```
002                             ORG   :DEB5
003                     *
004                     *
005                     *
006                     *  ==================================
007                     *** BASIC EXECUTION / RUN-TIME MODULE ***
008                     *  ==================================
009                     *
010                     * Generally, BC is used as entry pointer to the
011                     * textbuffer.
012                     *
013                     ********************
014                     * RUN basiccmd NEW *
015                     ********************
016                     *
017                     * Sets up heap, empty textbuffer and symboltable,
018                     * sets pointers of textbuffer and symboltable
019                     * correctly. Old buffer contents is not destroyed,
020                     * (except 4 locations), but not useable.
021                     * Valid as direct command only.
022                     *
023                     * Exit: A=1, CY=1.
024                     *
025 DEB5 00             RNEW    NOP
026 DEB6 00                     NOP
027 DEB7 00                     NOP
028 DEB8 2A9B02                 LHLD  :029B     Get startaddr heap
029 DEBB 229F02                 SHLD  :029F     Set start textbuf=start heap
030 DEBE 3600                   MVI   M,:00     Store 00 in 1st addr.
031 DEC0 23                     INX   H
032 DEC1 22A102                 SHLD  :02A1     Set start symtab
033 DEC4 3600                   MVI   M,:00     00 in 1st addr.
034 DEC6 23                     INX   H
035 DEC7 22A302                 SHLD  :02A3     Set end symtab
036
037                     * Entry from scratch/edit:
038
039 DECA 2A9D02         HRINIT  LHLD  :029D     Get heap size
040 DECD EB                     XCHG            in DE
041 DECE CD95D1                 CALL  :D195     Init heap to all available
042 DED1 3E01                   MVI   A,:01     Code for 'buffer crunched'
043 DED3 37                     STC
044 DED4 C9                     RET
045                     *
046                     ********************
047                     * RUN basiccmd CONT *
048                     ********************
049                     *
050                     * Resets step flag, decr. CONFL (error if it was
051                     * 0), restores FRAME from stack, restores stack-
052                     * pointer and continues program execution.
053                     * Valid as direct command only.
054                     *
055 DED5 AF             RCONT   XRA   A
056 DED6 321601         RCN10   STA   :0116     Reset step flag
057 DED9 212601                 LXI   H,:0126   Get pntr suspended program
058 DEDC 35                     DCR   M         Update it
059 DEDD 3E19                   MVI   A,:19
060 DEDF FAF5D9                 JM    :D9F5     Evt. run error 'CAN'T CONT'
061 DEE2 2A2701                 LHLD  :0127     Get current base stack level
062 DEE5 EB                     XCHG            in DE
063 DEE6 211500                 LXI   H,:0015   FRAME length
```

```
064 DEE9 19                     DAD   D          Top of FRAME in stack
065 DEEA E5                     PUSH  H          Save it
066 DEEB 010001                 LXI   B,:0100    Addr SYSBOT
067 DEEE CD4FDE                 CALL  :DE4F      Load FRAME from stack
068 DEF1 E1                     POP   H          Get addr FRAME top in stack
069 DEF2 222701                 SHLD  :0127      Update current base stack
070                                              level
071 DEF5 F9                     SPHL             Update stackpointer
072 DEF6 2A0201                 LHLD  :0102      Get start current command
073 DEF9 44                     MOV   B,H
074 DEFA 4D                     MOV   C,L        Store it in BC
075 DEFB C37FC8                 JMP   :C87F      Run BASIC line
076                     *
077                     **********************
078                     * RUN basiccmd STEP *
079                     **********************
080                     *
081 DEFE 3EFF           RSTEP   MVI   A,:FF      Init value STEP flag
082 DF00 C3D6DE                JMP   :DED6      Set STEP flag and continu
083                     *
084                     **********************
085                     * RUN basiccmd STOP *
086                     **********************
087                     *
088                     * Entry: No conditions.
089                     * Exit:  A=03, CY=1. HL points after string.
090                     *
091 DF03 CDFFDA         RSTOP   CALL  :DAFF      Print 'STOPPED'
092 DF06 C9DB                   DBL   :DBC9
093 DF08 3E03                  MVI   A,:03      Code for 'susp. execution'
094 DF0A 37                     STC
095 DF0B C9                     RET
096                     *
097                     **********************
098                     * RUN basiccmd END *
099                     **********************
100                     *
101                     * Entry: No conditions.
102                     * Exit:  A=01, CY=1. HL points after string.
103                     *
104 DF0C CDFFDA         REND    CALL  :DAFF      Print 'END PROGRAM'
105 DF0F CFDB                   DBL   :DBCF
106 DF11 3E01                  MVI   A,:01      Code for 'stop execution'
107 DF13 37                     STC
108 DF14 C9                     RET
109                     *
110                     ********************
111                     * RUN basiccmd IF *
112                     ********************
113                     *
114                     * Used for IF .. GOTO <linenr> and for
115                     * IF .. THEN <linenr>.
116                     *
117                     RIFG
118 DF15 CD63E7         RIFTL   CALL  :E763      (0) Run logical expression
119 DF18 3C                     INR   A
120 DF19 CA63DF                 JZ    :DF63      Run GOTO if condition true
121
122                     * If condition false:
123
124 DF1C 03                      INX   B
125 DF1D 03                      INX   B          Skip linenr
```

```
126 DF1E B7                     ORA   A         No special action
127 DF1F C9                     RET
128                     *
129                     ****************************************
130                     * RUN basiccmd IF .. THEN <statement> *
131                     ****************************************
132                     *
133 DF20 CD63E7         RIFTC   CALL  :E763     (O) Run logical expression
134 DF23 3C                     INR   A
135 DF24 C28FE1                 JNZ   :E18F     (O) Ignore rest if false
136
137                     * If condition true:
138
139 DF27 03                     INX   B         Skip length line
140 DF28 B7                     ORA   A         No special action
141 DF29 C9                     RET             Execute rest of line
142                     *
143                     ******************
144                     * basiccmd GOSUB *
145                     ******************
146                     *
147                     * Saves current program state on the interpreter
148                     * stack and branches to a named line. The running
149                     * FOR loop (if any) is saved in order to avoid
150                     * problems if any unpaired NEXT is encountered.
