```
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
```

ClibPDF - www.fastio.com

```
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
187                  * 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
```

ClibPDF - www.fastio.com

```
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 2A7800          LHLD  :0078     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 2A7800          LHLD  :0078     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
```

ClibPDF - www.fastio.com

```
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
435 E1FC C9                RET
```

ClibPDF - www.fastio.com


```
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
```

ClibPDF - www.fastio.com

```
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 ?
187 E281 D2C5E2           JNC   :E2C5      Then request off screen
```

ClibPDF - www.fastio.com


```
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 110800             LXI   D,:0008
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
```

ClibPDF - www.fastio.com

```
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
```

ClibPDF - www.fastio.com

```
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 m
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
```

ClibPDF - www.fastio.com

```
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.
063                 *        AF preserved. BCHL corrupted.
```

ClibPDF - www.fastio.com


```
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
```

ClibPDF - www.fastio.com

```
E540 00              NOP
E541 7E              MOV   A,M       )
E542 23              INX   H         ) Get addr from table in HL
E543 66              MOV   H,M       )
E544 6F              MOV   L,A       )
                *
                * LOAD POINTERS WITH SCREEN PARAMETERS:
                *
                * Set up vector area 0084-0098 with variables
                * describing the current state of the screen
                * in the current mode.
                *
E545 37         VARS    STC
E546 F5                 PUSH  PSW
E547 C5                 PUSH  B
E548 D5                 PUSH  D
E549 E5                 PUSH  H
E54A E5                 PUSH  H
E54B E5                 PUSH  H
E54C 2A8000             LHLD  :0080     Get addr 1st byte screen RAM
E54F 44                 MOV   B,H       in BC
E550 4D                 MOV   C,L
E551 E1                 POP   H         Get startaddr table
E552 79                 MOV   A,C       )
E553 96                 SUB   M         )
E554 5F                 MOV   E,A       ) Calc end area used in new
E555 23                 INX   H         ) mode. Store it in DE.
E556 78                 MOV   A,B       )
E557 9E                 SBB   M         )
E558 57                 MOV   D,A       )
E559 210000             LXI   H,:0000
E55C DA60E5             JC    :E560     Jump if insufficient space
E55F EB                 XCHG
E560 37         VRS05   STC
E561 CDA6E5             CALL  :E5A6     Make room for new mode
E564 D296E5             JNC   :E596     Jump if no room available
E567 2A8800             LHLD  :0088     Get old addr after end
                                        graphics area
E56A 229900             SHLD  :0099     and save it
E56D 2A8A00             LHLD  :008A     Get old startaddr char area
E570 229B00             SHLD  :009B     and save it

                * Set up area 0084-0093:

E573 118400             LXI   D,:0084   Start of variables which
                                        need offsets
E576 2E08               MVI   L,:08     Nr of pointers to be set
E578 E3         VRS10   XTHL            Get addr screen parameters
E579 79                 MOV   A,C
E57A 96                 SUB   M         Calc lobyte
E57B 12                 STAX  D         And store it in pointer
E57C 23                 INX   H         Next byte
E57D 13                 INX   D
E57E 78                 MOV   A,B
E57F 9E                 SBB   M         Calc hibyte
E580 12                 STAX  D         And store it in pointer
E581 23                 INX   H
E582 13                 INX   D
E583 E3                 XTHL
E584 2D                 DCR   L         Decr counter
E585 C278E5             JNZ   :E578     Next parameter
```



```
                * Set up area 0094-0098:

E588 E1                 POP   H         Get addr 1st parameter
E589 0605               MVI   B,:05     Nr unadjusted constant bytes
E58B 7E         VRS20   MOV   A,M       Get parameter
E58C 12                 STAX  D         and store it in pointer
E58D 23                 INX   H
E58E 13                 INX   D
E58F 05                 DCR   B         Decr counter
E590 C28BE5             JNZ   :E58B     Next parameter
E593 C32EE1             JMP   :E12E     Popall, CY=0, ret

                * If no room available:

E596 E1         VRS30   POP   H
E597 C338E1             JMP   :E138     Popall (CY=1), ret
                *
                * VECTORS TO TABLES SCREEN PARAMETERS:
                *
                * The startaddresses of the tables with para-
                * meters for the graphic modes are given.
                *
E59A 45E0       TABM    DBL   :E045     mode 1/2
E59C 6FE0               DBL   :E06F     mode 3/4
E59E 99E0               DBL   :E099     mode 5/6
                *
E5A0 5AE0       TABMA   DBL   :E05A     mode 1A/2A
E5A2 84E0               DBL   :E084     mode 3A/4A
E5A4 AEE0               DBL   :E0AE     mode 5A/6A
                *
                ***************************************
                * PERFORM MEMORY MANAGEMENT ROUTINE *
                ***************************************
                *
                * Entry: HL points to last free byte in RAM.
                *
E5A6 23         SMKRM   INX   H
E5A7 E5                 PUSH  H
E5A8 2AC400             LHLD  :00C4     Get addr mem.management
                                        routine
E5AB E3                 XTHL            Put it on stack
E5AC C9                 RET             Perform this routine and
                                        return afterwards to origin.
                                        returnaddress.
                *
                *********************************
                * SET UP AN EMPTY GRAPHICS AREA *
                *********************************
                *
                * Initialises an area of the screen into graphic
                * state and blanks it. In 16-colour modes, all
                * pixels are set 'on'. The foreground colour is
                * the first COLORG colour, the background is black.
                *
                * Entry: D:  Mode code.
                *        C:  Number of graphic lines (1-256).
                *        HL: Start of area.
                * Exit:  All registers preserved.
                *
E5AD F5         SGINIT  PUSH  PSW
E5AE C5                 PUSH  B
E5AF D5                 PUSH  D
```

ClibPDF - www.fastio.com

```
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

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

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
```

