFTER 'ENVELOPE' *
                        ************************************
                        *
                        * Exit: BC updated, E preserved, AFDHL corrupted.
                        *
EDA4 CDFCEF     SCN15   CALL  :EFFC     List <expr>, print
                                        space
EDA7 0A                 LDAX  B         Get length of expr
EDA8 03                 INX   B
EDA9 57                 MOV   D,A       into D
EDAA 15         S1510   DCR   D
EDAB FAB8ED             JM    :EDB8     If ready
EDAE CD47ED             CALL  :ED47     List <V>,<T>
EDB1 CDF5EF             CALL  :EFF5     Print ';'
EDB4 3B                 DATA  :3B
EDB5 C3AAED             JMP   :EDAA     Next <V>,<T>
EDB8 0A         S1520   LDAX  B         Get last byte of expr
EDB9 03                 INX   B
EDBA FEFF               CPI   :FF       Terminator?
EDBC C8                 RZ              Then abort
EDBD 0B                 DCX   B
EDBE C3A2EE             JMP   :EEA2     List <expr>
                        *
                        *********************************
                        * LIST EXPRESSION AFTER 'MODE' *
                        *********************************
                        *
EDC1 0A         SCN16   LDAX  B         Get mode byte
EDC2 03                 INX   B
EDC3 1630               MVI   D,:30
EDC5 B7                 ORA   A         Mode 0 (FF) ?
EDC6 FAD4ED             JM    :EDD4     Then print '0'
EDC9 1F                 RAR             CY=1 if A-mode
EDCA 3C         RSA10   INR   A
EDCB F5                 PUSH  PSW
EDCC 82                 ADD   D         Convert to ASCII
EDCD CD60DD             CALL  :DD60     Print modenr
EDD0 F1                 POP   PSW
EDD1 3F                 CMC
EDD2 1641               MVI   D,:41     Prepare print 'A'
EDD4 7A         S1610   MOV   A,D
EDD5 D460DD             CNC   :DD60     Print 'A' if A-mode
EDD8 C9                 RET
                        *
                        ************************************************
                        * LIST EXPRESSION AFTER 'INPUT'-'READ'-'DIM' *
                        ************************************************
                        *
                        * Input with string:
EDD9 CDA2EE     SCN17   CALL  :EEA2     List string
EDDC CDF5EF             CALL  :EFF5     Print ';'
EDDF 3B                 DATA  :3B

                        * Rest:

EDE0 0A         SCN18   LDAX  B         Get nr of variables
EDE1 03                 INX   B
EDE2 57                 MOV   D,A       into D
EDE3 D5         S1810   PUSH  D
```

```
498 EDE4 CDFCEE             CALL  :EEFC     List variable reference
499 EDE7 D1                 POP   D
500 EDE8 15                 DCR   D         Decr nr of variables
501 EDE9 C8                 RZ              Abort if ready
502 EDEA CD70CE             CALL  :CE70     Print ','
503 EDED C3E3ED             JMP   :EDE3     List next variable
504                 *
505                 ****************************************
506                 * part of RUN 'DIM': CALC. REQD. SPACE *
507                 ****************************************
508                 *
509 EDF0 C28FDE      RDM40   JNZ   :DE8F     If length element <= 254:
510                                           then HL=HL*A
511 EDF3 7C                  MOV   A,H       Else:
512 EDF4 B7                  ORA   A         Set flags on hibyte
513 EDF5 65                  MOV   H,L
514 EDF6 2E00                MVI   L,:00
515 EDF8 C8                  RZ              Abort if H=0
516 EDF9 37                  STC             Else: CY=1, L into H
517 EDFA C9                  RET
518                 *
519                 ************************
520                 * RUBBISH - (not used) *
521                 ************************
522                 *
523 EDFB CE          LOE352  DATA  :CE
524 EDFC C3                  DATA  :C3
525 EDFD F4                  DATA  :F4
526 EDFE ED                  DATA  :ED
527                 *
528                 ********************************
529                 * LIST EXPRESSION AFTER 'LET' *
530                 ********************************
531                 *
532                 * Entry: BC: Points to assign statement.
533                 * Exit:  BC updated, AFDEHL corrupted.