151                     * The stackpointer at the subroutine-entry is held
152                     * to enable breaking out of a FOR-NEXT loop.
153                     *
154 DF2A CDEDE6         RGOSUB  CALL  :E6ED     (O) Get linenr and find it
155
156                     * Entry from ON - GOSUB:
157
158 DF2D D1             RGS10   POP   D         Kill return addr
159 DF2E 221901                 SHLD  :0119     Save new PC
160 DF31 CD3CE1                 CALL  :E13C     (O) Save FOR loop contents
161 DF34 C5                     PUSH  B         Save program position
162 DF35 2A1901                 LHLD  :0119     Get new PC
163 DF38 44                     MOV   B,H       ) Is new text position
164 DF39 4D                     MOV   C,L       )
165 DF3A 2A1301                 LHLD  :0113     Get stack level last GOSUB
166 DF3D E5                     PUSH  H         Save evt link to previous
167                                             subroutine entry
168 DF3E 210000                 LXI   H,:0000
169 DF41 220401                 SHLD  :0104     No running loop
170 DF44 39                     DAD   SP        SP in HL
171 DF45 221301                 SHLD  :0113     SP in STKGOS for return
172
173                     * Entry on return:
174
175 DF48 B7             RGS20   ORA   A         No special action
176 DF49 C38FC8                 JMP   :C88F     Into Basic monitor
177                     *
178                     ************************
179                     * RUN basiccmd RETURN *
180                     ************************
181                     *
182                     * Reverses the effect of a previous 'GOSUB'.
183                     *
184 DF4C D1             RRET    POP   D         Kill returnaddr
185 DF4D 2A1301                 LHLD  :0113     Get stack level last GOSUB
186 DF50 7C                     MOV   A,H
187 DF51 B5                     ORA   L         0 if no active call
```

```
188 DF52 3E01              MVI   A,:01
189 DF54 CAF5D9            JZ    :D9F5      Then run error 'RETURN
190                                         WITHOUT GOSUB'
191 DF57 F9                SPHL             Else: re-instate old stack
192 DF58 E1                POP   H
193 DF59 221301            SHLD  :0113      Link back to previous
194                                         GOSUB
195 DF5C C1                POP   B          Restore text pntr
196 DF5D CD67E1            CALL  :E167      (0) Pop FRAME
197 DF60 C348DF            JMP   :DF48      Back to Basic monitor
198                *
199                **********************
200                * RUN basiccmd GOTO *
201                **********************
202                *
203                * Simply transfers control to a named line.
204                *
205                * Entry from IF - GOTO:
206                *
207 DF63 CDEDE6    RGOTO   CALL  :E6ED      (0) Get linenr and find it
208
209                * Entry from ON - GOTO:
210
211 DF66 44          RGT10   MOV   B,H      ) Set textpointer to line
212 DF67 4D                  MOV   C,L      )
213 DF68 B7                  ORA   A        No special action
214 DF69 C9                  RET
215                *
216                ****************************
217                * RUN basiccmd ON .. GOTO *
218                ****************************
219                *
220 DF6A CD78DF    RONGT   CALL  :DF78      Process command
221 DF6D DA66DF            JC    :DF66      Use GOTO if OK
222 DF70 C9                RET              If outside list
223                *
224                ****************************
225                * RUN basiccmd ON .. GOSUB *
226                ****************************
227                *
228 DF71 CD78DF    RONGS   CALL  :DF78      Process command
229 DF74 DA2DDF            JC    :DF2D      Use GOSUB if OK
230 DF77 C9                RET              If outside list
231                *
232                ************************
233                * COMMON 'ON ..' CODE *
234                ************************
235                *
236                * Exit: CY=1: OK.
237                *       CY=0: Outside list.
238                *
239 DF78 CD1DE7    RONFN   CALL  :E71D      ) (0) Get index of number
240 DF7B 5F                MOV   E,A        ) in list in E
241 DF7C 0A                LDAX  B          Get nr of linenrs in list
242 DF7D 03                INX   B
243 DF7E 6F                MOV   L,A
244 DF7F 2600              MVI   H,:00
245 DF81 29                DAD   H
246 DF82 09                DAD   B          Pointer after list
247 DF83 E5                PUSH  H          Save pointer
248 DF84 1D                DCR   E
249 DF85 1C                INR   E
```

```
250 DF86 CA9BDF             JZ    :DF9B     Index of 0: outside list
251 DF89 BB                 CMP   E
252 DF8A DA9BDF             JC    :DF9B     If index too large
253 DF8D 1600               MVI   D,:00
254 DF8F 1B                 DCX   D
255 DF90 EB                 XCHG
256 DF91 29                 DAD   H
257 DF92 09                 DAD   B         Pntr to reqd linenr
258 DF93 44                 MOV   B,H       ) in BC
259 DF94 4D                 MOV   C,L       )
260 DF95 CDEDE6             CALL  :E6ED     (0) Find reqd line
261 DF98 C1                 POP   B
262 DF99 37                 STC             CY=1: OK
263 DF9A C9                 RET
264
265                 * If outside list:
266
267 DF9B C1         ROF10   POP   B
268 DF9C B7         ROF11   ORA   A         CY=0: outside list
269 DF9D C9                 RET
270                 *
271                 **********************
272                 * RUN basiccmd RUN *
273                 **********************
274                 *
275                 * No linenumber is given.
