```
002                             ORG   :E000
003                     *
004                     *
005                     *
006                     *   ======================
007                     *** SCREEN DRIVING PACKAGE ***
008                     *   ======================
009                     *
010                     * Called by RST 5; DATA XX. XX indicates the
011                     * offset of E000 for the different entrypoints.
012                     *
013                     ***************
014                     * ENTRYPOINTS *
015                     ***************
016                     *
017                     * Screen functions:
018
019 E000 C3C3E0         ZSINIT  JMP   :E0C3     Initialise screen
020 E003 C302E1         ZSOUTC  JMP   :E102     Output one character
021 E006 C337E2         ZSCLT   JMP   :E237     Set text colours
022 E009 C379E2         ZSCUS   JMP   :E279     Set cursor position
023 E00C C3CCE2         ZSCUA   JMP   :E2CC     Ask cursor position and
024                                               size character screen
025 E00F C316E3         ZSCUM   JMP   :E316     Set cursor mode
026 E012 C344E3         ZSCUI   JMP   :E344     Flash cursor
027 E015 C38BE3         ZSFETC  JMP   :E38B     Get character from line
028 E018 C3D9E3         ZSMODE  JMP   :E3D9     Change mode
029 E01B C3A4E6         ZSCLG   JMP   :E6A4     Set graphics colours
030 E01E C310E7         ZSDOT   JMP   :E710     Draw a dot on the screen
031 E021 C31BE7         ZSDRAW  JMP   :E71B     Draw a line on the screen
032 E024 C318E8         ZSFILL  JMP   :E818     Fill a rectangular area
033 E027 C384E8         ZSCRN   JMP   :E884     Ask colour of a point on the
034                                               screen and the size of the
035                                               graphics screen
036                     * Edit functions:
037
038 E02A C3F4EB         ZEDIT   JMP   :EBF4     Initialise editor
039 E02D C31EEC         ZEDOB   JMP   :EC1E     Run edit command
040                     *
041                     ************************
042                     * CONSTANT TABLE MODE 0 *
043                     ************************
044                     *
045                     * These constant tables are moved into the screen
046                     * variables in RAM (0084-0098) when the appropriate
047                     * mode is entered.
048                     *
049                     * Except the last 4 data blocks, all values
050                     * are offset from the screen top address (BFFF for
051                     * a 48K machine). This is valid for all modes.
052                     *
053 E030 B00C         CON0    DBL   :0CB0     First free RAM byte
054 E032 0000                 DBL   :0000     Top of rolled area
055 E034 0000                 DBL   :0000     End graphics area
056 E036 1000                 DBL   :0010     Start character area
057 E038 A00C                 DBL   :0CA0     End character area
058 E03A B00C                 DBL   :0CB0     End screen
059 E03C 0000                 DBL   :0000     End area used splitting mode
060 E03E 0000                 DBL   :0000     Start archive save area
061                     *
062 E040 0000                 DBL   :0000     Number of blobs horizontally
063 E042 00                   DATA  :00       Number of lines of graphics
```

```
064 E043 00                 DATA  :00       Number saved graphics lines
065 E044 00                 DATA  :00       Number of bytes/line
066                 *
067                 ****************************
068                 * CONSTANT TABLE MODES 1/2 *
069                 ****************************
070                 *
071 E045 3806       CON1    DBL   :0638     First free RAM byte
072 E047 3001               DBL   :0130     Top area rolled up for mode
073 E049 2806               DBL   :0628     End graphics area
074 E04B 2806               DBL   :0628     CHS (dummy)
075 E04D 6008               DBL   :0860     End character area
076 E04F 3806               DBL   :0638     End screen
077 E051 4807               DBL   :0748     End area used splitting mode
078 E053 4007               DBL   :0740     Start graphic archive area
079                 *
080 E055 4800               DBL   :0048     Number of blobs horizontally
081 E057 41                 DATA  :41       Number of lines of graphics
082 E058 0C                 DATA  :0C       Number archive area lines
083 E059 18                 DATA  :18       Number of bytes/line
084                 *
085                 ******************************
086                 * CONSTANT TABLE MODES 1A/2A *
087                 ******************************
088                 *
089 E05A 6008       CON1A   DBL   :0860     First free RAM byte
090 E05C 3001               DBL   :0130     Top of rolled area
091 E05E 0805               DBL   :0508     End graphics area
092 E060 1805               DBL   :0518     Start character area
093 E062 3007               DBL   :0730     End character area
094 E064 4007               DBL   :0740     End screen
095 E066 4807               DBL   :0748     End area used splitting mode
096 E068 2806               DBL   :0628     Start graph temp save area
097                 *
098 E06A 4800               DBL   :0048     Number of blobs horizontally
099 E06C 41                 DATA  :41       Number of lines of graphics
100 E06D 0C                 DATA  :0C       Number saved graphics lines
101 E06E 18                 DATA  :18       Number of bytes/line
102                 *
103                 ****************************
104                 * CONSTANT TABLE MODES 3/4 *
105                 ****************************
106                 *
107 E06F 7C17       CON3    DBL   :177C     First free RAM byte
108 E071 6004               DBL   :0460     Top area rolled up
109 E073 6C17               DBL   :176C     End graphics area
110 E075 6C17               DBL   :176C     CHS (dummy)
111 E077 A419               DBL   :19A4     End character area
112 E079 7C17               DBL   :177C     End screen
113 E07B BC1B               DBL   :1BBC     End area used splitting mode
114 E07D 5415               DBL   :1554     Start graph archive area
115                 *
116 E07F A000               DBL   :00A0     Number of blobs horizontally
117 E081 82                 DATA  :82       Number of lines of graphics
118 E082 18                 DATA  :18       Number archive area lines
119 E083 2E                 DATA  :2E       Number of bytes/line
120                 *
121                 ******************************
122                 * CONSTANT TABLE MODES 3A/4A *
123                 ******************************
124                 *
125 E084 A419       CON3A   DBL   :19A4     First free RAM byte
```

```
126 E086 6004                   DBL   :0460     Top of rolled area
127 E088 1C13                   DBL   :131C     End graphics area
128 E08A 2C13                   DBL   :132C     Start character area
129 E08C 4415                   DBL   :1544     End character area
130 E08E 5415                   DBL   :1554     End screen
131 E090 BC1B                   DBL   :1BBC     End area used splitting mode
132 E092 6C17                   DBL   :176C     Start graph temp save area
133                     *
134 E094 A000                   DBL   :00A0     Number of blobs horizontally
135 E096 82                     DATA  :82       Number of lines of graphics
136 E097 18                     DATA  :18       Number saved graphics lines
137 E098 2E                     DATA  :2E       Number of bytes/line
138                     *
139                     ******************************
140                     * CONSTANT TABLE MODES 5/6 *
141                     ******************************
142                     *
143 E099 205A           CON5    DBL   :5A20     First free RAM byte
144 E09B 880F                   DBL   :0F88     Top area rolled up
145 E09D 105A                   DBL   :5A10     End graphics area
146 E09F 105A                   DBL   :5A10     CHS (dummy)
147 E0A1 485C                   DBL   :5C48     End character area
148 E0A3 205A                   DBL   :5A20     End screen
149 E0A5 8869                   DBL   :6988     End area used splitting mode
150 E0A7 D04C                   DBL   :4CD0     Start graph archive area
151                     *
152 E0A9 5001                   DBL   :0150     Number of blobs horizontally
153 E0AB 00                     DATA  :00       Number of lines of graphics
154 E0AC 2C                     DATA  :2C       Number saved graphics lines
155 E0AD 5A                     DATA  :5A       Number of bytes/line
156                     *
157                     ********************************
158                     * CONSTANT TABLE MODES 5A/6A *
159                     ********************************
160                     *
161 E0AE 485C           CON5A   DBL   :5C48     First free RAM byte
162 E0B0 880F                   DBL   :0F88     Top of rolled area
163 E0B2 984A                   DBL   :4A98     End graphics area
164 E0B4 A84A                   DBL   :4AA8     Start character area
165 E0B6 C04C                   DBL   :4CC0     End character area
166 E0B8 D04C                   DBL   :4CD0     End screen
167 E0BA 8869                   DBL   :6988     End area used splitting mode
168 E0BC 105A                   DBL   :5A10     Start graph temp save area
169                     *
170 E0BE 5001                   DBL   :0150     Number of blobs horizontally
171 E0C0 00                     DATA  :00       Number of lines of graphics
172 E0C1 2C                     DATA  :2C       Number saved graphics lines
173 E0C2 5A                     DATA  :5A       Number of bytes/line
174                     *
175                     **********************
176                     * INITIALISE SCREEN *
177                     **********************
178                     *
179                     * The screen is initialised into all character
180                     * format (mode 0), and the cursor mode is set
181                     * and it is positioned in the top left corner.