534                 *
535 EDFF CDFCEE      SCN20   CALL  :EEFC     List lefthand variable
536                                           reference
537 EE02 CDF5EF              CALL  :EFF5     Print '='
538 EE05 3D                  DATA  :3D
539 EE06 C3A2EE              JMP   :EEA2     List righthand expr
540                 *
541 EE09                    END
```

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

FTEST  EC8A   FTS10  EC98   LOE252 EC23   LOE257 EC89
LOE262 ECB6   LOE263 ECC5   LOE343 ED68   LOE352 EDFB
LOE354 ED97   LOE355 ECF8   MPT46  EC6D   RDM40  EDF0
RPEEK  EC16   RPI    EC1D   RRD10  EC44   RRND   EC27
RSA10  EDCA   RSCRN  EC9D   RSGN   EC7B   S1210  ED6E
S1410  ED90   S1510  EDAA   S1520  EDB8   S1610  EDD4
S1810  EDE3   S3EXP  ED53   SC14A  ED9B   SCM10  ECEA
SCN1   ED3A   SCN10  ED5C   SCN11  ED50   SCN12  ED6B
SCN13  ED7A   SCN14  ED84   SCN15  EDA4   SCN16  EDC1
SCN17  EDD9   SCN18  EDE0   SCN2   ED41   SCN20  EDFF
SCN3   ED3B   SCN5   ED44   SCN6   ED4D   SCN7   ED47
SCN8   ED56   SCN9   ED62   SCOEX  ED4A   SCOM   ECCC
SCOEX  ED45   SHINF  ECAB
```



```
002                         ORG   :EE09
003                 *
004                 *
005                 *
006                 ************************************
007                 * LIST EXPRESSION AFTER 'IF' (1) *
008                 ************************************
009                 *
010                 * Lists '<expr> THEN <expr>'.
011                 *
012 EE09 CDFCEF     SCN21   CALL  :EFFC     List <expr>; print space
013 EE0C CD78CE             CALL  :CE78     Print 'THEN'
014 EE0F 15EE       MPT17   DBL   :EE15
015 EE11 03                 INX   B
016 EE12 C3CCEC             JMP   :ECCC     List statement
017                 *
018                 * DATA :
019                 *
020 EE15 04         LOE350  DATA  :04
021 EE16 54                 DATA  :54       T
022 EE17 48                 DATA  :48       H
023 EE18 45                 DATA  :45       E
024 EE19 4E                 DATA  :4E       N
025                 *
026                 ************************************
027                 * LIST EXPRESSION AFTER 'IF' (2) *
028                 ************************************
029                 *
030                 * Lists '<expr> GOTO <linenr>'.
031                 *
032 EE1A CDFCEF     SCN22   CALL  :EFFC     List <expr>; print space
033 EE1D CD78CE             CALL  :CE78     Print 'GOTO'
034 EE20 F9CB               DBL   :CBF9
035 EE22 C3AEEF             JMP   :EFAE     List linenr
036                 *
037                 ************************************
038                 * LIST EXPRESSION AFTER 'IF' (3) *
039                 ************************************
040                 *
041                 * Lists '<expr> THEN <linenr>'.
042                 *
043 EE25 CDFCEF     SC22A   CALL  :EFFC     List <expr>; print space
044 EE28 CD78CE             CALL  :CE78     Print 'THEN'
045 EE2B 15EE               DBL   :EE15
046 EE2D C3AEEF             JMP   :EFAE     List linenr
047                 *
048                 ********************************
049                 * LIST EXPRESSION AFTER 'FOR' *
050                 ********************************
051                 *
052                 * Lists '<LET statement> TO <expr> (STEP <expr>)'.