276                 *
277 DF9E 210000     RRUN    LXI   H,:0000
278 DFA1 221501             SHLD  :0115     Reset trace/step flag
279 DFA4 2A9F02             LHLD  :029F     Get start textbuf
280 DFA7 44         RRN10   MOV   B,H       )
281 DFA8 4D                 MOV   C,L       ) Store it in BC
282 DFA9 CD01E4             CALL  :E401     (0) Run RESTORE; Set data
283                                          pntr to start program
284 DFAC 3100F9             LXI   SP,:F900  Reset stackpointer
285 DFAF AF                 XRA   A
286 DFB0 322601             STA   :0126     No suspended program
287 DFB3 CD23CB             CALL  :CB23     Empty HEAP + symtab
288 DFB6 B7                 ORA   A         No special action
289 DFB7 C38FC8             JMP   :C88F     Run program
290                 *
291                 ******************************
292                 * RUN basiccmd RUN <linenr> *
293                 ******************************
294                 *
295                 * After RUN, a line number is given.
296                 *
297 DFBA CDEDE6     RRUNN   CALL  :E6ED     (0) Read linenr and find
298                                          it in textbuf
299 DFBD C3A7DF             JMP   :DFA7     Process RUN
300                 *
301                 **********************
302                 * RUN basiccmd POKE *
303                 **********************
304                 *
305 DFC0 CDF8E6     RPOKE   CALL  :E6F8     (0) Get addr in HL
306
307                 * Entry for other modules:
308
309 DFC3 CD1DE7     RPEN    CALL  :E71D     (0) Get argument in A
310 DFC6 77                 MOV   M,A       Store it
311 DFC7 B7                 ORA   A         No special action
```

```
312 DFC8 C9                 RET
313                 *
314                 **********************
315                 * RUN basiccmd OUT *
316                 **********************
317                 *
318 DFC9 CD1DE7       ROUT    CALL  :E71D     (0) Get portnr in A
319 DFCC 57                 MOV   D,A       and in D
320 DFCD CD1DE7             CALL  :E71D     (0) Get data in A
321 DFD0 5F                 MOV   E,A       and in E
322 DFD1 B7                 ORA   A
323 DFD2 C3C8D8             JMP   :D8C8     Output to DCE-bus
324                 *
325                 ************************
326                 * RUN basiccmd WAIT *
327                 ************************
328                 *
329                 * WAIT I,J,K reads the status of Real World port
330                 * I, EXOR's it with K and AND's it with J until
331                 * a result equal to J is obtained.
332                 *
333 DFD5 CD1DE7       RWAIT   CALL  :E71D     (0) Get portnr
334 DFD8 57                 MOV   D,A       in D
335 DFD9 CD1DE7             CALL  :E71D     (0) Get bits needed high
336 DFDC 67                 MOV   H,A       in H
337 DFDD 0A                 LDAX  B         Get next byte from text
338 DFDE 03                 INX   B
339 DFDF D6FF               SUI   :FF       Check if ony 2 arguments
340 DFE1 00                 NOP
341 DFE2 C4DACE             CNZ   :CEDA     If 3 arg: Get XOR mask
342 DFE5 6F                 MOV   L,A       in L
343 DFE6 CDE0D8     RWT20   CALL  :D8E0     Input from DCE-bus
344 DFE9 7B                 MOV   A,E       into A
345 DFEA AD                 XRA   L         XOR with mask
346 DFEB A4                 ANA   H         AND with bits needed high
347 DFEC BC                 CMP   H         Correct value reached?
348 DFED C8                 RZ              Then abort
349 DFEE CDA5D6             CALL  :D6A5     Check keyb for new inputs
350 DFF1 D2E6DF             JNC   :DFE6     Next DCE-input if no Break
351
352                 * If suspended:
353
354 DFF4 C312E0             JMP   :E012     (0) Quit: 'cmd broken in'
355                 *
356                 ***************************
357                 * RUN basiccmd WAIT MEM *
358                 ***************************
359                 *
360                 * As WAIT, but with I is a memory location.
361                 *
362 DFF7 CDF8E6       RWTEM   CALL  :E6F8     (0) Get memory addr in HL
363 DFFA CD1DE7             CALL  :E71D     (0) Get bit mask
364 DFFD 57                 MOV   D,A       in D
365 DFFE 0A                 LDAX  B         Get next byte from text
366 DFFF 03                 INX   B
367 E000 D6FF               SUI   :FF       Check if only 2 arguments
368 E002 00                 NOP
369 E003 C4DACE             CNZ   :CEDA     If 3 arg: Get XOR mask
370 E006 5F                 MOV   E,A       in E
371 E007 7B         RWM20   MOV   A,E       XOR mask in A
372 E008 AE                 XRA   M         XOR with memory
373 E009 A2                 ANA   D         AND with bit mask
```

```
374 E00A BA                     CMP   D          Correct value reached?
375 E00B C8                     RZ               Then abort
376 E00C CDA5D6                 CALL  :D6A5      Check keyb for inputs
377 E00F D207E0                 JNC   :E007      Cont if no Break pressed
378
379                     * If suspended:
380
381 E012 3E02           RWT30   MVI   A,:02      Code 'command broken in'
382 E014 37                     STC
383 E015 C9                     RET
384                     *
385                     *
386                     *
387 E016                        END
```

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

HRINIT	DECA	RCN10	DED6	RCONT	DED5	REND	DF0C
RGOSUB	DF2A	RGOTO	DF63	RGS10	DF2D	RGS20	DF48
RGT10	DF66	RIFG	DF15	RIFTC	DF20	RIFTL	DF15
RNEW	DEB5	ROF10	DF9B	ROF11	DF9C	RONFN	DF78
RONGS	DF71	RONGT	DF6A	ROUT	DFC9	RPEN	DFC3
RPOKE	DFC0	RRET	DF4C	RRN10	DFA7	RRUN	DF9E
RRUNN	DFBA	RSTEP	DEFE	RSTOP	DF03	RWAIT	DFD5
RWM20	E007	RWT20	DFE6	RWT30	E012	RWTEM	DFF7

```
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
```

```
                        **********************
                        * RUN basiccmd LIST *
                        **********************
                        *
                        * The whole textbuffer contents is listed.