182                     * The normal mode set routine is used (memory
183                     * management routine).
184                     *
185                     * This is the only time the package is told the
186                     * startaddress of the screen. The colour format
                        * is as for SCOLT, SCOLG. The cursor format is
```

```
188                 * as for SCURM.
189                 *
190                 * Entry: HL: Top location screen RAM.
191                 *        DE: Points to list with initialisation
192                 *            parameters (start at C7E0).
193                 * Exit:  All registers maybe corrupted.
194                 *
195 E0C3 228000      SINIT   SHLD  :0080      Store startaddr screen
196 E0C6 D5                 PUSH  D
197 E0C7 11F0FF             LXI   D,:FFF0
198 E0CA 19                 DAD   D
199 E0CB 228200             SHLD  :0082      Set top of screen
200 E0CE E1                 POP   H
201 E0CF AF                 XRA   A
202 E0D0 329D00             STA   :009D      Select mode 1
203 E0D3 CD16E3             CALL  :E316      Set cursor type + info
204 E0D6 23                 INX   H
205 E0D7 23                 INX   H
206 E0D8 CD37E2             CALL  :E237      Init. colours COLORT
207 E0DB 3D                 DCR   A
208 E0DC 329D00             STA   :009D      Select mode 0
209 E0DF 110400             LXI   D,:0004
210 E0E2 19                 DAD   D          Get addr COLORG parameters
211 E0E3 CDA4E6             CALL  :E6A4      Init. colours COLORG
212 E0E6 19                 DAD   D          Get addr screen management
213                                          parameters
214 E0E7 5E                 MOV   E,M        )
215 E0E8 23                 INX   H          ) Get addr screen management
216 E0E9 56                 MOV   D,M        ) routine
217 E0EA 23                 INX   H
218 E0EB EB                 XCHG
219 E0EC 22C400             SHLD  :00C4      Store addr SMKRM
220 E0EF EB                 XCHG
221 E0F0 5E                 MOV   E,M        ) Get addr emergency
222 E0F1 23                 INX   H          ) stop routine
223 E0F2 56                 MOV   D,M        )
224 E0F3 EB                 XCHG
225 E0F4 22C600             SHLD  :00C6      Store addr em.stop routine
226 E0F7 3E10               MVI   A,:10
227 E0F9 329D00             STA   :009D      Select init. screen mode
228                                          (no text, no graphics)
229 E0FC 3EFF               MVI   A,:FF
230 E0FE CDD9E3             CALL  :E3D9      Set up screen for mode 0
231 E101 C9                 RET
232                 *
233                 ***********************************
234                 * OUTPUT A CHARACTER TO SCREEN *
235                 ***********************************
236                 *
237                 * Displays one character on the screen.
238                 *
239                 * Entry: A: Character to be displayed.
240                 * Exit:  ABCDEHL preserved.
241                 *        CY=1: Character ignored.
242                 *
243 E102 37          SOUTC   STC              CY=1
244 E103 F5                 PUSH  PSW
245 E104 C5                 PUSH  B
246 E105 D5                 PUSH  D
247 E106 E5                 PUSH  H
248 E107 CD1CE2             CALL  :E21C      Change to char mode if
249                                          not yet done
```

```
250 E10A 2A7200            LHLD  :0072     Get cursor position
251 E10D CD6BE3            CALL  :E36B     Delete cursor
252 E110 FE0D              CPI   :0D       Car.ret ?
253 E112 CA3DE1            JZ    :E13D     Then print it
254 E115 FE0C              CPI   :0C       Form feed ?
255 E117 CA59E1            JZ    :E159     Then clear screen
256 E11A FE08              CPI   :08       Backspace ?
257 E11C CA66E1            JZ    :E166     Then cancell last character
258 E11F F5                PUSH  PSW
259 E120 3A7A00            LDA   :007A     Get addr last byte on line
260 E123 BD                CMP   L         Reached ?
261 E124 CAA9E1            JZ    :E1A9     Then extend lines
262 E127 F1        OTC05   POP   PSW
263 E128 77                MOV   M,A       Put char on screen
264 E129 2B                DCX   H
265 E12A 2B                DCX   H         Points to next screen loc
266 E12B CD30E3    OTC10   CALL  :E330     Put cursor on screen
267                *
268                OTC20
269 E12E E1        XRCC    POP   H
270 E12F D1                POP   D
271 E130 C1                POP   B
272 E131 F1                POP   PSW
273 E132 3F                CMC             CY=0: char accepted
274 E133 C9                RET
275
276                * If character not accepted:
277
278 E134 F1        OTC25   POP   PSW
279 E135 CD30E3    OTC26   CALL  :E330     Put cursor on screen
280 E138 E1        XRET    POP   H
281 E139 D1                POP   D
282 E13A C1                POP   B
283 E13B F1                POP   PSW       CY=1: char ignored
284 E13C C9                RET
285
286                * If carriage return:
287
288 E13D 2A7800    OTC30   LHLD  :0078     Get startaddr current line
289 E140 EB                XCHG            in DE
290 E141 217AFF            LXI   H,:FF7A
291 E144 19                DAD   D         Get startaddr next line
292 E145 EB                XCHG            in DE
293 E146 2A8C00            LHLD  :008C     Get end char area
294 E149 CDFBE6            CALL  :E6FB     Check if end is reached
295 E14C EB                XCHG            Next line mode byte in HL
296 E14D CCCBE1            CZ    :E1CB     If end reached: scroll up
297                                        one line
298 E150 CCFDE1            CZ    :E1FD     and init. this line with
299                                        blanks
300 E153 CD87E6    OTC35   CALL  :E687     Cursor on begin next line
301 E156 C32EE1            JMP   :E12E     Quit; char accepted
302
303                * If form feed:
304
305 E159 2A8C00    OTC40   LHLD  :008C     Get end character area
306 E15C EB                XCHG            in DE
307 E15D 2A8A00            LHLD  :008A     Get start character area
308 E160 CDFDE1            CALL  :E1FD     Init screen with spaces
309 E163 C353E1            JMP   :E153     Cursor top left corner of
310                                        char area; popall; ret
311
```

```
312                     * If backspace:
313
314 E166 EB         OTC50   XCHG            Cursor position in DE
315 E167 2A7B00             LHLD  :007B     Get startaddr current line
316 E16A 01F8FF             LXI   B,:FFF8   Left border width
317 E16D 09                 DAD   B         Get addr 1st char on line
318 E16E CDFBE6             CALL  :E6FB     Cursor at begin of line?
319 E171 EB                 XCHG
320 E172 CA35E1             JZ    :E135     Then ignore char; abort
321 E175 23                 INX   H         ) Cursor one location
322 E176 23                 INX   H         ) backwards
323 E177 3620               MVI   M,:20     Load space in this location
324 E179 3A7B00             LDA   :007B     Get number of extended lines
325 E17C B7                 ORA   A
326 E17D CA2BE1             JZ    :E12B     If no cont line: put cursor
327                                           on screen
328 E180 FA2BE1             JM    :E12B     If char accepted: put
329                                           cursor on screen
330
331                     * Backspace on a continuation line:
332
333 E183 D5                 PUSH  D         Save addr 1st byte on line
334                                           on stack
335 E184 EB                 XCHG            HL is cursor position
336 E185 01F2FF             LXI   B,:FFF2
337 E188 09                 DAD   B         HL = end indent area
338 E189 CDFBE6             CALL  :E6FB     Compare DE-HL
339 E18C EB                 XCHG
340 E18D D1                 POP   D
341 E18E C22BE1             JNZ   :E12B     If not there: put cursor on
342                                           screen; quit, char accepted
343 E191 EB                 XCHG
344 E192 3620               MVI   M,:20     Else cancel cont char (C)
345 E194 217B00             LXI   H,:007B
346 E197 35                 DCR   M         Decr. number extended lines
347 E198 2A7B00             LHLD  :007B     Get startaddr current line
348 E19B 118600             LXI   D,:0086
349 E19E 19                 DAD   D         Pnts to start previous line
350 E19F CD98E6             CALL  :E698     Store addr line mode byte as
351                                           current one and set last
352                                           byte on that line
353 E1A2 1180FF             LXI   D,:FF80
354 E1A5 19                 DAD   D
355 E1A6 C32BE1             JMP   :E12B     Put cursor on screen; quit,
356                                           char accepted
357
358                     * If end of line is reached:
359
360 E1A9 3A7B00     OTC80   LDA   :007B     Get number extended lines
361 E1AC FE03               CPI   :03       Max (3) reached ?
362 E1AE D234E1             JNC   :E134     Then put cursor on screen,
363                                           ret
364 E1B1 3C                 INR   A         Incr. number ext. lines
365 E1B2 47                 MOV   B,A       Store it in B
366 E1B3 3E0D               MVI   A,:0D
367 E1B5 CD02E1             CALL  :E102     Output car.ret
368 E1B8 78                 MOV   A,B
369 E1B9 327B00             STA   :007B     Update nr ext. lines
370 E1BC 2A7200             LHLD  :0072     Get cursor position addr
371 E1BF CD6BE3             CALL  :E36B     Delete cursor
372 E1C2 3643               MVI   M,:43     Print 'C' at left of line
373 E1C4 11F2FF             LXI   D,:FFF2
```

```
374 E1C7 19                 DAD   D          Indent 6 pos
375 E1C8 C327E1             JMP   :E127      Store char on new pos; put
376                                          cursor on screen
377                 *
378                 *************
379                 * SCROLLING *
380                 *************
381                 *
382                 * Scrolls up text area. Moves the character area of
383                 * the screen up one line.