053                 *
054 EE30 CDFFED     SCN23   CALL  :EDFF     List expr of LET statement
055 EE33 CD6BCE             CALL  :CE6B     Print space
056 EE36 CD78CE             CALL  :CE78     Print 'TO'
057 EE39 4CEE               DBL   :EE4C
058 EE3B CDA2EE             CALL  :EEA2     List <expr>
059 EE3E 0A                 LDAX  B         Get next byte
060 EE3F 03                 INX   B
061 EE40 FEFF               CPI   :FF       Terminator?
062 EE42 C8                 RZ              Then abort
063 EE43 0B                 DCX   B         Else:
```

```
064 EE44 CD75CE             CALL  :CE75     Print 'STEP'
065 EE47 3DCD               DBL   :CD3D
066 EE49 C3A2EE             JMP   :EEA2     List <expr>
067                 *
068                 * DATA:
069                 *
070 EE4C 02         RLA20   DATA  :02
071 EE4D 54                 DATA  :54       T
072 EE4E 4F                 DATA  :4F       O
073                 *
074                 *********************************
075                 * LIST EXPRESSION AFTER 'NEXT' *
076                 *********************************
077                 *
078 EE4F C3FCEE     SCN24   JMP   :EEFC     List var.ref.
079                 *
080                 *********************************
081                 * LIST EXPRESSIONS AFTER 'PRINT' *
082                 *********************************
083                 *
084 EE52 0A         SCN25   LDAX  B         Get nr of expr
085 EE53 03                 INX   B
086 EE54 57                 MOV   D,A       into D
087 EE55 15         S2510   DCR   D
088 EE56 F8                 RM              Abort if ready
089 EE57 03                 INX   B
090 EE58 CDA2EE             CALL  :EEA2     List <expr>
091 EE5B 0A                 LDAX  B         Get next byte
092 EE5C 03                 INX   B
093 EE5D FEFF               CPI   :FF       Terminator?
094 EE5F C8                 RZ              Then abort
095 EE60 CD60DD             CALL  :DD60     Else: print this byte
096 EE63 C355EE             JMP   :EE55     List next expr
097                 *
098                 *********************************
099                 * LIST EXPRESSION AFTER 'ON' (1) *
100                 *********************************
101                 *
102                 * Lists '<expr> GOTO <linenrs>'.
103                 *
104 EE66 CDFCEF     SCN26   CALL  :EFFC     List <expr>; print space
105 EE69 CD78CE             CALL  :CE78     Print 'GOTO'
106 EE6C F9CB               DBL   :CBF9
107 EE6E C379EE             JMP   :EE79     List linenrs
108                 *
109                 *********************************
110                 * LIST EXPRESSION AFTER 'ON' (2) *
111                 *********************************
112                 *
113                 * Lists '<expr> GOSUB <linenrs>'.
114                 *
115 EE71 CDFCEF     SCN27   CALL  :EFFC     List <expr>; print space
116 EE74 CD78CE             CALL  :CE78     Print 'GOSUB'
117 EE77 01CC               DBL   :CC01
118 EE79 0A         S2710   LDAX  B         Get nr of linenrs
119 EE7A 03                 INX   B
120 EE7B 57                 MOV   D,A       into D
121 EE7C CDAEEF     S2720   CALL  :EFAE     List linenr
122 EE7F 15                 DCR   D
123 EE80 C8                 RZ              Abort if ready
124 EE81 CD70CE             CALL  :CE70     Print ','
125 EE84 C37CEE             JMP   :EE7C     List next linenr
```



```
126                 *
127                 *********************************
128                 * LIST EXPRESSION AFTER 'CALLM' *
129                 *********************************
130                 *
131 EE87 CDA2EE     SCN28   CALL  :EEA2     List <expr>
132 EE8A C36EED             JMP   :ED6E     Print ','; List next expr
133                 *
134                 **********************
135                 * SET I/O DIRECTION *
136                 **********************
137                 *
138                 * Part of RESET (C719). Only used for A = 0.
139                 * Depending on A, the input switch #0296 and
140                 * the output switch #0131 are set.
141                 * Default DINC is RS232.
142                 *
143                 *           INSW:       OTSW:
144                 *      A=0: keyboard    screen/RS232
145                 *      A=1: DINC        screen
146                 *      A=2: DINC        editbuffer
147                 *      A=3: DINC        DOUTC
148                 *
149 EE8D 329602     MPT02   STA   :0296     Select keyb or DINC
150 EE90 323101             STA   :0131     Select screen/RS232/edit/
151                                         DOUTC
152 EE93 C9                 RET
153                 *
154                 ***************
155                 * SET VOLUMES *
156                 ***************
157                 *
158                 * Part of RUN TALK (CD64).
159                 *
160                 * Entry: A:  Parameter code.
161                 *        HL: Pointer to volume byte.