                        *
                        RLIST
E197 3EFF               RLIS0   MVI   A,:FF     Init. mode 0
E199 EF                         RST   5         Change mode
E19A 18                         DATA  :18
E19B 3E0C                       MVI   A,:0C
E19D EF                         RST   5         Clear screen
E19E 03                         DATA  :03
E19F 2A9F02             RLS05   LHLD  :029F     Get startaddr. textbuf
E1A2 EB                         XCHG            in DE
E1A3 2AA102                     LHLD  :02A1     Get start symtab
E1A6 2B                         DCX   H         End textbuf in HL
E1A7 C3CFE1                     JMP   :E1CF     Perform listing
                        *
                        ************************************
                        * RUN basiccmd LIST <linenumber> *
                        ************************************
                        *
                        * Entry: BC points to linenumber.
                        * Exit:  BC updated. AFDEHL corrupted.
                        *
E1AA CDE7E6             RLIS1   CALL  :E6E7     Read linenr and find
                                                it in textbuf
E1AD 00                         NOP
E1AE 54                         MOV   D,H       ) Linenr in DE
E1AF 5D                         MOV   E,L       )
E1B0 DC39DE                     CC    :DE39     If linenr found: calc addr
                                                after string in HL
E1B3 C3CFE1                     JMP   :E1CF     Perform listing
                        *
                        ******************************
                        * RUN basiccmd LIST <range> *
                        ******************************
                        *
                        * Entry: BC points to 1st linenumber.
                        * Exit:  BC updated. AFDEHL corrupted.
                        *
E1B6 2A9F02             RLIS2   LHLD  :029F     Get start textbuf
E1B9 CDD9E6                     CALL  :E6D9     Read 1st linenr
E1BC C4F6CA                     CNZ   :CAF6     If given: Find it in textbuf
E1BF EB                         XCHG            Addr in DE
E1C0 2AA102                     LHLD  :02A1     Get start symtab
E1C3 2B                         DCX   H         End textbuf in HL
E1C4 CDD9E6                     CALL  :E6D9     Read 1st linenr
E1C7 37                         STC
E1C8 3F                         CMC
E1C9 C4F6CA                     CNZ   :CAF6     If 1st nr found: find 2nd
E1CC DC39DE                     CC    :DE39     If found: Calc addr after
                                                string in HL


                        * Perform listing:

E1CF C5                 RLS10   PUSH  B
E1D0 42                         MOV   B,D       ) Start listed area in BC
E1D1 4B                         MOV   C,E       )
E1D2 221B01                     SHLD  :011B     Store end listed area
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
380                                          correct
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:
386                                          Wait for spacebar
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
```

```
                        ****************************
                        * INPUT FROM EDIT BUFFER *
                        ****************************
                        *
                        * Entry: A=0, CY=0.
                        *
E291 F5         IFBNL   PUSH  PSW
E292 2AA200             LHLD  :00A2     Get startaddr. editbuf
E295 223201             SHLD  :0132     Store it in EFEPT
E298 7E         IFB10   MOV   A,M       Get char from editbuf
E299 B7                 ORA   A         Char is 0?
E29A CAAAE2             JZ    :E2AA     Then editbuf empty
E29D 23                 INX   H
E29E FE0D               CPI   :0D       Car.ret?
E2A0 C298E2             JNZ   :E298     Get next char if not
                        * If char is car.ret:
E2A3 22A200             SHLD  :00A2     Update startaddr. editbuf
E2A6 0E00               MVI   C,:00     1st pos on line
E2A8 F1                 POP   PSW       No special action
E2A9 C9                 RET
                        * If buffer empty:
E2AA 323501     IFB20   STA   :0135     Set input from keyboard
E2AD CDCADE             CALL  :DECA     Organise HEAP + buffers
E2B0 F1                 POP   PSW       special action
E2B1 37                 STC             CY=1
E2B2 C9                 RET
                        *
                        ************************
                        * RUN basiccmd PRINT *
                        ************************
                        *
                        * Entry: BC: Position in textbuffer.
                        *
E2B3 0A         RPRINT  LDAX  B         Get length
E2B4 03                 INX   B
E2B5 57                 MOV   D,A       Count of expr in D
E2B6 B7                 ORA   A
E2B7 CAF7E2             JZ    :E2F7     Jump if only car.ret
E2BA D5         RPR10   PUSH  D         Save nr of expressions
E2BB 0A                 LDAX  B         Get expr type
E2BC 03                 INX   B
E2BD FE20               CPI   :20
E2BF CAD2E2             JZ    :E2D2     Jump if string
                        * If INT/FPT number:
E2C2 FE00               CPI   :00
E2C4 CD08E8             CALL  :E808     Eval expr. Result in MACC
E2C7 F5                 PUSH  PSW       Save flags on expr.type
E2C8 C453DB             CNZ   :DB53     If INT: print INT number
E2CB F1                 POP   PSW
E2CC CC59DB             CZ    :DB59     If FPT: print FPT number
E2CF C3E3E2             JMP   :E2E3
                        * If string:
E2D2 CD91E7     RPR40   CALL  :E791     Evaluate string expr
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).
```

```
312                     *       HL: Startaddress data line.