384                 * Only the characters are moved, not the control and
385                 * colour bytes.
386                 *
387                 * Entry: None.
388                 * Exit:  AF preserved, BC corrupted.
389                 *        DE: End of bottom line.
390                 *        HL: Start of bottom line.
391                 *
392 E1CB 017AFF     SCROLL LXI   B,:FF7A   -86 (length one line)
393
394                 * Entry from Edit:
395                 * Scroll screen for number of positions given in
396                 * BC (-2 = 1 position left):
397
398 E1CE F5         SCR10  PUSH  PSW
399 E1CF 2A8A00            LHLD  :008A      Get startaddr character area
400 E1D2 54                MOV   D,H        ) and store it in DE
401 E1D3 5D                MOV   E,L        )
402 E1D4 09                DAD   B          Get addr line mode byte
403                                          next line
404 E1D5 EB                XCHG             in DE
405 E1D6 01F8FF     SCR20  LXI   B,:FFF8
406 E1D9 09                DAD   B          Get 1st useable location
407                                          on 1st line
408 E1DA EB                XCHG             in DE
409 E1DB 09                DAD   B          Get 1st useable location
410                                          on 2nd line
411 E1DC EB                XCHG             in DE; 1st line in HL
412 E1DD 063C              MVI   B,:3C      max 60 characters
413 E1DF 1A         SCR30  LDAX  D          Get char from 2nd line
414 E1E0 77                MOV   M,A        and move it to 1st line
415 E1E1 1B                DCX   D
416 E1E2 1B                DCX   D          Next char 2nd line
417 E1E3 2B                DCX   H
418 E1E4 2B                DCX   H          Next loc 1st line
419 E1E5 05                DCR   B
420 E1E6 C2DFE1            JNZ   :E1DF      Next char to be moved 1 line
421 E1E9 01FAFF            LXI   B,:FFFA
422 E1EC 09                DAD   B          Get addr line mode byte
423                                          2nd line
424 E1ED EB                XCHG             in DE
425 E1EE 09                DAD   B          Get addr line mode byte
426                                          3rd line
427 E1EF EB                XCHG             in DE; 2nd line in HL
428 E1F0 E5                PUSH  H
429 E1F1 2A8C00            LHLD  :008C      Get addr end character area
430 E1F4 CDFBE6            CALL  :E6FB      Check if end reached
431 E1F7 E1                POP   H
432 E1F8 DAD6E1            JC    :E1D6      If not at end: scroll next
433                                          line
434 E1FB F1                POP   PSW
                           RET
```

```
436                     *
437                     *
438                     *
439 E1FD                        END
```

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

CON0	E030	CON1	E045	CON1A	E05A	CON3	E06F
CON3A	E084	CON5	E099	CON5A	E0AE	OTC05	E127
OTC10	E12B	OTC20	E12E	OTC25	E134	OTC26	E135
OTC30	E13D	OTC35	E153	OTC40	E159	OTC50	E166
OTC80	E1A9	SCR10	E1CE	SCR20	E1D6	SCR30	E1DF
SCROLL	E1CB	SINIT	E0C3	SOUTC	E102	XRCC	E12E
XRET	E138	ZEDIT	E02A	ZEDOB	E02D	ZSCLG	E01B
ZSCLT	E006	ZSCRN	E027	ZSCUA	E00C	ZSCUI	E012
ZSCUM	E00F	ZSCUS	E009	ZSDOT	E01E	ZSDRAW	E021
ZSFETC	E015	ZSFILL	E024	ZSINIT	E000	ZSMODE	E018
ZSOUTC	E003						

```
002                             ORG   :E1FD
003                     *
004                     *
005                     *
006                     ************************************
007                     * INITIALISE SCREEN CHARACTER AREA *
008                     ************************************
009                     *
010                     * Fills screen with spaces (clears screen).
011                     * The line mode byte is set #7A, the line colour
012                     * byte to #40, all colour bytes to #00 (4-colour
013                     * text), all character bytes to #20.
014                     *
015                     * Entry: HL: 1st byte after header.
016                     *        DE: end character area.
017                     * Exit:  All registers preserved.
018                     *
019 E1FD F5             FILLS   PUSH  PSW
020 E1FE C5                     PUSH  B
021 E1FF D5                     PUSH  D
022 E200 E5                     PUSH  H
023 E201 367A           FIS10   MVI   M,:7A     Set control byte for char
024                                             mode
025 E203 2B                     DCX   H
026 E204 3640                   MVI   M,:40     Set line colour byte
027 E206 2B                     DCX   H
028 E207 0642           FIS20   MVI   B,:42     Number of bytes/line
029 E209 3620           FIS30   MVI   M,:20     Data byte is space
030 E20B 2B                     DCX   H
031 E20C 3600                   MVI   M,:00     Colour byte is 00
032 E20E 2B                     DCX   H
033 E20F 05                     DCR   B
034 E210 C209E2                 JNZ   :E209     Next screen addr
035 E213 CDFBE6                 CALL  :E6FB     All lines done ?
036 E216 C201E2                 JNZ   :E201     Next line if not
037 E219 C338E1                 JMP   :E138     Popall, ret
038                     *
039                     ******************************
040                     * CHANGE TO CHARACTER MODE *
041                     ******************************
042                     *
043                     * If a character is output when the screen is in
044                     * all-graphic mode, the mode is changed to the
045                     * corresponding split-mode.
046                     * If not sufficient space available, mode 0 is
047                     * tried. If still insufficient space, the emergenc
048                     * stop routine is used.
049                     *
050 E21C F5             TMODE   PUSH  PSW
051 E21D 3A9D00                 LDA   :009D     Get current screen mode
052 E220 0F                     RRC             Already character mode ?
053 E221 DA31E2                 JC    :E231     Abort if true
054 E224 37                     STC             CY=1
055 E225 17                     RAL             Set for split mode
056 E226 CDD9E3                 CALL  :E3D9     Change mode
057 E229 3EFF                   MVI   A,:FF
058 E22B DCD9E3                 CC    :E3D9     Change to mode 0 if not
059                                             sufficient space
060 E22E DA33E2                 JC    :E233     Emergency stop if still
061                                             insufficient space
062 E231 F1             TMD10   POP   PSW
063 E232 C9                     RET
```

```
064
065                     * If no space for A-mode or mode 0:
066
067 E233 2AC600      TMD20   LHLD  :00C6      Get addr emergency stop
068                                            routine
069 E236 E9                 PCHL              Go to this routine
070                     *
071                     **********************
072                     * SET TEXT COLOURS *
073                     **********************
074                     *
075                     * The COLORT parameters are set; the header
076                     * and trailer of the character area are
077                     * initialised.
078                     * The colour values are between 0 and F. The
079                     * top 4 bits are ignored.
080                     * The colour change is immediate.
081                     *
082                     * The 1st 2 colours are the default background
083                     * and foreground colours for characters. The last
084                     * 2 are alternative, and may be used for (e.g.)
085                     * the cursor. Colours may be repeated.
086                     *
087                     * Entry: HL points to a vector of 4 bytes containing
088                     *        the colours to be set.
089                     * Exit:  All registers preserved.
090                     *
091 E237 F5           SCOLT   PUSH  PSW
092 E238 C5                   PUSH  B
093 E239 D5                   PUSH  D
094 E23A E5                   PUSH  H
095 E23B 117C00               LXI   D,:007C     Addr 1st byte colour
096                                              register memory
097 E23E CD54E2               CALL  :E254       Init. COLORT reg memory
098 E241 3A9D00               LDA   :009D       Get current screen mode
099 E244 1F                   RAR               Char mode?
100 E245 2A8A00               LHLD  :008A       Get startaddr char area
101 E248 DC67E2               CC    :E267       If char mode: set colours
102                                              header area
103 E24B 2A8E00               LHLD  :008E       Get addr end of screen
104 E24E DC67E2               CC    :E267       If char mode: set colours
105                                              trailer area
106 E251 C338E1               JMP   :E138       Popall, ret
107                     *
108                     ************************
109                     * SET COLOUR PARAMETERS *
110                     ************************
111                     *
112                     * Loads colour data from ROM into the RAM pointers.