ClibPDF - www.fastio.com

```
002                             ORG   :E5FC
003                     *
004                     *
005                     *
006                     ********************************
007                     * INITIALISE HEADER / TRAILER *
008                     ********************************
009                     *
010                     * Sets up 4 background colour lines which can act
011                     * as header/trailer. Sets up colours in colour RAM.
012                     * The header/trailer area consists of 4 groups of
013                     * 4 bytes: 00 00 xx 3x, in which xx is the colour
014                     * and 3x the mode word:
015                     *       x=6: Header.
016                     *       x=F: Trailer.
017                     *       x=0: Middle area (split mode).
018                     *
019                     * Entry: HL: 1st byte header/trailer area.
020                     *        DE: Address table with required colours.
021                     *        A : Screen mode.
022                     *        B : Depth -1 in scans of each blanking
023                     *            line: 06 (header), 0F (trailer),
024                     *            00 (middle).
025                     * Exit:  HL: Points after header/trailer area.
026                     *        AFBCDE preserved.
027                     *
028 E5FC F5             SSUBL   PUSH  PSW
029 E5FD C5                     PUSH  B
030 E5FE D5                     PUSH  D
031 E5FF 4F                     MOV   C,A       Store screen mode in C
032 E600 78                     MOV   A,B
033 E601 F630                   ORI   :30       Set 4 colour to make mode
034                                             word + rept.count
035 E603 F5                     PUSH  PSW       Save mode word on stack
036 E604 0600                   MVI   B,:00
037 E606 7B                     MOV   A,E       Get lobyte addr colours
038 E607 D67C                   SUI   :7C
039 E609 CA1EE6                 JZ    :E61E     If char mode then 4 colours
040 E60C 79                     MOV   A,C       Get screen mode
041 E60D 1F                     RAR
042 E60E 1F                     RAR
043 E60F 48                     MOV   C,B
044 E610 DA1FE6                 JC    :E61F     Jump if 4-colour mode
045
046                     * 16-colour modes only:
047
048 E613 F1                     POP   PSW       Restore header/trailer info
049 E614 F680                   ORI   :80       Set 16-colour (msb=1)
050 E616 F5                     PUSH  PSW
051 E617 1A                     LDAX  D         Get colour
052 E618 87                     ADD   A         )
053 E619 87                     ADD   A         ) Move lonibble into
054 E61A 87                     ADD   A         ) hinibble
055 E61B 87                     ADD   A         )
056 E61C 06FF                   MVI   B,:FF     Set all foreground
057 E61E 4F             SBL10   MOV   C,A       Colour in C
058
059                     * Load header/trailer:
060
061 E61F F1             SBL20   POP   PSW       Get mode word
062 E620 F5                     PUSH  PSW
063 E621 77                     MOV   M,A       Load 1st byte pointer
```



```
064 E622 2B                     DCX   H         Next addr in block
065 E623 1A                     LDAX  D         Get colour info
066 E624 13                     INX   D
067 E625 77                     MOV   M,A       Load 2nd byte
068 E626 2B                     DCX   H         Next addr in block
069 E627 70                     MOV   M,B       Load 3rd byte
070 E628 2B                     DCX   H
071 E629 71                     MOV   M,C       Load 4th byte
072 E62A 2B                     DCX   H
073 E62B FEB0                   CPI   :B0       All blocks done ?
074 E62D DA1FE6                 JC    :E61F     Next one if not
075 E630 F1                     POP   PSW
076 E631 D1                     POP   D
077 E632 C1                     POP   B
078 E633 F1                     POP   PSW
079 E634 C9                     RET
080                     *
081                     ****************************
082                     * SET UP A 4-LINE TEXT AREA *
083                     ****************************
084                     *
085                     * Locates the last few lines of text on the
086                     * screen. If the screen was in split mode, the
087                     * whole contents of the old screen is located.
088                     * If it was mode 0, the last few lines above
089                     * and the cursor line are located.
090                     * The text is then moved to a required position,
091                     * including the cursor, etc.
092                     *
093                     * Entry: HL: Points to address where the text to
094                     *            be put.
095                     * Exit:  HL: Points to new top of text.
096                     *        AFBCDE preserved.
097                     *
098 E635 F5               SMVTXT  PUSH  PSW
099 E636 C5                       PUSH  B
100 E637 D5                       PUSH  D
101 E638 44                       MOV   B,H       ) New top of text in BC
102 E639 4D                       MOV   C,L       )
103 E63A 2A9B00                   LHLD  :009B     Get previous start char
104 E63D E5                       PUSH  H         on stack
105 E63E EB                       XCHG            and in DE
106 E63F 21E8FD                   LXI   H,:FDE8   Length split screen char
107                                               area
108 E642 19                       DAD   D         Calc 1st line mode byte
109                                               outside screen frame
110 E643 EB                       XCHG            in DE
111 E644 2A7200                   LHLD  :0072     Get cursor pos addr
112 E647 CDFBE6                   CALL  :E6FB     Check if cursor is still
113                                               inside frame
114 E64A DA75E6                   JC    :E675     Jump if not
115 E64D EB                       XCHG            HL is addr 1st line mode
116                                               outside screen frame
117 E64E D1                       POP   D         DE is prev start char
118 E64F E5               SMV10   PUSH  H         Save end preserved text area
119 E650 2A7200                   LHLD  :0072     Get cursor pos addr
120 E653 CD6BE3                   CALL  :E36B     Delete cursor
121 E656 E3                       XTHL            Previous start char in HL
122 E657 CDC2E6                   CALL  :E6C2     Roll screen area to new top
123                                               of text; cursor on last line
124 E65A E3                       XTHL            Cursor pos in HL
125 E65B CDF2E6                   CALL  :E6F2     Calc cursor pos against new
```