313                     *
314 E361 CD36E4         LOE56   CALL  :E436     (not used)
315                     *
316 E364 0A             RRDIP   LDAX  B         Get nr of data reqd
317 E365 03                     INX   B
318 E366 57                     MOV   D,A       Into D
319 E367 15             RRI10   DCR   D
320 E368 FAC1E3                 JM    :E3C1     Jump if ready
321 E36B 1C             RRI20   INR   E         Offset nex char
322 E36C CAA8E3                 JZ    :E3A8     Jump if at end of line
323 E36F 1D                     DCR   E
324 E370 CD63E9         RRI30   CALL  :E963     Get varptr in HL
325 E373 C5                     PUSH  B
326 E374 D5                     PUSH  D
327 E375 4B                     MOV   C,E       Offset in C
328 E376 E5                     PUSH  H         Save varptr
329 E377 213E01                 LXI   H,:013E   Startaddr EBUF
330 E37A E5                     PUSH  H         Save it on stack
331 E37B E630                   ANI   :30
332 E37D CF                     RST   1         Encode a constant
333 E37E 06                     DATA  :06
334 E37F 59                     MOV   E,C       Offset in E
335 E380 C1                     POP   B
336 E381 E1                     POP   H         Get varptr
337 E382 FE20                   CPI   :20       String type ?
338 E384 7B                     MOV   A,E       Offset in A
339 E385 CAA2E3                 JZ    :E3A2     Jump if string type
340 E388 CDB4E4                 CALL  :E4B4     Perform LET (INT/FPT)
341
342                     * Check separator/terminator:
343
344 E38B 4F             RRI40   MOV   C,A       Offset in C
345 E38C CDD2DD                 CALL  :DDD2     Get char from line, neglect
346                                             tab and space
347 E38F 0C                     INR   C         Offset +1
348 E390 FE2C                   CPI   :2C       ',' ?
349 E392 CA99E3                 JZ    :E399
350 E395 FE0D                   CPI   :0D       CR ?
351 E397 0EFF                   MVI   C,:FF     Dummy offset for end of
352                                             line
353 E399 D1             RRI50   POP   D
354 E39A 59                     MOV   E,C       Offset in E
355 E39B C1                     POP   B
356 E39C C2C3E3                 JNZ   :E3C3     Error if char not ',' or
357                                             car.ret
358 E39F C367E3                 JMP   :E367     Read next data line
359
360                     * If string type:
361
362 E3A2 CDB8E4           RRI60   CALL  :E4B8     Perform LET (STR)
363 E3A5 C38BE3                   JMP   :E38B     Check terminator/separator
364
365                     * If end of line reached ('CR'):
366
367 E3A8 3A1701          RRI70   LDA   :0117     Get flag for running inputs
368 E3AB B7                     ORA   A
369 E3AC CABBE3                 JZ    :E3BB     Quit if not running inputs
370 E3AF CD55DD                 CALL  :DD55     Cursor to begin next line
371 E3B2 CDD0E3                 CALL  :E3D0     Print '?', read input line
372 E3B5 DAC2E3                 JC    :E3C2     Abort if break pressed
                                JMP   :E370     Cont reading
```

```
374
375                     * If whole line read:
376
377 E3BB CDDCE3       RRI80   CALL  :E3DC      Find next dataline in txtbuf
378 E3BE C370E3               JMP   :E370      Cont reading
379
380                     * If reading done:
381
382 E3C1 B7           RRI90   ORA   A          No special action
383 E3C2 C9           RRI99   RET
384
385                     * If error:
386
387 E3C3 2A3201       RRISN   LHLD  :0132      Get EFEPT
388 E3C6 11FCFF             LXI   D,:FFFC
389 E3C9 19                 DAD   D            EFEPT-4
390 E3CA 220001             SHLD  :0100        Store start current line
391 E3CD C30BDA             JMP   :DA0B        Run 'SYNTAX ERROR'
392                     *
393                     ****************************
394                     * PREPARE GETTING INPUTS *
395                     ****************************
396                     *
397                     * Continuation of 0E447.
398                     * Prints a prompt ('?') on the screen and gets a
399                     * textline from input.