113                     * The high nibbles are 8x, 9x, Ax, Bx.
114                     * Used for both COLORT and COLORG.
115                     *
116                     * Entry: HL: Points to colour parameters.
117                     *        DE: Address colour memory in RAM.
118                     * Exit:  DE: Points after colour memory.
119                     *        Other registers corrupted.
120                     *
121 E254 018010      VCOPY   LXI   B,:1080
122 E257 7E          VCP10   MOV   A,M         Get colour from ROM
123 E258 E60F               ANI   :0F
124 E25A B1                 ORA   C           Add bits 4-7
125 E25B 12                 STAX  D           Store in RAM
```

```
126 E25C 23                 INX   H          Next colour
127 E25D 13                 INX   D          Next RAM location
128 E25E 79                 MOV   A,C        )
129 E25F 80                 ADD   B          ) Add #10 to C
130 E260 4F                 MOV   C,A        )
131 E261 FEC0               CPI   :C0        Check if finished
132 E263 C257E2             JNZ   :E257      Next one if not
133 E266 C9                 RET
134                 *
135                 ****************************************
136                 * LOAD COLOURS IN HEADER/TRAILER AREA *
137                 ****************************************
138                 *
139                 * Sets blanking area colour bytes according to
140                 * information given.
141                 * The colourbytes for the character area are
142                 * loaded into the screen header and trailer area.
143                 *
144                 * Entry: HL: Points to 1st control byte after
145                 *            blanking area.
146                 *         DE: Points after table with colours
147                 *            in RAM.
148                 * Exit:  AFDE preserved, BCHL corrupted.
149                 *
150 E267 F5        BCOLS  PUSH  PSW
151 E268 D5               PUSH  D
152 E269 010400           LXI   B,:0004    Distance between colour byte
153 E26C 2B               DCX   H          Addr 1st colour byte of
154                                        screen RAM
155 E26D 1B        BCS10  DCX   D          Addr colour table
156 E26E 1A               LDAX  D          Get colour byte
157 E26F 09               DAD   B          HL = addr in screen RAM
158 E270 77               MOV   M,A        Load byte into screen RAM
159 E271 E630             ANI   :30        Finished ?
160 E273 C26DE2           JNZ   :E26D      Next colourbyte if not
161 E276 D1               POP   D
162 E277 F1               POP   PSW
163 E278 C9               RET
164                 *
165                 ************************
166                 * SET CURSOR POSITION *
167                 ************************
168                 *
169                 * Moves the cursor from its current position to
170                 * any requested position.
171                 * Position 0,0 is the bottom left corner.
172                 *
173                 * Entry: HL contains the y,x position required
174                 *           for the cursor.
175                 * Exit:  BCDEHL preserved.
176                 *        CY=0: OK. F corrupted, A preserved.
177                 *        CY=1: Request off screen.
178                 *              A=01 (error code 'off screen').
179                 *
180 E279 B7        SCURS  ORA   A
181 E27A E5               PUSH  H
182 E27B D5               PUSH  D
183 E27C C5               PUSH  B
184 E27D F5               PUSH  PSW
185 E27E 7D               MOV   A,L        X-coord in A
186 E27F FE3C             CPI   :3C        After end of line ?
                          JNC   :E2C5      Then request off screen
```

```
188 E284 87                     ADD   A          X-coord *2
189 E285 4F                     MOV   C,A        in C
190 E286 0618                   MVI   B,:18      Nr of lines in mode 0
191 E288 3A9D00                 LDA   :009D      Get current screen mode
192 E28B B7                     ORA   A
193 E28C FA95E2                 JM    :E295      Jump if mode 0
194 E28F 0604                   MVI   B,:04      Nr of lines in A-modes
195 E291 1F                     RAR
196 E292 D2C5E2                 JNC   :E2C5      Error if all-graphics mode
197 E295 7C             SCS10   MOV   A,H        Y-coord in A
198 E296 B8                     CMP   B          More than max value
199 E297 D2C5E2                 JNC   :E2C5      Then request off screen
200 E29A CD6BE3                 CALL  :E36B      Delete old cursor
201 E29D 3C                     INR   A
202 E29E 218600                 LXI   H,:0086    Length 1 char line
203 E2A1 CD46EB                 CALL  :EB46      Calc length reqd number of
204                                              lines (HL=A*HL)
205 E2A4 EB                     XCHG             in DE
206 E2A5 2A8C00                 LHLD  :008C      Store end archive area
207 E2A8 19                     DAD   D          Start of reqd line
208 E2A9 CD98E6                 CALL  :E698      Store addr line mode byte
209                                              current line and store last
210                                              byte on that line
211 E2AC 110B00                 LXI   D,:000B
212 E2AF CDF2E6                 CALL  :E6F2      HL=start of right border
213 E2B2 59                     MOV   E,C
214 E2B3 1600                   MVI   D,:00
215 E2B5 CDF2E6                 CALL  :E6F2      Subtract char offset
216 E2B8 CD30E3                 CALL  :E330      Put cursor on screen
217 E2BB 3E00                   MVI   A,:00
218 E2BD 327B00                 STA   :007B      No extended lines
219 E2C0 F1                     POP   PSW        No-error return
220 E2C1 C1             SCS20   POP   B
221 E2C2 D1                     POP   D
222 E2C3 E1                     POP   H
223 E2C4 C9                     RET
224
225                     * If error 'off screen':
226
227 E2C5 F1             SCS30   POP   PSW
228 E2C6 3E01                   MVI   A,:01      Set error code
229 E2C8 3F                     CMC              Change CY to 1
230 E2C9 C3C1E2                 JMP   :E2C1      Pop, ret
231                     *
232                     ***************************************************
233                     * ASK CURSOR POSITION AND SIZE CHARACTER SCREEN *
234                     ***************************************************
235                     *
236                     * Returns the position of the cursor and the
237                     * range of possible values.
238                     * Values given in DE are maximum values of
239                     * coordinates.
240                     * If the mode is all graphics: DE=HL=0.
241                     *
242                     * Entry: None.
243                     * Exit:  HL gives y,x cursor position.
244                     *        DE gives y,x size of character part of
245                     *           the screen (mode 0: #17,#3B; A-modes:
246                     *           #03,#3B).
247                     *        AFBC preserved.
248                     *
249 E2CC F5             SCURA   PUSH  PSW
```