162                 *
163 EE94 E5         RTK40   PUSH  H         Save pntr to volume
164 EE95 6E                 MOV   L,M       Volume in L
165 EE96 260F               MVI   H,:0F
166 EE98 1F                 RAR             Check parameter code
167 EE99 D27CE6             JNC   :E67C     Jump if channel 0/2
168
169                 * If channel 1/N:
170
171 EE9C 29                 DAD   H
172 EE9D 29                 DAD   H
173 EE9E 29                 DAD   H
174 EE9F C37BE6             JMP   :E67B     Continu
175                 *
176                 **********************
177                 * LIST AN EXPRESSION *
178                 **********************
179                 *
180                 * Entry: BC:   Points to expression in program.
181                 *        (BC): 1.... Expr starts with operator.
182                 *              01... Variable reference.
183                 *              001.. Function call.
184                 *              Else  Constant.
185                 * Exit:  BC points after expression.
186                 *        DE preserved, AFHL corrupted.
187                 *
```

```
188 EEA2 D5          SCEXP   PUSH  D
189 EEA3 0A                  LDAX  B          Get opcode
190 EEA4 B7                  ORA   A
191 EEA5 F2DFEE              JP    :EEDF      If no starting operator
192
193                  * If starting with operator:
194
195 EEA8 03                  INX   B
196 EEA9 E61F                ANI   :1F        Only 5 bits of opcode
197 EEAB FE1A                CPI   :1A        '(' ?
198 EEAD F5                  PUSH  PSW        Save opcode
199 EEAE DCA2EE              CC    :EEA2      List expr if binary
200                                           operation
201 EEB1 2186CF              LXI   H,:CF86    Addr table opcode strings
202 EEB4 54          LOE300  MOV   D,H        ) in DE
203 EEB5 5D                  MOV   E,L        )
204 EEB6 CD39DE              CALL  :DE39      HL points after table
205 EEB9 F1                  POP   PSW        Get opcode
206 EEBA F5                  PUSH  PSW
207 EEBB AE                  XRA   M          Comp it with table
208 EEBC 23                  INX   H
209 EEBD E61F                ANI   :1F
210 EEBF C2B4EE              JNZ   :EEB4      Check next opcode if not
211                                           found
212 EEC2 EB                  XCHG             If found: addr string in HL
213 EEC3 23                  INX   H
214 EEC4 7E                  MOV   A,M        Get 1st char
215 EEC5 2B                  DCX   H          Pnts to length
216 EEC6 CD02DE              CALL  :DE02      Check if upper case char
217 EEC9 F5                  PUSH  PSW
218 EECA DC6BCE              CC    :CE6B      Print space if 1st char is
219                                           a letter
220 EECD CD32DB              CALL  :DB32      Print string from table
221 EED0 F1                  POP   PSW
222 EED1 DC6BCE              CC    :CE6B      Print space if 1st char was
223                                           a letter
224 EED4 CDA2EE              CALL  :EEA2      List remaining operand
225 EED7 F1                  POP   PSW        Get orig. opcode
226 EED8 FE1A                CPI   :1A        Was it '(' ?
227 EEDA CC55EF              CZ    :EF55      Then print ')'
228 EEDD D1                  POP   D
229 EEDE C9                  RET
230
231                  * Not starting with operator:
232
233 EEDF 07          LOE301  RLC
234 EEE0 07                  RLC
235 EEE1 DAF2EE              JC    :EEF2      Jump if var.ref
236 EEE4 07                  RLC
237 EEE5 DAEDEE              JC    :EEED      Jump if function call
238
239                  * If constant:
240
241 EEE8 CD84EF              CALL  :EF84      List constant
242 EEEB D1                  POP   D
243 EEEC C9                  RET
244
245                  * If function call:
246
247 EEED CD5AEF      LOE302  CALL  :EF5A      List function reference
248 EEF0 D1                  POP   D
249 EEF1 C9                  RET
```



```
250
251                  * If variable reference:
252
253 EEF2 CDFCEE      LOE303  CALL  :EEFC      List a var.reference
254                                           (array with arguments)
255 EEF5 D1                  POP   D
256 EEF6 C9                  RET
257                  *
258                  ******************************
259                  * LIST A VARIABLE REFERENCE *
260                  ******************************
261                  *
262                  * SCARN:  Entry for arrays without arguments (name
263                  *         only).
264                  * LOE305: Entry for arrays with arguments.
265                  *
266                  * Entry: BC points to variable reference in program.