400                     *
401 E3D0 D5           INPGT   PUSH  D
402 E3D1 EF                   RST   5            Ask cursor pos. and size
403 E3D2 0C                   DATA  :0C          char. screen
404 E3D3 D1                   POP   D
405 E3D4 3E3F                 MVI   A,:3F
406 E3D6 CDA0C6               CALL  :C6A0        Print '?'; input a textline
407 E3D9 5D                   MOV   E,L
408 E3DA 1C                   INR   E            E pnts after last char of
409                                               input
410 E3DB C9                   RET
411                     *
412                     ************************************
413                     * FIND NEXT DATALINE IN TEXTBUFFER *
414                     ************************************
415                     *
416 E3DC F5           DATAF   PUSH  PSW
417 E3DD D5                   PUSH  D
418 E3DE E5                   PUSH  H
419 E3DF 2A2401               LHLD  :0124        Get addr current dataline
420 E3E2 7E           DTF10   MOV   A,M          Get length data string
421 E3E3 B7                   ORA   A
422 E3E4 3E02                 MVI   A,:02
423 E3E6 CAF5D9               JZ    :D9F5        Run error 'OUT OF DATA' if
424                                               no data available
425 E3E9 54                   MOV   D,H          ) Addr dataline in DE
426 E3EA 5D                   MOV   E,L          )
427 E3EB CD39DE               CALL  :DE39        Calc end dataline
428 E3EE 222401               SHLD  :0124        Store it in DATAP
429 E3F1 EB                   XCHG               End in DE, start in HL
430 E3F2 23                   INX   H
431 E3F3 23                   INX   H
432 E3F4 23                   INX   H            Pnts to token
433 E3F5 7E                   MOV   A,M          Get token
434 E3F6 FEA2                 CPI   :A2          Is it DATA?
435 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   LOE351 E34A
LOE56  E361   RED05  E1FB   RED10  E20D   RED19  E21B
RED20  E21F   RED30  E24D   REDI1  E253   REDI2  E25C
REDIN  E265   REDIT  E1F5   RINPQ  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 DATAQ=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                     *
```

```
064                     ******************
065                     * cont. of 0E3DC *
066                     ******************
067                     *
068 E436 23             MPT36   INX   H
069 E437 7E                     MOV   A,M       Get length dataline
070 E438 323401                 STA   :0134     Store it in EFECT
071 E43B 23                     INX   H         Pnts to start data bytes
072 E43C 223201                 SHLD  :0132     Startaddr string in EFEPT
073 E43F 00                     NOP
074 E440 00                     NOP
075 E441 E1                     POP   H
076 E442 D1                     POP   D
077 E443 1E00                   MVI   E,:00
078 E445 F1                     POP   PSW
079 E446 C9                     RET
080                     *
081                     ****************************
082                     * PREPARE GETTING INPUTS *
083                     ****************************
084                     *
085 E447 221D01         MPT5A   SHLD  :011D     Store SP+2 in ERSSP
086 E44A 3EFF                   MVI   A,:FF
087 E44C 321701                 STA   :0117     Set flag for running inputs
088 E44F C3D0E3                 JMP   :E3D0     Print '?', input a textline
089                     *
090                     ******************
091                     * cont. of 0E401 *
092                     ******************
093                     *
094 E452 322301         MPT32   STA   :0123     Save offset next char to
095 E455 C9                     RET             encode
096                     *
097 E456 FF                     DATA  :FF
098 E457 FF                     DATA  :FF
099 E458 FF                     DATA  :FF
100 E459 FF                     DATA  :FF
101                     *
102                     ********************
103                     * RUN basiccmd LET *
104                     ********************
105                     *
106                     * Valid as direct command or in program. Computes
107                     * variable pointer or variable reference and
108                     * checks type. 'LET' can be explicitly or
109                     * implicitly given.
110                     *
111                     * Entry: BC: Points to program line.
112                     * Exit:  BC: Updated.
113                     *        A:  Type of variable.
114                     *        FDEHL corrupted.
115                     *
116                     RLETX
117 E45A CD63E9         RLETI   CALL  :E963     Get varptr in HL, T/L in A
118 E45D E630                   ANI   :30       Type only
119 E45F F5                     PUSH  PSW       Save type
120 E460 FE20                   CPI   :20       String type?
121 E462 C294E4                 JNZ   :E494     Jump if INT/FPT
122
123                     * If string type:
124
125 E465 E5             RLT10   PUSH  H         Save varptr
```

```
126 E466 7E                 MOV   A,M       )
127 E467 23                 INX   H         )
128 E468 66                 MOV   H,M       ) Addr string in HL
129 E469 6F                 MOV   L,A
130 E46A E3                 XTHL            ) Save stringaddr
131 E46B E5                 PUSH  H         ) and varptr
132 E46C CD9DE7             CALL  :E79D     Get value being assigned
133 E46F 7B                 MOV   A,E       Get status
134 E470 FE02               CPI   :02
135 E472 CA7FE4             JZ    :E47F     Jump if temp on heap
136 E475 7E                 MOV   A,M       Get string length
137 E476 EB                 XCHG
138 E477 CD8BD1             CALL  :D18B     Get place in heap for string
139 E47A E5                 PUSH  H
140 E47B CD72D1             CALL  :D172     Transfer string into heap
141 E47E E1                 POP   H
142 E47F EB         RLT20   XCHG            Pntr to string in DE
143 E480 E1                 POP   H         Pntr to variable in HL
144 E481 73                 MOV   M,E       ) Stringpntr in variable
145 E482 23                 INX   H         )
146 E483 72                 MOV   M,D       )
147
148                 * Entry from scratch:
149
150 E484 2A9F02     RLT25   LHLD  :029F     Get startaddr. textbuf
151 E487 EB                 XCHG            in DE
152 E488 E1                 POP   H         Get pntr old value
153 E489 7C                 MOV   A,H
154 E48A B5                 ORA   L
155 E48B C414DE             CNZ   :DE14     If <>0: Test if on heap
156 E48E DC87D1             CC    :D187     Then clear string in heap
157 E491 C3B2E4             JMP   :E4B2     Ready
158
159                 * If FPT/INT type:
160
161 E494 E5           RLT30   PUSH  H
162 E495 CD19E8             CALL  :E819     Eval numeric arguments
163 E498 7C                 MOV   A,H
164 E499 B5                 ORA   L
165 E49A C2A3E4             JNZ   :E4A3     Copy num expr if not in MACC
166 E49D E1                 POP   H
167 E49E E7                 RST   4         Copy MACC to variable
168 E49F 0F                 DATA  :0F
169 E4A0 C3B2E4             JMP   :E4B2     Ready
170
171                 * If just constant or variable:
172
173 E4A3 D1           RLT50   POP   D         Get pntr to variable
174 E4A4 D5                 PUSH  D
175 E4A5 C5                 PUSH  B
176 E4A6 0604               MVI   B,:04     Nr of bytes
177 E4A8 7E         RLT60   MOV   A,M       Get byte
178 E4A9 12                 STAX  D         Store it in variable
179 E4AA 23                 INX   H
180 E4AB 13                 INX   D
181 E4AC 05                 DCR   B         Decr byte count
182 E4AD C2A8E4             JNZ   :E4A8     Next byte if not ready
183 E4B0 C1                 POP   B
184 E4B1 E1                 POP   H
185
186                 * Ready:
187
```

```
188 E4B2 F1          RLT90   POP   PSW
189 E4B3 C9                  RET
190
191                  * Entry from READ/INPUT for FPT/INT:
192
193 E4B4 F5          RLT70   PUSH  PSW
194 E4B5 C394E4              JMP   :E494     Value to variable
195
196                  * Entry from READ/INPUT for STR:
197
198 E4B8 F5          RLT80   PUSH  PSW
199 E4B9 C365E4              JMP   :E465     Stringvalue to variable
200                  *
201                  **********************
202                  * RUN basiccmd SOUND *
203                  **********************
204                  *
205                  * Formats: SOUND <CHAN><ENV><VOL><TG><FREQ>.
206                  *          SOUND <CHAN> OFF.