```
250 E2CD C5                 PUSH  B
251 E2CE 210000             LXI   H,:0000
252 E2D1 54                 MOV   D,H
253 E2D2 5D                 MOV   E,L       DE=HL=0
254 E2D3 3A9D00             LDA   :009D     Get current screen mode
255 E2D6 1F                 RAR             Char mode ?
256 E2D7 D213E3             JNC   :E313     Abort if not
257 E2DA 2A7800             LHLD  :0078     Get startaddr cursor line
258 E2DD E5                 PUSH  H         Save it on stack
259 E2DE 11F8FF             LXI   D,:FFF8   Size left border
260 E2E1 19                 DAD   D         Get addr 1st char byte
261 E2E2 EB                 XCHG            in DE
262 E2E3 2A7200             LHLD  :0072     Get cursor pos addr
263 E2E6 EB                 XCHG
264 E2E7 CDF2E6             CALL  :E6F2     Calc difference of cursor
265                                         pos from begin of line
266 E2EA D1                 POP   D         Get startaddr current line
267 E2EB 7D                 MOV   A,L       (x-coord cursor)*2 in A
268 E2EC B7                 ORA   A
269 E2ED 1F                 RAR             Now x-coord cursor in A
270 E2EE F5                 PUSH  PSW       Save it on stack
271 E2EF 2A8C00             LHLD  :008C     Get addr end char area
272 E2F2 018600             LXI   B,:0086   Length 1 char line
273 E2F5 AF                 XRA   A         Init Y-pos
274 E2F6 F5         SCA10   PUSH  PSW       Save it on stack
275 E2F7 09                 DAD   B         Get line mode byte next line
276 E2F8 CDFBE6             CALL  :E6FB     Is current line this line?
277 E2FB CA03E3             JZ    :E303     Then jump
278 E2FE F1                 POP   PSW       Get Y-coord
279 E2FF 3C                 INR   A         Incr it
280 E300 C3F6E2             JMP   :E2F6     Check if on next line
281 E303 E1         SCA20   POP   H         Y-coord cursor in H
282 E304 F1                 POP   PSW       X-coord cursor in A
283 E305 6F                 MOV   L,A       and now in L
284 E306 1617               MVI   D,:17     Nr of lines for mode 0 -1
285 E308 3A9D00             LDA   :009D     Get current screen mode
286 E30B B7                 ORA   A
287 E30C FA11E3             JM    :E311     Jump if mode 0
288 E30F 1603               MVI   D,:03     Nr of lines for A-modes -1
289 E311 1E3B       SCA30   MVI   E,:3B     Nr of char/line -1
290 E313 C1         SCA40   POP   B
291 E314 F1                 POP   PSW
292 E315 C9                 RET
293                 *
294                 *
295                 *
296 E316                    END
```

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

BCOLS	E267	BCS10	E26D	FILLS	E1FD	FIS10	E201
FIS20	E207	FIS30	E209	SCA10	E2F6	SCA20	E303
SCA30	E311	SCA40	E313	SCOLT	E237	SCS10	E295
SCS20	E2C1	SCS30	E2C5	SCURA	E2CC	SCURS	E279
TMD10	E231	TMD20	E233	TMODE	E21C	VCOPY	E254
VCP10	E257						

```
002                             ORG   :E316
003                     *
004                     *
005                     *
006                     ********************
007                     * SET CURSOR MODE *
008                     ********************
009                     *
010                     * The format of cursor info is 1 byte cursor type
011                     * and 1 byte of information.
012                     * If the type = 0, the cursor flashes in colour.
013                     * The info is a mask which is exored with the
014                     * colour byte for that character to flash it.
015                     * If the type = 1, the cursor alternates between
016                     * the actual character and the one in the info.
017                     *
018                     * If flash entry is never called, cursor will be
019                     * steady in the alternate colour (type 0) or per-
020                     * manently the alternate character (type 1).
021                     *
022                     * Entry: HL points to new cursor info.
023                     * Exit:  All registers preserved.
024                     *
025 E316 F5             SCURM   PUSH  PSW
026 E317 C5                     PUSH  B
027 E318 D5                     PUSH  D
028 E319 E5                     PUSH  H
029 E31A 3A9D00                 LDA   :009D     Get current screen mode
030 E31D 1F                     RAR             Char mode ?
031 E31E DC6BE3                 CC    :E36B     Then delete current cursor
032 E321 7E                     MOV   A,M       Get new cursor type
033 E322 327400                 STA   :0074     Store it in pointer
034 E325 23                     INX   H
035 E326 7E                     MOV   A,M       Get new cursor info
036 E327 327500                 STA   :0075     Store it in pointer
037
038                     * Entry from CURSET:
039
040 E32A DC44E3         SCM10   CC    :E344     Flash cursor once if in
041                                             char mode
042 E32D C338E1                 JMP   :E138     Popall, ret
043                     *
044                     **************
045                     * SET CURSOR *
046                     **************
047                     *
048                     * Sets some cursor on the screen. Does not delete
049                     * a previous cursor. The screen must already be
050                     * in a character mode.
051                     * Gets the contents of the cursor position address
052                     * and stores it in the pointers.
053                     *
054                     * Entry: HL: Address new cursor position.
055                     * Exit:  All registers preserved.
056                     *
057 E330 F5             CURSET  PUSH  PSW
058 E331 C5                     PUSH  B
059 E332 D5                     PUSH  D
060 E333 E5                     PUSH  H
061 E334 E5                     PUSH  H
062 E335 56                     MOV   D,M       Get contents addr pointed at
063                                             by new cursor
```

```
064 E336 2B                 DCX   H
065 E337 2B                 DCX   H
066 E338 2B                 DCX   H
067 E339 5E                 MOV   E,M       Get colour byte of this addr
068 E33A E1                 POP   H
069 E33B CD8DD6             CALL  :D68D     Store contents and colour
070                                         byte in cursor pointers
071 E33E 00                 NOP
072 E33F 00                 NOP
073 E340 37                 STC             CY=1
074 E341 C32AE3             JMP   :E32A     Flash cursor, popall, ret
075                 *
076                 ****************
077                 * FLASH CURSOR *
078                 ****************
079                 *
080                 * Flashes the cursor once if in char mode,
081                 * otherwise does nothing.
082                 *
083                 * Entry: None.
084                 * Exit:  All registers preserved.
085                 *
086                 SCURI
087 E344 F5        CURFL  PUSH  PSW
088 E345 E5               PUSH  H
089 E346 2A7200           LHLD  :0072     Get cursor pos addr
090 E349 7C               MOV   A,H
091 E34A B5               ORA   L         Check if addr is 0000
092 E34B CA5DE3           JZ    :E35D     Abort if no cursor
093 E34E 3A7400           LDA   :0074     Get cursor type
094 E351 B7               ORA   A         Check type
095 E352 3A7500           LDA   :0075     Get cursor info
096 E355 C260E3           JNZ   :E360     Jump if char type
097
098                 * If 'colour' type:
099
100 E358 2B               DCX   H         )
101 E359 2B               DCX   H         ) Get addr colour byte
102 E35A 2B               DCX   H         )
103 E35B AE               XRA   M         Exor mask with colour byte
104 E35C 77        CFL05  MOV   M,A       And reload colour byte
105 E35D E1        CFL10  POP   H
106 E35E F1               POP   PSW
107 E35F C9               RET
108
109                 * If 'char' type:
110
111 E360 BE         CFL20  CMP   M         Check contents screen loc
112 E361 77        CFL30  MOV   M,A       Move cursor info in loc
113 E362 C25DE3           JNZ   :E35D     Abort if contents screen
114                                         loc is changed now
115 E365 3A7700           LDA   :0077     Else: get contents scrn loc
116 E368 C35CE3           JMP   :E35C     Store it in this loc
117                 *
118                 ****************
119                 * DELETE CURSOR *
120                 ****************
121                 *
122                 * Deletes the current cursor. Loads the address
123                 * pointed at by the cursor with the data stored
124                 * in RAM (0076/77).
125                 * Routine valid for character modes only.
```

```
126                     *
127                     * Entry: None.
128                     * Exit:  All registers preserved.
129                     *
130 E36B F5             CURDEL  PUSH  PSW
131 E36C C5                     PUSH  B
132 E36D D5                     PUSH  D
133 E36E E5                     PUSH  H
134 E36F 2A7600                 LHLD  :0076     Get contents cursor loc
135 E372 EB                     XCHG            in DE
136 E373 2A7200                 LHLD  :0072     Get cursor pos addr
137 E376 E5                     PUSH  H         Save it on stack
138 E377 210000                 LXI   H,:0000
139 E37A 227200                 SHLD  :0072     Move cursor to addr 0000
140 E37D E1                     POP   H         Restore cursor pos addr
141 E37E 7C                     MOV   A,H
142 E37F B5                     ORA   L         Check if addr is 0000
143 E380 CA88E3                 JZ    :E388     Abort if no cursor
144 E383 72                     MOV   M,D       Load data into screen loc
145                                             pointed at by cursor
146 E384 2B                     DCX   H
147 E385 2B                     DCX   H
148 E386 2B                     DCX   H
149 E387 73                     MOV   M,E       Load colourbyte into loc
150                                             pointed at
151 E388 C338E1         CDL10   JMP   :E138     Popall, ret
152                     *
153                     ****************************
154                     * GET CHARACTER FROM LINE *
155                     ****************************
156                     *
157                     * Returns a character from some position on
158                     * the current line.
159                     *
160                     * Entry: C: Line position of required character.
161                     *           (max. legal value = 219).
162                     * Exit:  A: Required character (car.ret if at
163                     *           or past cursor).
164                     *        BCDEHLF preserved.