267                  *
268 EEF7 16BF          SCARN   MVI   D,:BF      Set mask 'no arg'
269 EEF9 C3FEEE              JMP   :EEFE
270                  *
271 EEFC 16FF          LOE305  MVI   D,:FF      Set mask 'with arg'
272 EEFE D5          LOE306  PUSH  D          Save mask
273 EEFF 0A                  LDAX  B          Get byte
274 EF00 03                  INX   B
275 EF01 E63F                ANI   :3F        Skip bit 6,7
276 EF03 57                  MOV   D,A        Rest in D
277 EF04 0A                  LDAX  B          Get next byte
278 EF05 03                  INX   B
279 EF06 5F                  MOV   E,A        in E (Now DE is offset
280                                           of start of symtab)
281 EF07 2AA102              LHLD  :02A1      Get start symtab
282 EF0A 19                  DAD   D          Add offset from start
283 EF0B D1                  POP   D          Get mask
284 EF0C E5                  PUSH  H          Save var.addr in symtab
285 EF0D CD95CA              CALL  :CA95      Find name in symtab
286 EF10 7E                  MOV   A,M        Get T/L byte
287 EF11 A2                  ANA   D          AND with mask
288 EF12 F5                  PUSH  PSW
289 EF13 23                  INX   H
290 EF14 E60F                ANI   :0F        Get length
291 EF16 5E                  MOV   E,M        Get 1st byte of name in E
292 EF17 CD44DB              CALL  :DB44      List name; addr in HL,
293                                           length in A
294 EF1A 1600                MVI   D,:00
295 EF1C 213402              LXI   H,:0234    Startaddr for IMPTAB
296 EF1F 19                  DAD   D          Addr var.type in IMPTAB
297 EF20 F1                  POP   PSW        Get mask
298 EF21 F5                  PUSH  PSW
299 EF22 E630                ANI   :30        Bits 4,5 only
300 EF24 BE                  CMP   M          Comp with IMPTAB
301 EF25 CA3CEF              JZ    :EF3C      Jump if identical
302 EF28 FE00                CPI   :00        FPT ?
303 EF2A 1621                MVI   D,:21      Then D: '!'
304 EF2C CA38EF              JZ    :EF38
305 EF2F FE10                CPI   :10        INT ?
306 EF31 1625                MVI   D,:25      Then D: '%'
307 EF33 CA38EF              JZ    :EF38
308 EF36 1624                MVI   D,:24      Else: D: '$'
309 EF38 7A          LOE307  MOV   A,D
310 EF39 CD60DD              CALL  :DD60      Print type sign
311 EF3C F1          LOE308  POP   PSW        Get mask
```

```
EF3D E640              ANI   :40       Bit 6 only
EF3F E1                POP   H         Get addr of string
EF40 C8                RZ

                     * Bit 6=1 (array with arguments):

EF41 0A                LDAX  B         Get nr of expressions
EF42 03                INX   B
EF43 57                MOV   D,A       in D
EF44 CDF5EF            CALL  :EFF5     Print '('
EF47 28                DATA  :28
EF48 03          LOE309 INX   B
EF49 D5                PUSH  D
EF4A CDA2EE            CALL  :EEA2     List expression
EF4D D1                POP   D
EF4E 15                DCR   D         Ready ?
EF4F C470CE            CNZ   :CE70     If not: print ','
EF52 C248EF            JNZ   :EF48     and list next expr
EF55 CDF5EF      LOE310 CALL  :EFF5     If ready: Print ')'
EF58 29                DATA  :29
EF59 C9                RET
                     *
                     ******************************
                     * LIST A FUNCTION REFERENCE *
                     ******************************
                     *
                     * Finds functionname in table with startaddress
                     * #CFE6 and prints it. Eventual arguments are
                     * printed between brackets.
                     *
                     * Entry: BC points to function code (#20).
                     * Exit:  BC updated, AFEHL preserved, D=0.