207                  *          SOUND OFF.
208                  *
209                  * Entry: BC: Points to program line.
210                  * Exit:  BC updated, AFDEHL corrupted.
211                  *
212 E4BC 0A          RSOUND  LDAX  B         Get 1st expr.
213 E4BD FEFF                CPI   :FF       Is it sound OFF?
214 E4BF CA01E5              JZ    :E501     Then turn all sound off
215 E4C2 3E02                MVI   A,:02
216 E4C4 CD43E7              CALL  :E743     Get channelnr (0,1,2)
217 E4C7 21C201              LXI   H,:01C2   Startaddr SCB0
218 E4CA 110E00              LXI   D,:000E   Length SCB
219 E4CD F5                  PUSH  PSW       Save channelnr
220 E4CE 3D          SND10   DCR   A
221 E4CF FAD6E4              JM    :E4D6     If chan.0 or ready
222 E4D2 19                  DAD   D         Absolute addr SCB in HL
223 E4D3 C3CEE4              JMP   :E4CE     If chan.2
224 E4D6 D1          SND20   POP   D         Channelnr in D
225 E4D7 CD1FE5              CALL  :E51F     Set up SCB <ENV><VOL>
226 E4DA DAFCE4              JC    :E4FC     If channel to be OFF
227 E4DD 3E03                MVI   A,:03
228 E4DF CD43E7              CALL  :E743     Get <TG>
229 E4E2 F5                  PUSH  PSW
230 E4E3 E601                ANI   :01       Tremoloflag only
231 E4E5 77                  MOV   M,A       Into SCB
232 E4E6 23                  INX   H
233 E4E7 00                  NOP
234 E4E8 00                  NOP
235 E4E9 F1                  POP   PSW       Restore <TG>
236 E4EA E602                ANI   :02       Glissandoflag only
237 E4EC 1F                  RAR
238 E4ED 3C                  INR   A
239 E4EE 23                  INX   H         1 free byte
240 E4EF 77                  MOV   M,A       Glissandoflag in SCB
241 E4F0 23                  INX   H         )
242 E4F1 23                  INX   H         ) Ignore current period
243 E4F2 23                  INX   H         )
244 E4F3 E5                  PUSH  H         Save pntr to reqd period
245 E4F4 CDF8E6              CALL  :E6F8     Get <period>
246 E4F7 EB                  XCHG            in DE
247 E4F8 E1                  POP   H
248 E4F9 73                  MOV   M,E       ) Reqd period in SCB
249 E4FA 23                  INX   H         )
```

```
250 E4FB 72                 MOV   M,D        )
251 E4FC CD96D9       SND70 CALL  :D996      Enable sound interrupts
252 E4FF B7                 ORA   A          No special action
253 E500 C9                 RET
254
255                   * If SOUND OFF:
256
257 E501 03           SND80 INX   B
258 E502 CD8FD9             CALL  :D98F      Disable sound interrupts
259 E505 C5                 PUSH  B
260 E506 CDA6D8             CALL  :D8A6      Stop oscillators
261 E509 C1                 POP   B
262 E50A B7                 ORA   A          No special action
263 E50B C9                 RET
264                   *
265                   **********************
266                   * RUN basiccmd NOISE *
267                   **********************
268                   *
269                   * Sets up a noise control block.
270                   * Formats: NOISE <ENV><VOL>.
271                   *          NOISE OFF.
272                   *
273                   * Entry: BC: Points to expression.
274                   * Exit:  Noise off: BC updated, DE preserved,
275                   *                   AFHL corrupted.
276                   *        Noise on:  BC updated, AFDEHL corrupted.
277                   *
278 E50C 21EC01        RNOISE LXI   H,:01EC   Startaddr NCB
279 E50F 1603                 MVI   D,:03     Channelnr.3
280 E511 CD1FE5               CALL  :E51F     Set up NCB
281 E514 DAFCE4               JC    :E4FC     If channel to be off
282 E517 3600                 MVI   M,:00     No tremolo
283 E519 23                   INX   H
284 E51A 3600                 MVI   M,:00     Current volume = 0
285 E51C C3FCE4               JMP   :E4FC     Enable sound interrupts
286                   *
287                   ****************
288                   * SET ENVELOPE *
289                   ****************
290                   *
291                   * Sets up a sound or noise control block for
292                   * a channel.
293                   *
294                   * Entry: D:  Channelnumber (0,1,2 or 3).
295                   *        BC: Points to programline.
296                   *        HL: Points to startaddr SCB/NCB.
297                   * Exit:  If sound/noise off (CY=1):
298                   *        FF in 1st byte of SCB/NCB.
299                   *            HL: points to 1st byte SCB/NCB.
300                   *            BC: points to 2nd byte SCB/NCB.
301                   *            DE preserved, AF corrupted.
302                   *        If sound/noise on (CY=0):
303                   *        00 in 1st byte SCB/NCB.
304                   *            HL: Points after free byte.
305                   *            DE: Envelopeaddr +1.
306                   *            BC: Points beyond volume.
307                   *            AF: corrupted.