ClibPDF - www.fastio.com

```
                                        frame start
E65E 09                 DAD   B         Calc new cursor pos addr
E65F CD30E3             CALL  :E330     Keep cursor on same pos on
                                        line
E662 2A7800             LHLD  :0078     Get old start line pointer
E665 CDF2E6             CALL  :E6F2     HL=HL-DE
E668 09                 DAD   B         Calc new cursor pos
E669 CD98E6             CALL  :E698     Store addr line mode byte
                                        current line and last addr
                                        on that line
E66C E1                 POP   H         Get end preserved text area
E66D CDF2E6             CALL  :E6F2     HL=HL-DE
E670 09                 DAD   B
E671 D1                 POP   D
E672 C1                 POP   B
E673 F1                 POP   PSW
E674 C9                 RET

                * Scroll frame 1 line if cursor outside frame:

E675 E1         SMV20   POP   H
E676 2A7800             LHLD  :0078     Get startaddr cursor line
E679 117AFF             LXI   D,:FF7A
E67C 19                 DAD   D         HL: start line after cursor
E67D E5                 PUSH  H
E67E 111802             LXI   D,:0218
E681 19                 DAD   D         Subtract 4 lines and get
E682 EB                 XCHG            line mode byte in DE
E683 E1                 POP   H         Get end reqd area
E684 C34FE6             JMP   :E64F
                *
                ********************************
                * PLACE CURSOR AT BEGIN OF LINE *
                ********************************
                *
                * Sets the cursor at the beginning of a line.
                * Several pointers are updated.
                *
                * SSETC: Given a pointer to the start of line,
                *        sets start and end line variables and
                *        places the cursor at the beginning of
                *        the line.
                * SSETL: Sets only start and end line positions.
                *
                * Entry: HL: Address line mode byte current line.
                * Exit:  ABCDEHL preserved.
                *
E687 F5         SSETC   PUSH  PSW
E688 D5                 PUSH  D
E689 E5                 PUSH  H
E68A 11F8FF             LXI   D,:FFF8
E68D 19                 DAD   D         Get addr 1st data byte
                                        on current line
E68E CD30E3             CALL  :E330     Put cursor on screen
E691 AF                 XRA   A
E692 327B00             STA   :007B     No extended lines
E695 E1                 POP   H
E696 D1                 POP   D
E697 F1                 POP   PSW
E698 F5         SSETL   PUSH  PSW
E699 227800             SHLD  :0078     Store addr line mode byte
                                        current line
```



```
E69C 3E80               MVI   A,:80     ) Calc lobyte last addr
E69E 85                 ADD   L         ) on this line
E69F 327A00             STA   :007A     Store it in LNEND
E6A2 F1                 POP   PSW
E6A3 C9                 RET
                *
                ************************
                * SET GRAPHICS COLOURS *
                ************************
                *
                * Sets the colours available in a 4 colour mode
                * and the initial background in a 16 colour mode.
                *
                * Entry: HL: Points to colours vector.
                * Exit:  All registers preserved.
                *
E6A4 F5         SCOLG   PUSH  PSW
E6A5 C5                 PUSH  B
E6A6 D5                 PUSH  D
E6A7 E5                 PUSH  H
E6A8 119E00             LXI   D,:009E   Addr 1st COLORG byte
E6AB CD54E2             CALL  :E254     Set COLORG parameters
E6AE 3A9D00             LDA   :009D     Get current screen mode
E6B1 B7                 ORA   A         Check mode type
E6B2 2A8200             LHLD  :0082     Get addr after header
E6B5 F467E2             CP    :E267     If not mode 0: Load COLORG
                                        parameters in header
E6B8 1F                 RAR             Check if char mode
E6B9 2A8E00             LHLD  :008E     Get addr after trailer
E6BC D467E2             CNC   :E267     If all graphics mode:
                                        Load colours in trailer
E6BF C338E1     SGC10   JMP   :E138     Popall, ret
                *
                ********************
                * MOVE SCREEN AREA *
                ********************
                *
                * Moves a block of screen data from any position
                * to any other.
                *
                * Entry: BC: Points to highaddress target area.
                *        DE: Points to highaddress source area.
                *        HL: Points to lowaddress -1 source area.
                * Exit:  All registers preserved.
                *
E6C2 F5         MOVES   PUSH  PSW
E6C3 C5                 PUSH  B
E6C4 D5                 PUSH  D
E6C5 E5                 PUSH  H
E6C6 CDF2E6             CALL  :E6F2     Calc length of block
                                        (neg.value)
E6C9 7B                 MOV   A,E
E6CA 91                 SUB   C
E6CB 7A                 MOV   A,D
E6CC 98                 SBB   B
E6CD DAE2E6             JC    :E6E2     Jump if move up

                * Move down:

E6D0 54                 MOV   D,H       ) Length in DE
E6D1 5D                 MOV   E,L       )
E6D2 09                 DAD   B         HL = lowest targetaddr -1
```

ClibPDF - www.fastio.com

```
250 E6D3 C1              POP   B         BC = lowest sourceaddr -1
251 E6D4 C5              PUSH  B
252 E6D5 7A       MVS10  MOV   A,D
253 E6D6 B3              ORA   E
254 E6D7 CAEFE6          JZ    :E6EF     Abort if ready
255 E6DA 13              INX   D
256 E6DB 23              INX   H
257 E6DC 03              INX   B
258 E6DD 0A              LDAX  B         Get byte from source area
259 E6DE 77              MOV   M,A       and move it into target area
260 E6DF C3D5E6          JMP   :E6D5     Next one
261
262               * Move up:
263
264 E6E2 7C       MVS20  MOV   A,H
265 E6E3 B5              ORA   L
266 E6E4 CAEFE6          JZ    :E6EF     Quit if ready
267 E6E7 23              INX   H
268 E6E8 1A              LDAX  D         Get byte from source area
269 E6E9 02              STAX  B         Move it into target area
270 E6EA 0B              DCX   B
271 E6EB 1B              DCX   D
272 E6EC C3E2E6          JMP   :E6E2     Next one
273               *
274 E6EF C338E1   MVS40  JMP   :E138     Popall, ret
275               *
276               ****************
277               * HL = HL - DE *
278               ****************
279               *
280               * Entry: None.
281               * Exit:  HL=HL-DE.
282               *        Other registers preserved.
283               *
284 E6F2 F5       SUBDE  PUSH  PSW
285 E6F3 7D              MOV   A,L
286 E6F4 93              SUB   E
287 E6F5 6F              MOV   L,A       L=L-E
288 E6F6 7C              MOV   A,H
289 E6F7 9A              SBB   D
290 E6F8 67              MOV   H,A       H=H-D
291 E6F9 F1              POP   PSW
292 E6FA C9              RET
293               *
294               ********************
295               * COMPARE HL - DE *
296               ********************
297               *
298               * Compares HL with DE (HL-DE).
299               *
300               * Exit: Z=0: Not identical:
301               *            CY=0: DE < HL.
302               *            CY=1: DE > HL.
303               *       Z=1: Identical.
304               *       AF corrupted, BCDEHL preserved.
305               *
306 E6FB 7C       COMP   MOV   A,H
307 E6FC 92              SUB   D
308 E6FD C0              RNZ
309 E6FE 7D              MOV   A,L
310 E6FF 93              SUB   E
311 E700 C9              RET
```



```
312               *
313               ************************
314               * ADD OFFSET TO ADDRESS *
315               ************************
316               *
317               * Sets HL = HL + A.
318               *
319               * Entry: HL: baseaddress.
320               *        A : offset.
321               * Exit:  HL = HL + A.
322               *        BCDE preserved.
323               *
324 E701 85       DADA   ADD   L          Add lobyte addr to offset
325 E702 6F              MOV   L,A        and store it in L
326 E703 D0              RNC
327 E704 24              INR   H          Incr hibyte if overflow
328 E705 C9              RET
329               *
330               ***********************************
331               * TWO COMPLEMENT OF 16-BITS DATA *
332               ***********************************
333               *
334               * Sets HL = - HL.
335               *
336               * Entry: Data to be complemented in HL.
337               * Exit:  HL contains two-complement.
338               *        AFBCDE preserved.
339               *
340 E706 F5       CMPHL  PUSH  PSW
341 E707 7D              MOV   A,L
342 E708 2F              CMA              Compl. L
343 E709 6F              MOV   L,A        and store it
344 E70A 7C              MOV   A,H
345 E70B 2F              CMA              Compl. H
346 E70C 67              MOV   H,A        and store it
347 E70D 23              INX   H          Add 1
348 E70E F1              POP   PSW
349 E70F C9              RET
350               *
351               ****************************
352               * DRAW A DOT ON THE SCREEN *
353               ****************************
354               *
355               * Draws a single blob of a colour anywhere
356               * on the screen.
357               *
358               * Entry: C,HL: Y,X coordinate of the dot.
359               *        A:    Colour of the dot.
360               * Exit:  CY=0: O.K.
361               *        CY=1: Error code in A.
362               *        ABCDEHL preserved.
363               *
364 E710 B7       SDOT   ORA   A
365 E711 F5              PUSH  PSW
366 E712 C5              PUSH  B
367 E713 D5              PUSH  D
368 E714 E5              PUSH  H
369 E715 41              MOV   B,C        Y-coord in B
370 E716 54              MOV   D,H        ) X-coord in DE
371 E717 5D              MOV   E,L        )
372 E718 C31DE8          JMP   :E81D      Into 'SFILL'
373               *
```