165                     *
166 E38B C5             SFETC   PUSH  B
167 E38C D5                     PUSH  D
168 E38D E5                     PUSH  H
169 E38E F5                     PUSH  PSW
170 E38F 218600                 LXI   H,:0086   Total nr. of bytes/line
171 E392 3A7B00                 LDA   :007B     Get number extended lines
172 E395 CD46EB                 CALL  :EB46     Calc total nr of bytes
173                                             (HL=A*HL)
174 E398 EB                     XCHG            in DE
175 E399 2A7800                 LHLD  :0078     Get addr line mode byte
176                                             current line
177 E39C 19                     DAD   D         Calc start of line on screen
178 E39D 11EAFF                 LXI   D,:FFEA
179 E3A0 19                     DAD   D         End indent area
180 E3A1 EB                     XCHG            in DE
181 E3A2 3EF9                   MVI   A,:F9     1st bytes on line not
182                                             useable
183 E3A4 81                     ADD   C         Add pos of required char on
184                                             line
185 E3A5 F5                     PUSH  PSW
186 E3A6 0600                   MVI   B,:00
187 E3A8 D2B2E3                 JNC   :E3B2     Jump if in 1st 7 positions
```

```
188 E3AB 05              DCR   B
189 E3AC D635      SFC10 SUI   :35       60 useable positions/line
190 E3AE 04              INR   B         Count nr of extended lines
191 E3AF D2ACE3          JNC   :E3AC     Jump if not on this line
192 E3B2 78        SFC20 MOV   A,B       Nr of extensions in A
193 E3B3 21E4FF          LXI   H,:FFE4   Nr of not used bytes/line
194 E3B6 CD46EB          CALL  :EB46     ) Add-ons for line ends
195 E3B9 19              DAD   D         )
196 E3BA F1              POP   PSW       Restore pos of char on line
197 E3BB 5F              MOV   E,A       into E
198 E3BC 3F              CMC
199 E3BD 9F              SBB   A
200 E3BE 57              MOV   D,A       D=char.count - nr of idents
201 E3BF EB              XCHG
202 E3C0 29              DAD   H         Pos *2 due to colour bytes
203 E3C1 EB              XCHG
204 E3C2 CDF2E6          CALL  :E6F2     Calc pos of reqd char
205 E3C5 EB              XCHG            Addr in DE
206 E3C6 2A7200          LHLD  :0072     Get cursor pos addr
207 E3C9 CDFBE6          CALL  :E6FB     Compare it with addr of char
208 E3CC 3E0D            MVI   A,:0D     Car.ret in A
209 E3CE D2D2E3          JNC   :E3D2     If on or after cursor
210 E3D1 1A              LDAX  D         Get character from line
211 E3D2 67        SFC30 MOV   H,A       Save it temporarily
212 E3D3 F1              POP   PSW       Restore flags
213 E3D4 7C              MOV   A,H       Get character in A
214 E3D5 E1              POP   H
215 E3D6 D1              POP   D
216 E3D7 C1              POP   B
217 E3D8 C9              RET
218                *
219                ***************
220                * CHANGE MODE *
221                ***************
222                *
223                * Change the mode of the screen.
224                *
225                * Entry: A: Code new mode.
226                * Exit:  ABCDEHL preserved.
227                *        CY=0: OK.
228                *        CY=1: Insufficient room for mode.
229                *
230 E3D9 37        SSETM STC             CY=1
231 E3DA F5              PUSH  PSW
232 E3DB C5              PUSH  B
233 E3DC D5              PUSH  D
234 E3DD E5              PUSH  H
235 E3DE FEFF            CPI   :FF       Mode 0 ?
236 E3E0 CC07E4          CZ    :E407     Then set up mode 0 screen
237 E3E3 C43EE4          CNZ   :E43E     Else: Set up screen for
238                                      other modes
239 E3E6 DA04E4          JC    :E404     Jump if no room available
240 E3E9 2A8E00          LHLD  :008E     Get end of screen
241 E3EC D5              PUSH  D
242 E3ED 111000          LXI   D,:0010   Nr of bytes in trailer
243 E3F0 19              DAD   D         Get 1st addr trailer area
244 E3F1 D1              POP   D         Get addr 1st colour
245 E3F2 060F            MVI   B,:0F     Depth of blank
246 E3F4 CDFCE5          CALL  :E5FC     Init trailer area
247 E3F7 2A8400          LHLD  :0084     Get 1st free byte
248 E3FA B7              ORA   A         Set flags on scrn mode byte
249 E3FB CDA6E5          CALL  :E5A6     Perform mem. management
```

```
250 E3FE 329D00             STA   :009D     Store current screen r
251 E401 C32EE1             JMP   :E12E     Popall (CY=0), ret
252
253                 * If error:
254
255 E404 C338E1     STM10   JMP   :E138     Popall (CY=1) ret
256                 *
257                 *
258                 *
259 E407                    END

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

CDL10  E388   CFL05  E35C   CFL10  E35D   CFL20  E360
CFL30  E361   CURDEL E36B   CURFL  E344   CURSET E330
SCM10  E32A   SCURI  E344   SCURM  E316   SFC10  E3AC
SFC20  E3B2   SFC30  E3D2   SFETC  E38B   SSETM  E3D9
STM10  E404
```

```
002                             ORG   :E407
003                     *
004                     *
005                     *
006                     ****************************
007                     * SET UP SCREEN FOR MODE 0 *
008                     ****************************
009                     *
010                     * Sets up a mode 0 screen, whatever the current mode
011                     * is. If already a character area exists, it is
012                     * moved to the top of the new screen.
013                     *
014                     * Entry: None.
015                     * Exit:  All registers corrupted.
016                     *        CY=0: OK.
017                     *        CY=1: No room.
018                     *
019 E407 37             SSM0    STC             CY=1
020 E408 F5                     PUSH  PSW
021 E409 2130E0                 LXI   H,:E030   Startaddr mode 0 table
022 E40C CD45E5                 CALL  :E545     Load pointers with parameter
023 E40F DA3CE4                 JC    :E43C     If no room available
024 E412 117C00                 LXI   D,:007C   Addr text colour table
025 E415 D5                     PUSH  D         Save addr
026 E416 2A8000                 LHLD  :0080     Get 1st byte screen RAM
027 E419 0606                   MVI   B,:06
028 E41B CDFCE5                 CALL  :E5FC     Set up header
029 E41E 3A9D00                 LDA   :009D     Get old screen mode
030 E421 FE10                   CPI   :10       During initialisation ?
031 E423 CA2DE4                 JZ    :E42D     Then jump
032 E426 1F                     RAR             Split screen ?
033 E427 D22DE4                 JNC   :E42D     Jump if not
034 E42A CD35E6                 CALL  :E635     If split mode: move old
035                                             text, cursor, etc
036 E42D EB             SS010   XCHG            Addr after header in DE
037 E42E 2A8C00                 LHLD  :008C     Get addr end char area
038 E431 EB                     XCHG
039 E432 CDFDE1                 CALL  :E1FD     Blank char area
040 E435 D487E6                 CNC   :E687     Set cursor at begin 1st line
041
042                     * Entry from SSMG:
043
044 E438 D1             SS015   POP   D
045 E439 F1                     POP   PSW
046 E43A 3F                     CMC             CY=0
047 E43B C9                     RET
048
049                     * If no room available:
050                     * Entry from SSMG, SSM, SSMA:
051
052 E43C F1             SS020   POP   PSW       CY=1
053 E43D C9                     RET
054                     *
055                     ************************************
056                     * SET UP SCREEN FOR GRAPHIC MODE *
057                     ************************************
058                     *
059                     * Entry: A:  Screen mode (split if odd).
060                     * Exit:  CY=0: O.K.:
061                     *        Split mode:        DE: Addr text colours.
062                     *        All graphic mode: DE: Addr graph colours.
                                AF preserved. BCHL corrupted.
```

```
064                     *       CY=1: Insufficient room.
065                     *
066 E43E 37             SSMG    STC
067 E43F F5                     PUSH  PSW
068 E440 57                     MOV   D,A       Screen mode in D
069 E441 E601                   ANI   :01       Z=1 if full colour mode
070 E443 7A                     MOV   A,D
071 E444 1F                     RAR             Disable split mode bit in A
072 E445 C4B6E4                 CNZ   :E4B6     If split mode: set up screen
073 E448 CC5FE4                 CZ    :E45F     If full colour mode: idem
074 E44B DA3CE4                 JC    :E43C     Abort if no room
075 E44E F1                     POP   PSW       Get mode code
076 E44F F5                     PUSH  PSW
077 E450 D5                     PUSH  D
078 E451 119E00                 LXI   D,:009E   Addr COLORG table
079 E454 2A8000                 LHLD  :0080     Get addr 1st byte screen RAM
080 E457 0606                   MVI   B,:06     Depth each blanking line
081                                             in header -1
082 E459 CDFCE5                 CALL  :E5FC     Set up header with COLORG
083                                             colours
084 E45C C338E4                 JMP   :E438     Quit, all OK
085                     *
086                     * SET UP A FULL GRAPHIC SCREEN:
087                     *
088                     * Sets up a screen RAM for an all-graphics mode.
089                     *
090                     * Entry: A: Mode code /2.
091                     *        D: Mode code.
092                     * Exit:  CY=0: O.K.:
093                     *              DE points to table graphic colours.
094                     *              AF preserved. BCHL corrupted.
095                     *        CY=1: Insufficient space.