308                   *
309 E51F CD8FD9        SNGEV  CALL  :D98F      Disable sound interrupts
310 E522 3600                 MVI   M,:00     00 in 1st byte SCB/NCB
311 E524 0A                   LDAX  B          Get 1st byte from progr.line
```

```
312 E525 FEFF              CPI   :FF
313 E527 CA52E5            JZ    :E552     Jump if channel to be off
314
315                * If channel on:
316
317 E52A 23                INX   H         Pntr to next byte of block
318 E52B E5                PUSH  H         Save pntr
319 E52C 3E01              MVI   A,:01
320 E52E CD43E7            CALL  :E743     Get envelopenr in A (0,1)
321 E531 21F501            LXI   H,:01F5   Addr envelope area
322 E534 0F                RRC
323 E535 0F                RRC
324 E536 5F                MOV   E,A       DE=64*A
325 E537 1600              MVI   D,:00
326 E539 19                DAD   D         Add offset for env area
327 E53A EB                XCHG            Pntr to envelope in DE
328 E53B E1                POP   H         Get input pntr SCB/NCB
329 E53C 73                MOV   M,E       ) Envelope addr in block
330 E53D 23                INX   H         )
331 E53E 72                MOV   M,D       )
332 E53F 23                INX   H
333 E540 13                INX   D
334 E541 73                MOV   M,E       ) Env addr +1 in block
335 E542 23                INX   H         )
336 E543 72                MOV   M,D       )
337 E544 23                INX   H
338 E545 3E0F              MVI   A,:0F
339 E547 CD43E7            CALL  :E743     Get volume multiplier in
340                                        A (0-15)
341 E54A 07                RLC             )
342 E54B 07                RLC             ) *8
343 E54C 07                RLC             )
344 E54D 77                MOV   M,A       Vol.*8 in block
345 E54E 23                INX   H         Pntr to basic volume
346 E54F 23                INX   H         Pntr to tremoloflag
347 E550 B7                ORA   A         Return 'channel on'
348 E551 C9                RET
349
350                * If channel to be off:
351
352 E552 03       SNG10    INX   B
353 E553 35                DCR   M         FF in 1st byte SCB/NCB
354 E554 7A                MOV   A,D       Get channelnr
355 E555 FE03              CPI   :03
356 E557 CA63E5            JZ    :E563     Jump if noisechannel
357 E55A 0F                RRC             ) Find disable data for
358 E55B 0F                RRC             ) chan. 0-2
359 E55C F636              ORI   :36       )
360 E55E 3206FC            STA   :FC06     Load sound cmd word
361 E561 37                STC             Return 'channel off'
362 E562 C9                RET
363
364                * If noise to be off:
365
366 E563 3A9502   SNG20    LDA   :0295     Get volume osc.2 + noise
367 E566 E60F              ANI   :0F       Vol. noise = 0
368 E568 329502            STA   :0295     Update POR1M
369 E56B 3205FD            STA   :FD05     and POR1
370 E56E 37                STC             Return 'channel off'
371 E56F C9                RET
372                *
373                *
```

```
                        **************************
                        * 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 CAB0E4                     JZ    :E4B0     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  *
                        **********************
```

```
436                     *
437                     * Entry: BC: Points to program line.
438                     *
439 E5BB 0A             RMODE   LDAX  B         Get reqd mode in A
440 E5BC 03                     INX   B
441 E5BD C3B5CE                 JMP   :CEB5     Change screen mode
442                     *
443 E5C0 C9             LOE99   RET
444                     *
445                     **********************
446                     * RUN basiccmd DOT *
447                     **********************
448                     *
449 E5C1 CDF9E5         RDOT    CALL  :E5F9     Eval dot addr + colour
450 E5C4 C5                     PUSH  B
451 E5C5 4B                     MOV   C,E       Colour in C
452 E5C6 EF                     RST   5         Draw dot on screen
453 E5C7 1E                     DATA  :1E
454 E5C8 C1             RDT10   POP   B
455 E5C9 DA02E6         RDT20   JC    :E602     Jump if screen error
456 E5CC B7                     ORA   A         No special action
457 E5CD C9                     RET
458                     *
459                     ************************
460                     * RUN basiccmd DRAW *
461                     ************************
462                     *
463 E5CE CDE0E5         RDRAW   CALL  :E5E0     Eval begin/end addr +
464                                             colour
465 E5D1 E3                     XTHL
466 E5D2 EF                     RST   5         Draw a line on screen
467 E5D3 21                     DATA  :21
468 E5D4 C3C8E5                 JMP   :E5C8     Evt. run screen error
469                     *
470                     ************************
471                     * RUN basiccmd FILL *
472                     ************************
473                     *
474 E5D7 CDE0E5         RFILL   CALL  :E5E0     Eval coordinates, colour
475 E5DA E3                     XTHL
476 E5DB EF                     RST   5         Fill a rectangular area
477 E5DC 24                     DATA  :24
478 E5DD C3C8E5                 JMP   :E5C8     Evt run screen error
479                     *
480                     ************************************
481                     * EVALUATE 2 COORDINATES + COLOUR *
482                     ************************************
483                     *
484 E5E0 CDF3E5         R2COC   CALL  :E5F3     Get 1st coordinate
485 E5E3 E3                     XTHL            X-coord on stack
486 E5E4 E5                     PUSH  H
487 E5E5 5F                     MOV   E,A       Y-coord in E
488 E5E6 CDF3E5                 CALL  :E5F3     Get 2nd coordinate,
489                                             x-coord in HL
490 E5E9 57                     MOV   D,A       Y-coord in D
491 E5EA CDFDE5                 CALL  :E5FD     Get colour in A
492 E5ED C5                     PUSH  B
493 E5EE D5                     PUSH  D
494 E5EF EB                     XCHG
495 E5F0 C1                     POP   B
496 E5F1 E1                     POP   H
497 E5F2 C9                     RET
```

```
498                     *
499                     ************************
500                     * EVALUATE COORDINATE *
501                     ************************
502                     *
503 E5F3 CDF8E6         RCOOR   CALL   :E6F8      Get x-coord in HL
504 E5F6 C31DE7                 JMP    :E71D      Get y-coord in A
505                     *
506                     ***********************************
507                     * EVALUATE COORDINATE + COLOUR *
508                     ***********************************
509                     *
510 E5F9 CDF3E5         RCOCO   CALL   :E5F3      Get x-coord in HL
511 E5FC 5F                     MOV    E,A        Y-coord in E
512 E5FD 3E17           RCOL    MVI    A,:17
513 E5FF C343E7                 JMP    :E743      Get colour (0-23) in A
514                     *
515                     *
516                     *
517 E602                        END
```

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

L0E99	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		