ClibPDF - www.fastio.com

```
                ****************************
                * DRAW A LINE ON THE SCREEN *
                ****************************
                *
                * Draws a line in a given colour between two
                * arbitrary points on the screen.
                *
                * The coordinates are given inclusively. The
                * line will be drawn starting at the left end,
                * whichever order the parameters are given in.
                *
                * Entry: B,DE: Y,X coordinate of one end of the
                *              line.
                *        C,HL: Idem of the other end.
                *        A:    Colour of the line.
                * Exit:  CY=0: O.K.
                *        CY=1: Errorcode in A.
                *        ABCDEHL preserved.
                *
E71B B7         SDRAW   ORA   A
E71C F5                 PUSH  PSW
E71D C5                 PUSH  B
E71E D5                 PUSH  D
E71F E5                 PUSH  H
E720 CD3AE8             CALL  :E83A     Check arguments, set colour
E723 F5                 PUSH  PSW
E724 CDFBE6             CALL  :E6FB     Check direction of line
E727 3E00               MVI   A,:00     Set 'no X,Y swop'
E729 D22EE7             JNC   :E72E     Jump if X > Y

                * Swop X,Y:

                        XCHG            Exchange X coordinates
E72D 2F                 CMA
                *
E72E 32C000     DRL30   STA   :00C0     FF if X,Y swop, else 00
E731 79                 MOV   A,C
E732 E607               ANI   :07
E734 57                 MOV   D,A       Offset in field in D
E735 F1                 POP   PSW       Get Y-pos left end
E736 E5                 PUSH  H         Save X length
E737 CDB9EB             CALL  :EBB9     Pntr to start of line in
                                        screen RAM
E73A E3                 XTHL
E73B D5                 PUSH  D         Save offset
E73C E5                 PUSH  H         Save DX
E73D CD06E7             CALL  :E706     HL = - DX
E740 E3                 XTHL
E741 E5                 PUSH  H         Save DX
E742 6B                 MOV   L,E
E743 2600               MVI   H,:00
E745 29                 DAD   H
E746 22B900             SHLD  :00B9     Store 2*DY (adj long
                                        sectors)
E749 7B                 MOV   A,E       DY in A
E74A 32BD00             STA   :00BD     Set count of sectors (1-256)
E74D E1                 POP   H
E74E CD60EB             CALL  :EB60     HL = DX / DY
E751 22BB00             SHLD  :00BB     SECT is INT(DX/DY)
E754 C1                 POP   B         Get -DX
E755 E5                 PUSH  H         Save SECT
E756 7B                 MOV   A,E
```



```
E757 CD46EB             CALL  :EB46     HL = SECT*DY
E75A 09                 DAD   B         HL = SECT*DY-DX
E75B 29                 DAD   H         HL = 2*(SECT*DY-DX)
E75C 22B500             SHLD  :00B5     Store amount to add into
                                        count
E75F E1                 POP   H         Get SECT
E760 7C                 MOV   A,H
E761 1F                 RAR
E762 7D                 MOV   A,L
E763 1F                 RAR             A=INT(SECT/2)
E764 32BE00             STA   :00BE     Store amount to trim off
                                        last sector
E767 3C                 INR   A
E768 6F                 MOV   L,A
E769 2600               MVI   H,:00     HL = INIT
E76B E5                 PUSH  H
E76C 29                 DAD   H         HL = 2*INIT
E76D 7B                 MOV   A,E
E76E CD46EB             CALL  :EB46     HL = 2*INIT*DY
E771 09                 DAD   B         HL = 2*INIT*DY-DX
E772 22B700             SHLD  :00B7     Set INIT running total
E775 E1                 POP   H         Get length 1st sector
E776 F1                 POP   PSW
E777 4F                 MOV   C,A       C is initial offset
E778 7B                 MOV   A,E
E779 B7                 ORA   A
E77A C284E7             JNZ   :E784     If more than 1 sector
E77D 32BE00             STA   :00BE     Store amount to trim off
                                        last sector
E780 2ABB00             LHLD  :00BB     Get lower of 2 possible
                                        sectors
E783 23                 INX   H         Frig length if only 1 sector
E784 11BD00     DRL40   LXI   D,:00BD   Addr of nr of sectors
E787 1A                 LDAX  D         Get count of sectors
E788 D601               SUI   :01       -1
E78A 12                 STAX  D         Store it again
E78B D297E7             JNC   :E797     Jump if not last sector

                * Trim off last sector:

E78E 3ABE00             LDA   :00BE     Get amount to trim off
                                        last sector
E791 2F                 CMA
E792 5F                 MOV   E,A
E793 16FF               MVI   D,:FF
E795 13                 INX   D
E796 19                 DAD   D
                *
E797 110100     DRL50   LXI   D,:0001   Init Y-size is 1
E79A 3AC000             LDA   :00C0     ) Check if swop X,Y dir.
E79D B7                 ORA   A         ) (line > 45 degrees)
E79E CAA2E7             JZ    :E7A2     Jump if not
E7A1 EB                 XCHG            Swop X,Y direction
E7A2 43         DRL60   MOV   B,E       Get Y-size in B
E7A3 EB                 XCHG            X-size in DE
E7A4 E1                 POP   H         Get memory pointer
E7A5 05                 DCR   B
E7A6 3ABF00             LDA   :00BF     ) Check for Y-invert
E7A9 B7                 ORA   A         )
E7AA F5                 PUSH  PSW       Save condition
E7AB C4FCE7             CNZ   :E7FC     Move pntr to bottom of
                                        sector
```