096                     *
097 E45F 37             SSM     STC
098 E460 F5                     PUSH  PSW
099 E461 219AE5                 LXI   H,:E59A   Addr table vectors full
100                                             graphic mode
101 E464 CD39E5                 CALL  :E539     Set up screen mode
102 E467 DA3CE4                 JC    :E43C     Jump if no room
103 E46A 3A9D00                 LDA   :009D     Get current screen mode
104 E46D 92                     SUB   D         ) Check if change split
105 E46E 3D                     DCR   A         ) to all graphics
106 E46F CA00D7                 JZ    :D700     Then check if sufficient
107                                             RAM available and change
108                                             mode
109 E472 CD6BE3                 CALL  :E36B     Delete cursor
110 E475 3A9600                 LDA   :0096     Get nr of graphics lines
111 E478 4F                     MOV   C,A       in C
112 E479 2A8200                 LHLD  :0082     Get addr top graph area
113 E47C CDADE5                 CALL  :E5AD     Blank whole screen
114 E47F 119E00         SSM10   LXI   D,:009E   Addr COLORG table
115 E482 F1                     POP   PSW
116 E483 3F                     CMC             CY=0: O.K.
117 E484 C9                     RET
118
119                     * Change from split to all-graphic mode:
120
121 E485 D20FD7         SSM21   JNC   :D70F     Set up screen mode
122 E488 CD6BE3                 CALL  :E36B     Delete cursor
123 E48B 2A8800                 LHLD  :0088     Get addr temp save area
124 E48E 44                     MOV   B,H       ) in BC
125 E48F 4D                     MOV   C,L       )
```

```
126 E490 C5              PUSH  B         Save it on stack
127 E491 2A9200          LHLD  :0092     Get startaddr archive
128                                      area
129 E494 EB              XCHG            in DE
130 E495 2A8C00          LHLD  :008C     Get addr end archive area
131 E498 CDC2E6          CALL  :E6C2     Move archive area into
132                                      temp save area
133 E49B 2A8600          LHLD  :0086     Get addr top of rolled area
134 E49E 44              MOV   B,H       ) in BC
135 E49F 4D              MOV   C,L       )
136 E4A0 2A8200          LHLD  :0082     Get addr top old graphics
137 E4A3 EB              XCHG            in DE
138 E4A4 2A9900          LHLD  :0099     Get end old screen
139 E4A7 CDC2E6          CALL  :E6C2     Move lower part screen
140                                      downwards
141 E4AA 42              MOV   B,D       ) BC is addr where to put
142 E4AB 4B              MOV   C,E       ) archive area
143 E4AC D1              POP   D         Get startaddr temp save area
144 E4AD 2A9000          LHLD  :0090     Get end temp save area
145 E4B0 CDC2E6          CALL  :E6C2     Move temp save area to
146                                      top of screen
147 E4B3 C37FE4          JMP   :E47F     Quit
148                *
149                * SET UP SCREEN FOR SPLIT MODE:
150                *
151                * Sets up a split screen for a given mode in the
152                * lower RAM.
153                *
154                * Entry: A: Mode code /2.
155                *        D: Mode code.
156                * Exit:  CY=0: O.K.:
157                *              DE: Address text colour table.
158                *              AF preserved, BCHL corrupted.
159                *        CY=1: Insufficient space.
160                *
161 E4B6 37        SSMA  STC
162 E4B7 F5              PUSH  PSW
163 E4B8 21A0E5          LXI   H,:E5A0   Startaddr table vectors
164                                      split modes
165 E4BB CD39E5          CALL  :E539     Set up screen mode
166 E4BE DA3CE4          JC    :E43C     Abort if insufficient space
167 E4C1 3A9D00          LDA   :009D     Get old screen mode
168 E4C4 92              SUB   D         ) Check if change from
169 E4C5 3C              INR   A         ) all-graph to split
170 E4C6 F5              PUSH  PSW       Preserve flags
171 E4C7 D5              PUSH  D         and new mode code
172 E4C8 C2F9E4          JNZ   :E4F9     If not splitting old mode:
173                                      clear graph and moved areas
174 E4CB CD06D7          CALL  :D706     Check suff RAM available;
175                                      prepare full graphic mode
176 E4CE 00              NOP
177 E4CF D20DD7          JNC   :D70D     Set up current mode if
178                                      not O.K.
179 E4D2 2A9200          LHLD  :0092     Get start temp save area
180 E4D5 44              MOV   B,H       ) in BC
181 E4D6 4D              MOV   C,L       )
182 E4D7 2A8200          LHLD  :0082     Get addr after header
183 E4DA EB              XCHG            in DE
184 E4DB 2A8600          LHLD  :0086     Get addr top of screen
185 E4DE CDC2E6          CALL  :E6C2     Move top of screen into
186                                      temp save area
187 E4E1 42              MOV   B,D       ) BC is addr top of screen
```

```
188 E4E2 4B                 MOV   C,E       )
189 E4E3 EB                 XCHG            Addr top rolled up area
190 E4E4 2A9900             LHLD  :0099     Get previous end of graphics
191 E4E7 CDC2E6             CALL  :E6C2     Move lower part of screen
192 E4EA 2A8E00             LHLD  :008E     Get final place for archive
193                                         code
194 E4ED 44                 MOV   B,H       ) in BC
195 E4EE 4D                 MOV   C,L       )
196 E4EF 2A9200             LHLD  :0092     Get addr start temp. save
197                                         area
198 E4F2 EB                 XCHG            in DE
199 E4F3 2A9000             LHLD  :0090     Get addr end split mode
200 E4F6 CDC2E6             CALL  :E6C2     Move temp save area into
201                                         archive area
202 E4F9 2A8A00     SMA10   LHLD  :008A     Get start addr char area
203 E4FC EB                 XCHG            in DE
204 E4FD 2A8C00             LHLD  :008C     Get addr end char area
205 E500 3A9D00             LDA   :009D     Get current screen mode
206 E503 1F                 RAR             Check mode
207 E504 0E04               MVI   C,:04     Nr of char lines in A-mode
208 E506 EB                 XCHG
209 E507 D4FDE1             CNC   :E1FD     Blank char area
210 E50A D487E6             CNC   :E687     Cursor on begin of line
211 E50D DC35E6             CC    :E635     Find old text and move it
212 E510 D1                 POP   D
213 E511 2A8200             LHLD  :0082     Get addr after header
214 E514 3A9700             LDA   :0097     Get nr saved graphics lines
215 E517 4F                 MOV   C,A       in C
216 E518 3A9600             LDA   :0096     Get nr of graphics lines
217 E51B 91                 SUB   C         minus saved ones
218 E51C 4F                 MOV   C,A       stored in C
219 E51D F1                 POP   PSW
220 E51E C4ADE5             CNZ   :E5AD     Blank visible graph area
221 E521 2A8E00             LHLD  :008E     Get addr end of screen
222 E524 3A9700             LDA   :0097     Get nr saved graphics lines
223 E527 4F                 MOV   C,A       in C
224 E528 C4ADE5             CNZ   :E5AD     Blank saved graph area
225 E52B 117C00             LXI   D,:007C   Addr text colour table
226 E52E 0600               MVI   B,:00     Middle as narrow as possible
227 E530 2A8800             LHLD  :0088     Get addr middle area
228 E533 F1                 POP   PSW       Get mode code
229 E534 CDFCE5             CALL  :E5FC     Set up middle area
230                                         (blanking)
231 E537 3F                 CMC
232 E538 C9                 RET
233                 *
234                 * SET UP SCREEN FOR MODE:
235                 *
236                 * Selects the right table according to the mode
237                 * number and sets the screen variables.
238                 *
239                 * Entry: A:  Mode code /2.
240                 *        HL: Points to screen parameter vectors
241                 *            for each pair of modes.
242                 * Exit:  CY=0: O.K.:
243                 *              AFHL corrupted, BCDE preserved.
244                 *        CY=1: Insufficient space.
245                 *
246 E539 E60E          TABP    ANI   :0E       Bits 1,2,3 only
247 E53B CD01E7               CALL  :E701     Add offset to start table
248 E53E 00                   NOP
249 E53F 00                   NOP
```

```
250 E540 00                 NOP
251 E541 7E                 MOV   A,M       )
252 E542 23                 INX   H         ) Get addr from table in HL
253 E543 66                 MOV   H,M       )
254 E544 6F                 MOV   L,A       )
255                 *
256                 * LOAD POINTERS WITH SCREEN PARAMETERS:
257                 *
258                 * Set up vector area 0084-0098 with variables
259                 * describing the current state of the screen
260                 * in the current mode.