                     *
EF5A 03          SFUN  INX   B
EF5B 0A                LDAX  B
EF5C 03                INX   B
EF5D 57                MOV   D,A
EF5E 21E6CF            LXI   H,:CFE6   Startaddr function table
EF61 15          LOE312 DCR   D
EF62 FA6BEF            JM    :EF6B     If found
EF65 CDAECA            CALL  :CAAE     Calc addr next string in tab
EF68 C361EF            JMP   :EF61     Test next function name
EF6B CD32DB      LOE313 CALL  :DB32     List function name
EF6E 7E                MOV   A,M       Get byte after string
EF6F E60F              ANI   :0F       Only nr of following args
EF71 57                MOV   D,A       in D
EF72 C8                RZ              Abort if no arguments
EF73 CDF5EF            CALL  :EFF5     Print '('
EF76 28                DATA  :28
EF77 CDA2EE      LOE314 CALL  :EEA2     List expression
EF7A 15                DCR   D         Decr nr of arg
EF7B C470CE            CNZ   :CE70     If <>0, print ','
EF7E C277EF            JNZ   :EF77     and list next expr
EF81 C355EF            JMP   :EF55     If ready: print ')'
                     *
                     ********************
                     * LIST A CONSTANT *
                     ********************
                     *
                     * The constant is decoded to ASCII, prettied and
                     * printed.
                     *
```



```
                     * Codes:   #10: FPT       #18: Quoted string
                     *          #14: INT       #19: Unquoted string
                     *          #15: HEX
                     *
                     * Entry: BC: Points to constant in program.
                     * Exit:  BC updated, DE preserved, AF corrupted.
                     *        HL: points after end of printed string.
                     *
EF84 0A          SCON  LDAX  B         Get type of constant
EF85 03                INX   B
EF86 60                MOV   H,B       ) Addr constant in HL
EF87 69                MOV   L,C       )

                     * If string:

EF88 FE18              CPI   :18       Quoted string ?
EF8A CAE1EF            JZ    :EFE1     Then list it
EF8D FE19              CPI   :19       Unquoted string ?
EF8F CAEDEF            JZ    :EFED     Then list it

                     * If number:

EF92 E7                RST   4         Copy constant value to MACC
EF93 0C                DATA  :0C
EF94 FE10              CPI   :10       FPT ?
EF96 CAA8EF            JZ    :EFA8     Then list FPT value
EF99 FE14              CPI   :14       INT ?
EF9B F5                PUSH  PSW
EF9C CCBDEF            CZ    :EFBD     List INT value
EF9F F1                POP   PSW
EFA0 C4DAEF            CNZ   :EFDA     Else: list HEX value
EFA3 03          SC010 INX   B
EFA4 03                INX   B
EFA5 03                INX   B
EFA6 03                INX   B         BC points after constant
EFA7 C9                RET
EFA8 CDD4EF      SC020 CALL  :EFD4     List FPT value
EFAB C3A3EF            JMP   :EFA3
                     *
                     **********************
                     * LIST A LINENUMBER *
                     **********************
                     *
                     * Entry: BC: Points to linenumber.
                     * Exit:  BC updated, DE preserved, AFHL corrupted.
                     *
EFAE 0A          LOE318 LDAX  B         Get hibyte linenr
EFAF 03                INX   B
EFB0 67                MOV   H,A       in H
EFB1 0A                LDAX  B         Get lobyte linenr
EFB2 03                INX   B
EFB3 6F                MOV   L,A       in L
EFB4 CD46EB      SLN10 CALL  :EB46     Linenr into MACC
EFB7 7C                MOV   A,H
EFB8 B5                ORA   L         Linenr <>0 ?
EFB9 C4BDEF            CNZ   :EFBD     Then list linenr
EFBC C9                RET
                     *
                     ************************************
                     * LIST A INT VALUE OF MACC CONTENTS *
                     ************************************
                     *
```

```
436                     * The value is the contents of the MACC, prepared
437                     * for output, and moved into the outputbuffer
438                     * DECBUF (#00E3). A Leading space is omitted.
439                     *
440                     * Exit: BCDE preserved. A corrupted.
441                     *       HL: Points after string in DECBUF.
442                     *
443 EFBD CD5FDB        SCINT   CALL  :DB5F     Convert MACC for INT
444                                             output into DECBUF
445 EFC0 2A33C0        SSSPC   LHLD  :C033     Get addr DECBUF
446 EFC3 D5                    PUSH  D
447 EFC4 56                    MOV   D,M       String length in D
448 EFC5 23                    INX   H
449 EFC6 7E                    MOV   A,M       1st char in A
450 EFC7 FE20                  CPI   :20       Space ?
451 EFC9 C2CEEF                JNZ   :EFCE     Jump if no leading space
452 EFCC 23                    INX   H         ) Omit leading space
453 EFCD 15                    DCR   D         )
454 EFCE 7A          SSS10     MOV   A,D       Get nr of char in A
455 EFCF D1                    POP   D
456 EFD0 CD44DB                CALL  :DB44     Print contents DECBUF
457 EFD3 C9                    RET
458                     *
459                     ************************************
460                     * LIST FPT VALUE OF CONTENTS MACC *
461                     ************************************
462                     *
463                     * Exit: BCDE preserved. AF corrupted.