ClibPDF - www.fastio.com

```
498 E7AE 1B             DCX   D         Interfacing
499 E7AF CDF7EA         CALL  :EAF7     Draw next sector of line
500 E7B2 13             INX   D         ) Re-instate real values
501 E7B3 04             INR   B         )
502 E7B4 F1             POP   PSW       Get earlier condition
503 E7B5 CABAE7         JZ    :E7BA     Jump if no Y-invert
504 E7B8 0601           MVI   B,:01     Init 1 blob down only
505 E7BA CDFCE7   DRL80 CALL  :E7FC     Move ptr up/down
506 E7BD 79             MOV   A,C       Get offset
507 E7BE 83             ADD   E         Add X-movement
508 E7BF 4F             MOV   C,A       ) Save result
509 E7C0 47             MOV   B,A       )
510 E7C1 E607           ANI   :07
511 E7C3 B9             CMP   C
512 E7C4 CAD2E7         JZ    :E7D2     Jump if not new field
513
514                 * If new field:
515
516 E7C7 4F              MOV   C,A       Update offset
517 E7C8 78             MOV   A,B       Get complement new offset
518 E7C9 A9             XRA   C         Clip bits 0,1,2 off
519 E7CA 1F             RAR             )
520 E7CB 1F             RAR             )
521 E7CC 2F             CMA             ) Update pointer
522 E7CD 5F             MOV   E,A       ) to new field
523 E7CE 16FF           MVI   D,:FF     )
524 E7D0 13             INX   D         )
525 E7D1 19             DAD   D         )
526                 *
527 E7D2 E5       DRL83 PUSH  H         Save memory pointer
528 E7D3 2AB500         LHLD  :00B5     Get amount to add into count
529 E7D6 EB             XCHG            in DE
530 E7D7 2AB700         LHLD  :00B7     Get count running total
531 E7DA 19             DAD   D         Add up
532 E7DB EB             XCHG            Result in DE
533 E7DC 2ABB00         LHLD  :00BB     Get lowest of 2 possible
534                                     sectors
535 E7DF EB             XCHG            in DE (distance to go)
536 E7E0 7C             MOV   A,H
537 E7E1 B7             ORA   A
538 E7E2 F2EDE7         JP    :E7ED     Jump if short sector
539
540                 * If long sector:
541
542 E7E5 13             INX   D         Go one blob further
543 E7E6 D5             PUSH  D
544 E7E7 EB             XCHG
545 E7E8 2AB900         LHLD  :00B9     Get adjustment for long
546                                     sectors
547 E7EB 19             DAD   D         Adjust error term
548 E7EC D1             POP   D
549 E7ED 22B700   DRL86 SHLD  :00B7     Update running total
550 E7F0 EB             XCHG
551 E7F1 3ABD00         LDA   :00BD     Get nr of sectors
552 E7F4 3C             INR   A
553 E7F5 C284E7         JNZ   :E784     Next sector if not ready
554
555                 * If ready:
556
557 E7F8 E1             POP   H
558 E7F9 C338E1         JMP   :E138     Popall, ret
559                 *
```



```
560                 *
561                 *
562 E7FC                    END
```

SYMBOL TABLE

CMPHL	E706	COMP	E6FB	DADA	E701	DRL30	E72E
DRL40	E784	DRL50	E797	DRL60	E7A2	DRL80	E7BA
DRL83	E7D2	DRL86	E7ED	MOVES	E6C2	MVS10	E6D5
MVS20	E6E2	MVS40	E6EF	SBL10	E61E	SBL20	E61F
SCOLG	E6A4	SDOT	E710	SDRAW	E71B	SGC10	E6BF
SMV10	E64F	SMV20	E675	SMVTXT	E635	SSETC	E687
SSETL	E698	SSUBL	E5FC	SUBDE	E6F2		