261                 *
262 E545 37           VARS   STC
263 E546 F5                 PUSH  PSW
264 E547 C5                 PUSH  B
265 E548 D5                 PUSH  D
266 E549 E5                 PUSH  H
267 E54A E5                 PUSH  H
268 E54B E5                 PUSH  H
269 E54C 2A8000             LHLD  :0080     Get addr 1st byte screen RAM
270 E54F 44                 MOV   B,H       in BC
271 E550 4D                 MOV   C,L
272 E551 E1                 POP   H         Get startaddr table
273 E552 79                 MOV   A,C       )
274 E553 96                 SUB   M         )
275 E554 5F                 MOV   E,A       ) Calc end area used in new
276 E555 23                 INX   H         ) mode. Store it in DE.
277 E556 78                 MOV   A,B       )
278 E557 9E                 SBB   M         )
279 E558 57                 MOV   D,A       )
280 E559 210000             LXI   H,:0000
281 E55C DA60E5             JC    :E560     Jump if insufficient space
282 E55F EB                 XCHG
283 E560 37          VRS05  STC
284 E561 CDA6E5             CALL  :E5A6     Make room for new mode
285 E564 D296E5             JNC   :E596     Jump if no room available
286 E567 2A8800             LHLD  :0088     Get old addr after end
287                                         graphics area
288 E56A 229900             SHLD  :0099     and save it
289 E56D 2A8A00             LHLD  :008A     Get old startaddr char area
290 E570 229B00             SHLD  :009B     and save it
291
292                 * Set up area 0084-0093:
293
294 E573 118400             LXI   D,:0084   Start of variables which
295                                         need offsets
296 E576 2E08               MVI   L,:08     Nr of pointers to be set
297 E578 E3          VRS10  XTHL            Get addr screen parameters
298 E579 79                 MOV   A,C
299 E57A 96                 SUB   M         Calc lobyte
300 E57B 12                 STAX  D         And store it in pointer
301 E57C 23                 INX   H         Next byte
302 E57D 13                 INX   D
303 E57E 78                 MOV   A,B
304 E57F 9E                 SBB   M         Calc hibyte
305 E580 12                 STAX  D         And store it in pointer
306 E581 23                 INX   H
307 E582 13                 INX   D
308 E583 E3                 XTHL
309 E584 2D                 DCR   L         Decr counter
310 E585 C278E5             JNZ   :E578     Next parameter
```

```
312                     * Set up area 0094-0098:
313
314 E588 E1                     POP   H         Get addr 1st parameter
315 E589 0605                   MVI   B,:05     Nr unadjusted constant bytes
316 E58B 7E             VRS20   MOV   A,M       Get parameter
317 E58C 12                     STAX  D         and store it in pointer
318 E58D 23                     INX   H
319 E58E 13                     INX   D
320 E58F 05                     DCR   B         Decr counter
321 E590 C28BE5                 JNZ   :E58B     Next parameter
322 E593 C32EE1                 JMP   :E12E     Popall, CY=0, ret
323
324                     * If no room available:
325
326 E596 E1             VRS30   POP   H
327 E597 C338E1                 JMP   :E138     Popall (CY=1), ret
328                     *
329                     * VECTORS TO TABLES SCREEN PARAMETERS:
330                     *
331                     * The startaddresses of the tables with para-
332                     * meters for the graphic modes are given.
333                     *
334 E59A 45E0            TABM    DBL   :E045     mode 1/2
335 E59C 6FE0                    DBL   :E06F     mode 3/4
336 E59E 99E0                    DBL   :E099     mode 5/6
337                     *
338 E5A0 5AE0            TABMA   DBL   :E05A     mode 1A/2A
339 E5A2 84E0                    DBL   :E084     mode 3A/4A
340 E5A4 AEE0                    DBL   :E0AE     mode 5A/6A
341                     *
342                     ****************************************
343                     * PERFORM MEMORY MANAGEMENT ROUTINE *
344                     ****************************************
345                     *
346                     * Entry: HL points to last free byte in RAM.
347                     *
348 E5A6 23              SMKRM   INX   H
349 E5A7 E5                     PUSH  H
350 E5A8 2AC400                 LHLD  :00C4     Get addr mem.management
351                                             routine
352 E5AB E3                     XTHL            Put it on stack
353 E5AC C9                     RET             Perform this routine and
354                                             return afterwards to origin.
355                                             returnaddress.
356                     *
357                     ************************************
358                     * SET UP AN EMPTY GRAPHICS AREA *
359                     ************************************
360                     *
361                     * Initialises an area of the screen into graphic
362                     * state and blanks it. In 16-colour modes, all
363                     * pixels are set 'on'. The foreground colour is
364                     * the first COLORG colour, the background is black.
365                     *
366                     * Entry: D:  Mode code.
367                     *        C:  Number of graphic lines (1-256).
368                     *        HL: Start of area.
369                     * Exit:  All registers preserved.
370                     *
371 E5AD F5              SGINIT  PUSH  PSW
372 E5AE C5                     PUSH  B
373 E5AF D5                     PUSH  D
```

```
374 E5B0 E5                 PUSH  H
375 E5B1 7A                 MOV   A,D       Mode in A
376 E5B2 E5                 PUSH  H
377 E5B3 110000             LXI   D,:0000   8 blobs 1st graph colour
378 E5B6 2E00               MVI   L,:00     Control byte: graph, low
379                                         def, 4-colour
380 E5B8 1F                 RAR
381 E5B9 1F                 RAR
382 E5BA DACBE5             JC    :E5CB     Jump if 4-colour mode
383
384                 * 16-colour mode only:
385
386 E5BD F5                 PUSH  PSW
387 E5BE 3A9E00             LDA   :009E     Get 1st colour
388 E5C1 87                 ADD   A         )
389 E5C2 87                 ADD   A         ) Move lonibble into
390 E5C3 87                 ADD   A         ) hinibble
391 E5C4 87                 ADD   A         )
392 E5C5 57                 MOV   D,A       Result in D
393 E5C6 1EFF               MVI   E,:FF     8 blobs foreground
394 E5C8 F1                 POP   PSW
395 E5C9 2E80               MVI   L,:80     Control byte: graph, low
396                                         def, 16-colour
397 E5CB 1F         SGI10   RAR
398 E5CC DADEE5             JC    :E5DE     Jump if mode 3/4
399
400                 * Mode 1/2 and 5/6 only:
401
402 E5CF 1F                 RAR
403 E5D0 260B               MVI   H,:0B     Low def fields/line
404 E5D2 3E03               MVI   A,:03     Low def bit mask
405 E5D4 D2E2E5             JNC   :E5E2     Jump if mode 1/2
406
407                 * Mode 5/6 only:
408
409 E5D7 262C               MVI   H,:2C     Super def fields/line
410 E5D9 3E20               MVI   A,:20     Super def bit mask
411 E5DB C3E2E5             JMP   :E5E2
412
413                 * Mode 3/4 only:
414
415 E5DE 2616         SGI20   MVI   H,:16     High def fields/line
416 E5E0 3E11               MVI   A,:11     High def bit mask
417                 *
418 E5E2 B5         SGI30   ORA   L         Add def bits to get mode
419                                         code
420 E5E3 47                 MOV   B,A
421 E5E4 7C                 MOV   A,H       Line length in A
422 E5E5 E1                 POP   H         Get top of area
423 E5E6 F5         SGI50   PUSH  PSW       Save line length
424 E5E7 70                 MOV   M,B       Load line control byte
425 E5E8 2B                 DCX   H
426 E5E9 3640               MVI   M,:40     Null line colour byte
427 E5EB 2B                 DCX   H
428 E5EC 73         SGI60   MOV   M,E       ) Load screen data locations
429 E5ED 2B                 DCX   H         ) with 1 blank field
430 E5EE 72                 MOV   M,D       )
431 E5EF 2B                 DCX   H
432 E5F0 3D                 DCR   A         Next screen location
433 E5F1 C2ECE5             JNZ   :E5EC     Jump if line not ready
434 E5F4 F1                 POP   PSW       Restore nr of locations in A
435 E5F5 0D                 DCR   C         Next screen line
```

```
436 E5F6 C2E6E5              JNZ   :E5E6     Jump if not ready
437 E5F9 C338E1              JMP   :E138     Popall, ret
438                  *
439                  *
440                  *
441 E5FC                     END
```

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

SGI10	E5CB	SGI20	E5DE	SGI30	E5E2	SGI50	E5E6
SGI60	E5EC	SGINIT	E5AD	SMA10	E4F9	SMKRM	E5A6
SS010	E42D	SS015	E438	SS020	E43C	SSM	E45F
SSM0	E407	SSM10	E47F	SSM21	E485	SSMA	E4B6
SSMG	E43E	TABM	E59A	TABMA	E5A0	TABP	E539
VARS	E545	VRS05	E560	VRS10	E578	VRS20	E58B
VRS30	E596						