464                     *       HL points after string in DECBUF.
465                     *
466 EFD4 CD9BCE        SCFPT   CALL  :CE9B     Convert MACC for FPT output
467 EFD7 C3C0EF                JMP   :EFC0     List its contents
468                     *
469                     ************************************
470                     * LIST HEX VALUE OF CONTENTS MACC *
471                     ************************************
472                     *
473                     * Exit: BCDE preserved. AF corrupted.
474                     *       HL points after string in DECBUF.
475                     *
476 EFDA CDF5EF        SCHEX   CALL  :EFF5     Print '#'
477 EFDD 23                    DATA  :23
478 EFDE C34ADB                JMP   :DB4A     List in hex
479                     *
480                     ************************
481                     * LIST A QUOTED STRING *
482                     ************************
483                     *
484                     * Entry: BC: Points to string.
485                     * Exit:  BC and HL point after string.
486                     *        AF corrupted. DE preserved.
487                     *
488 EFE1 CDF5EF        SQTS    CALL  :EFF5     Print "
489 EFE4 22                    DATA  :22
490 EFE5 CDEDEF                CALL  :EFED     List string
491 EFE8 CDF5EF                CALL  :EFF5     Print "
492 EFEB 22                    DATA  :22
493 EFEC C9                    RET
494                     *
495                     ************************
496                     * LIST UNQUOTED STRING *
497                     ************************
```



```
498                     *
499                     * Entry: BC points to string.
500                     * Exit:  BC annd HL point after string.
501                     *        AFDE preserved.
502                     *
503 EFED 60          SUQTS   MOV   H,B       ) Stringaddr in HL
504 EFEE 69                  MOV   L,C       )
505 EFEF CD32DB              CALL  :DB32     List string
506 EFF2 44                  MOV   B,H       )
507 EFF3 4D                  MOV   C,L       ) Addr after string in BC
508 EFF4 C9                  RET
509                     *
510                     ******************
511                     * LIST CHARACTER *
512                     ******************
513                     *
514                     * Entry: On stack: The address of the character to
515                     *                  be printed.
516                     * Exit:  BCDEHL preserved. F corrupted.
517                     *        A: Character printed.
518                     *
519 EFF5 E3          SCHRI   XTHL            Get addr from stack
520 EFF6 7E                  MOV   A,M       Get char
521 EFF7 23                  INX   H         HL pnts after char
522 EFF8 E3                  XTHL            Returnaddr on stack
523 EFF9 C360DD              JMP   :DD60     List char.
524                     *
525                     ********************************
526                     * LIST EXPRESSION; PRINT SPACE *
527                     ********************************
528                     *
529 EFFC C368CE        SEXPS   JMP   :CE68     List expr, print space
530                     *
531 EFFF 3E                    DATA  :3E
532                     *
533                     *
534                     *
535 F000                      END
```

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

L0E300 EEB4   L0E301 EEDF   L0E302 EEED   L0E303 EEF2
L0E305 EEFC   L0E306 EEFE   L0E307 EF38   L0E308 EF3C
L0E309 EF48   L0E310 EF55   L0E312 EF61   L0E313 EF6B
L0E314 EF77   L0E318 EFAE   L0E350 EE15   MPT02  EE8D
MPT17  EE0F   RLA20  EE4C   RTK40  EE94   S2510  EE55
S2710  EE79   S2720  EE7C   SC010  EFA3   SC020  EFA8
SC22A  EE25   SCARN  EEF7   SCEXP  EEA2   SCFPT  EFD4
SCHEX  EFDA   SCHRI  EFF5   SCINT  EFBD   SCN21  EE09
SCN22  EE1A   SCN23  EE30   SCN24  EE4F   SCN25  EE52
SCN26  EE66   SCN27  EE71   SCN28  EE87   SCON   EF84
SEXPS  EFFC   SFUN   EF5A   SLN10  EFB4   SQTS   EFE1
SSS10  EFCE   SSSPC  EFC0   SUQTS  EFED
```