ClibPDF - www.fastio.com

```
002                             ORG   :E7FC
003                     *
004                     *
005                     *
006                     ***************************
007                     * MOVE POINTER UP / DOWN *
008                     ***************************
009                     *
010                     * Subroutine of SDRAW (2E71B).
011                     * Takes a pointer to screen and moves it up or
012                     * down the screen a number of lines. The move
013                     * direction depends on DIRN1 (00BF).
014                     *
015                     * Entry: HL: Pointer.
016                     *        B:  Number of lines.
017                     * Exit:  HL: updated.
018                     *        AFBCDE preserved.
019                     *
020 E7FC F5             UPDTP   PUSH  PSW
021 E7FD D5                     PUSH  D
022 E7FE E5                     PUSH  H
023 E7FF 3A9800                 LDA   :0098     Get number bytes/line
024 E802 6F                     MOV   L,A       ) Store it in HL
025 E803 2600                   MVI   H,:00     )
026 E805 78                     MOV   A,B       Get nr of lines
027 E806 CD46EB                 CALL  :EB46     Calc total length in HL
028 E809 3ABF00                 LDA   :00BF     Get Y-direction
029 E80C B7                     ORA   A         Test if up or down
030 E80D C406E7                 CNZ   :E706     If down:  Calc 2-compl of HL
031 E810 D1                     POP   D
032 E811 19                     DAD   D         Update pntr
033 E812 CDC7EA                 CALL  :EAC7     Into or out archive area
034 E815 D1                     POP   D
035 E816 F1                     POP   PSW
036 E817 C9                     RET
037                     *
038                     ****************************************
039                     * FILL A RECTANGULAR AREA ON THE SCREEN *
040                     ****************************************
041                     *
042                     * Fills an arbitrary rectangle with a given colour.
043                     *
044                     * The middle of the rectangle is filled first, then
045                     * the left and then the right edge vertical strips.
046                     * The coordinates are given inclusively.
047                     * The rectangle is filled in the same order, which
048                     * ever order the parameters are given in.
049                     *
050                     * Entry: B,DE: Y,X coordinate of one corner.
051                     *        C,HL: Idem of the opposite corner.
052                     *        A:    Colour.
053                     * Exit:  ABCDEHL preserved.
054                     *        CY=0: OK.
055                     *        CY=1: A contains error code.
056                     *
057 E818 B7             SFILL   ORA   A
058 E819 F5                     PUSH  PSW
059 E81A C5                     PUSH  B
060 E81B D5                     PUSH  D
061 E81C E5                     PUSH  H
062 E81D CD3AE8         FIL10   CALL  :E83A     Check arguments, get colour
063 E820 57                     MOV   D,A       Get Y-coord left corner
```



```
064 E821 3ABF00                 LDA   :00BF     ) Check if Y invert
065 E824 B7                     ORA   A         )
066 E825 7A                     MOV   A,D
067 E826 CA2AE8                 JZ    :E82A     Jump if no Y-inversion
068 E829 93                     SUB   E         Y-pos bottom left
069 E82A E5             FIL20   PUSH  H         Save X-size
070 E82B CDB9EB                 CALL  :EBB9     Get memory address
071 E82E 79                     MOV   A,C
072 E82F E607                   ANI   :07
073 E831 4F                     MOV   C,A       Offset in C
074 E832 43                     MOV   B,E       Height in B
075 E833 D1                     POP   D         Width in DE
076 E834 CDF7EA                 CALL  :EAF7     Fill block
077 E837 C338E1                 JMP   :E138     Popall, ret
078                     *
079                     *********************************
080                     * CHECK ARGUMENTS, GET COLOUR *
081                     *********************************
082                     *
083                     * Checks the arguments given to a entry point and
084                     * sets up the colour variables. Swops order of two
085                     * points given if necessary.
086                     *
087                     * Entry: ABCDEHL: See SFILL.
088                     *        All registers and a returnaddr on stack.
089                     * Exit:  CY=0: O.K.:
090                     *              A,BC:  Set to left corner.
091                     *              DE,HL: Set to Y,X lengths of line.
092                     *              DIRN1: <>0 if Y-direction negative.
093                     *        CY=1: Error report:
094                     *              A=1:   Off screen.
095                     *              A=2:   Colour not available.
096                     *
097 E83A CDC3E9         ARGCHK  CALL  :E9C3     Set up colour variables
098 E83D DA75E8                 JC    :E875     Jump if colour not av.
099 E840 CD7AEB                 CALL  :EB7A     Check if room available
100 E843 DA7FE8                 JC    :E87F     Jump if not
101 E846 C5                     PUSH  B
102 E847 48                     MOV   C,B       Y-coord one point in C
103 E848 EB                     XCHG
104 E849 CD7AEB                 CALL  :EB7A     Check if room available
105 E84C EB                     XCHG
106 E84D C1                     POP   B
107 E84E DA7FE8                 JC    :E87F     Jump if not
108 E851 CDFBE6                 CALL  :E6FB     Compare HL-DE
109 E854 D25DE8                 JNC   :E85D     Right most point to C,HL
110
111                     * Swop Y-coord.:
112
113 E857 EB                     XCHG
114 E858 F5                     PUSH  PSW
115 E859 78                     MOV   A,B       )
116 E85A 41                     MOV   B,C       ) Swop 2 points
117 E85B 4F                     MOV   C,A       )
118 E85C F1                     POP   PSW
119                     *
120 E85D CDF2E6         ARC10   CALL  :E6F2     Calc horizontal length
121 E860 D5                     PUSH  D         Save X-pos left corner
122 E861 79                     MOV   A,C
123 E862 90                     SUB   B
124 E863 1600                   MVI   D,:00     Clear Y-invert flag
125 E865 D26BE8                 JNC   :E86B     Jump if end above start
```

ClibPDF - www.fastio.com

```
126 E868 2F              CMA
127 E869 3C              INR   A
128 E86A 15              DCR   D
129                      *
130 E86B 5F              ARC20 MOV   E,A       Get vertical length
131 E86C 7A                    MOV   A,D
132 E86D 32BF00                STA   :00BF     Set Y-invert flag
133 E870 1600                  MVI   D,:00
134 E872 78                    MOV   A,B       Get Y-pos left corner
135 E873 C1                    POP   B         Get X-pos left corner
136 E874 C9                    RET
137
138                      * If colour error:
139
140 E875 3E02            ARC90 MVI   A,:02     Error colour not available
141 E877 E1              ARC98 POP   H
142 E878 E1                    POP   H
143 E879 D1                    POP   D
144 E87A C1                    POP   B
145 E87B 33                    INX   SP
146 E87C 33                    INX   SP
147 E87D 37                    STC             Return error
148 E87E C9                    RET
149
150                      * If off screen error:
151
152 E87F 3E01            ARC99 MVI   A,:01     Error off screen
153 E881 C377E8                JMP   :E877     Abort
154                      *
155                      ******************************************
156                      * ASK COLOUR OF A POINT ON THE SCREEN AND *
157                      *     ASK SIZE OF THE GRAPHICS SCREEN     *
158                      ******************************************
159                      *
160                      * Aks the colour of a given point on the screen
161                      * and the size of the graphics area of the screen.
162                      *
163                      * Entry: C,HL: Y,X coordinate of the dot required.
164                      * Exit:  CY=0: O.K.:
165                      *              A:    Colour at requested point.
166                      *              B,DE: Max. coordinates of the
167                      *                    graphics area.
168                      *              CHL preserved.
169                      *        CY=1: A:    Error code.
170                      *              BCDEHL preserved.
171                      *
172 E884 E5              SSCRN PUSH  H
173 E885 C5                    PUSH  B
174 E886 CD7AEB                CALL  :EB7A     Check if room available
175 E889 DAD8E8                JC    :E8D8     Jump if not
176 E88C 7D                    MOV   A,L
177 E88D E607                  ANI   :07
178 E88F F5                    PUSH  PSW       Remember field offset
179 E890 79                    MOV   A,C       )
180 E891 44                    MOV   B,H       ) Coord in A,B,C
181 E892 4D                    MOV   C,L       )
182 E893 CDB9EB                CALL  :EBB9     Get address of point
183 E896 3A9D00                LDA   :009D     Get current screen mode
184 E899 1F                    RAR
185 E89A 1F                    RAR
186 E89B DAB8E8                JC    :E8B8     Jump if 4-colour mode
187
```



```
188                      * If 16-colour mode:
189
190 E89E CDF6E8                CALL  :E8F6     Colours to buffer
191 E8A1 21A300                LXI   H,:00A3   Addr SCXBUF
192 E8A4 F1                    POP   PSW
193 E8A5 CD01E7                CALL  :E701     Calc addr in buffer
194 E8A8 7E                    MOV   A,M       Get colour for reqd blob
195 E8A9 2A9600          SSC10 LHLD  :0096     Get nr of graphics lines
196 E8AC 45                    MOV   B,L
197 E8AD 05                    DCR   B         lobyte -1 in B
198 E8AE 2A9400                LHLD  :0094     Get nr of hor. blobs
199 E8B1 2B                    DCX   H         -1
200 E8B2 EB                    XCHG            in DE
201 E8B3 E1                    POP   H
202 E8B4 4D                    MOV   C,L       Y-coord in C
203 E8B5 E1                    POP   H         X-coord in HL
204 E8B6 B7                    ORA   A         CY=0
205 E8B7 C9                    RET
206
207                      * If 4-colour mode:
208
209 E8B8 56              SSC30 MOV   D,M       )
210 E8B9 2B                    DCX   H         ) Get screen data of
211 E8BA 5E                    MOV   E,M       ) point in DE
212 E8BB F1                    POP   PSW
213 E8BC 4F                    MOV   C,A       Field offset in C
214 E8BD 0601                  MVI   B,:01
215 E8BF CDE1EB                CALL  :EBE1     Set mask for bits
216 E8C2 219E00                LXI   H,:009E   Pntr to COLORG colours
217 E8C5 78                    MOV   A,B
218 E8C6 A2                    ANA   D         Test top bit result
219 E8C7 CACCE8                JZ    :E8CC     Skip if 0
220 E8CA 23                    INX   H
221 E8CB 23                    INX   H
222 E8CC 78              SSC40 MOV   A,B
223 E8CD A3                    ANA   E         Test bottom bit result
224 E8CE CAD2E8                JZ    :E8D2     Skip if 0
225 E8D1 23                    INX   H
226 E8D2 7E              SSC50 MOV   A,M       Get result from table
227 E8D3 E60F                  ANI   :0F       Colour bits only
228 E8D5 C3A9E8                JMP   :E8A9
229
230                      * If off screen:
231
232 E8D8 E1              SSC99 POP   H
233 E8D9 C1                    POP   B
234 E8DA 3E01                  MVI   A,:01     Error 'off screen'
235 E8DC 37                    STC
236 E8DD C9                    RET
237                      *
238                      ******************
239                      * UPDATE A FIELD *
240                      ******************
241                      *
242                      * Given a mask of bits to be changed, a colour to
243                      * set them to and a memory address where the field
244                      * starts. This routine reads, updates and replaces
245                      * a field.
246                      *
247                      * Entry: HL: Memory address of start of field.
248                      *         B: Mask of bits to be changed.
249                      *         C: Colour to change to (hinibble).
```

ClibPDF - www.fastio.com

```
250                     * Exit:  All registers preserved.
251                     *
252 E8DE F5             SUPDTE  PUSH  PSW
253 E8DF C5                     PUSH  B
254 E8E0 D5                     PUSH  D
255 E8E1 E5                     PUSH  H
256 E8E2 79                     MOV   A,C       )
257 E8E3 0F                     RRC             )
258 E8E4 0F                     RRC             ) Move hinibble C into
259 E8E5 0F                     RRC             ) lonibble
260 E8E6 0F                     RRC             )
261 E8E7 4F                     MOV   C,A       )
262 E8E8 C5                     PUSH  B
263 E8E9 CDF6E8                 CALL  :E8F6     Get current state of screen
264 E8EC C1                     POP   B
265 E8ED CDB2E9                 CALL  :E9B2     Change as required
266 E8F0 E1                     POP   H
267 E8F1 E5                     PUSH  H
268 E8F2 C3BAC6                 JMP   :C6BA     Set up screen bits for
269                                             mode 1
270                     *
271 E8F5 FF                     DATA  :FF
272                     *
273                     ***********************************
274                     * LOAD BUFFER SCXBUF FROM SCREEN *
275                     ***********************************
276                     *
277                     * Takes 2 bytes of screen info in 16-colour mode
278                     * and places them in SCXBUF (00A3-AB) in 'standard
279                     * form'.
280                     *
281                     * Entry: HL: Points to 1st byte of info on screen.
282                     * Exit:  All registers corrupted.
283                     *
284 E8F6 23             SSFM    INX   H
285 E8F7 7E                     MOV   A,M       Get previous colour byte
286 E8F8 E60F                   ANI   :0F       Background only
287 E8FA 4F                     MOV   C,A       Previous background in C
288 E8FB 2B                     DCX   H
289 E8FC 56                     MOV   D,M       Select byte in D
290 E8FD 2B                     DCX   H
291 E8FE 5E                     MOV   E,M       Colour byte in E
292 E8FF 2B                     DCX   H
293 E900 E5                     PUSH  H         Save pntr to next field
294                                             select byte
295 E901 7B                     MOV   A,E       Colour byte in A
296 E902 E6F0                   ANI   :F0       )
297 E904 0F                     RRC             )
298 E905 0F                     RRC             ) Foreground colour
299 E906 0F                     RRC             ) in lonibble
300 E907 0F                     RRC             )
301 E908 47                     MOV   B,A       Foreground colour in B
302 E909 21A300                 LXI   H,:00A3   Addr SCXBUF
303 E90C 7A                     MOV   A,D       Get bit mask
304 E90D 1608                   MVI   D,:08     8 bytes to set
305 E90F 07             SSF10   RLC
306 E910 71                     MOV   M,C       Set background
307 E911 D21EE9                 JNC   :E91E     Jump if background
308 E914 70                     MOV   M,B       Else: set foreground
309 E915 F5                     PUSH  PSW
310 E916 7B                     MOV   A,E       Get colour byte
311 E917 E60F                   ANI   :0F       Background colour only
```



```
312 E919 4F                     MOV   C,A       Background is current BG
313 E91A 32AC00                 STA   :00AC     Store it as colour carried
314                                             out to next field
315 E91D F1                     POP   PSW
316 E91E 23             SSF20   INX   H         Next pos in SCXBUF
317 E91F 15                     DCR   D         Count -1
318 E920 C20FE9                 JNZ   :E90F     Loop if more bits
319 E923 C1                     POP   B         Get pntr to next field
320                                             select byte
321 E924 36FF                   MVI   M,:FF     Flag no carry out in SBGOU
322 E926 0A                     LDAX  B         Get next select byte
323 E927 34             SSF30   INR   M         Set 'carry out' flag
324 E928 07                     RLC
325 E929 D227E9                 JNC   :E927     Loop counting carry out
326 E92C C9                     RET
327                     *
328                     ***********************************
329                     * SET UP SCREEN BITS FOR MODE 1 *
330                     ***********************************
331                     *
332                     * Takes the 8 blobs represented in standard form
333                     * in SCXBUF, and tries to represent them in a way
334                     * which the screen requires for mode 1.
335                     * Up to 2 colours is easy. 3 require to attempt
336                     * to carry in the 1st colour from the previous
337                     * byte.
338                     *
339                     * Entry: HL: Points to 1st of the 2 screen bytes.
340                     * Exit:  Screen will be updated as well as
341                     *        possible.
342                     *        All registers preserved.
343                     *
344                     SBF00
345 E92D 23             SBFM    INX   H
346 E92E 7E                     MOV   A,M
347 E92F F680                   ORI   :80
348 E931 5F                     MOV   E,A       Prev background in E
349 E932 1600                   MVI   D,:00     Init bit map
350 E934 21A300                 LXI   H,:00A3   Addr SCXBUF
351 E937 00                     NOP
352 E938 00                     NOP
353 E939 3AAB00         SBF05   LDA   :00AB     Get flag for colour carried
354                                             out
355 E93C B7                     ORA   A
356 E93D CA45E9                 JZ    :E945     Jump if no carry out
357 E940 3AAC00                 LDA   :00AC     Get colour carried out
358 E943 F680                   ORI   :80       Set msb=1
359 E945 4F             SBF10   MOV   C,A       Background colour in C
360 E946 7E                     MOV   A,M
361 E947 23                     INX   H
362 E948 47                     MOV   B,A       Set 1st blob as FG colour
363 E949 7A                     MOV   A,D       )
364 E94A 37                     STC             ) 1 bit in right end bit
365 E94B 17                     RAL             ) mask
366 E94C 57                     MOV   D,A       )
367 E94D 7E             SBF30   MOV   A,M
368 E94E 23                     INX   H
369 E94F B8                     CMP   B         Is next blob FG colour ?
370 E950 37                     STC
371 E951 CA61E9                 JZ    :E961     Jump if true
372 E954 F680                   ORI   :80
373 E956 B9                     CMP   C         Test if same as BG ?
```

ClibPDF - www.fastio.com

```
374 E957 CA60E9          JZ    :E960     Jump if true
375 E95A 0D              DCR   C
376 E95B 0C              INR   C
377 E95C FA86E9          JM    :E986     No luck if BG used already
378 E95F 4F              MOV   C,A       Else set it to BG
379 E960 B7        SBF40 ORA   A
380 E961 7A        SBF45 MOV   A,D       )
381 E962 17              RAL             ) New bit in bottom of mask
382 E963 57              MOV   D,A       )
383 E964 7D        SBF50 MOV   A,L
384 E965 FEAB            CPI   :AB       End of buffer reached ?
385 E967 C24DE9          JNZ   :E94D     Loop until all set up
386 E96A E1              POP   H
387 E96B E5              PUSH  H
388 E96C 23              INX   H
389 E96D 7B              MOV   A,E
390 E96E E60F            ANI   :0F
391 E970 5F              MOV   E,A       Only lonibble of E
392 E971 7E              MOV   A,M
393 E972 E6F0            ANI   :F0       Only hinibble of M
394 E974 B3              ORA   E
395 E975 77              MOV   M,A       Add both nibbles together
396 E976 2B              DCX   H
397 E977 78              MOV   A,B
398 E978 87              ADD   A
399 E979 87              ADD   A
400 E97A 87              ADD   A
401 E97B 87              ADD   A         FG colour to top bits
402 E97C 47              MOV   B,A
403 E97D 79              MOV   A,C
404 E97E E60F            ANI   :0F       Low nibble only
405 E980 B0              ORA   B
406 E981 72              MOV   M,D       Bit map from D
407 E982 2B              DCX   H
408 E983 77              MOV   M,A       Colours from E
409 E984 E1        SBF90 POP   H
410 E985 C9              RET
411
412                * 3 colours needed:
413
414 E986 7B        SBF80 MOV   A,E
415 E987 B7              ORA   A
416 E988 F284E9          JP    :E984     Jump if tried BG carried in
417 E98B E60F            ANI   :0F       Previous BG
418 E98D B8              CMP   B         Test against 1st blob colour
419 E98E E1              POP   H
420 E98F E5              PUSH  H
421 E990 23              INX   H
422 E991 23              INX   H
423 E992 7E              MOV   A,M       Get bit map
424 E993 37        SBF83 STC
425 E994 17              RAL
426 E995 D293E9          JNC   :E993     Ignore leading BG
427 E998 CAA1E9          JZ    :E9A1     Jump if colour matches
428                                      anyway
429 E99B 3C              INR   A
430 E99C C284E9          JNZ   :E984     No good if BG used
431
432                * Background not in use:
433
434 E99F 58               MOV   E,B      Set previous BG
435 E9A0 00              NOP
```



```
436 E9A1 21A300    SBF85 LXI   H,:00A3   Addr SCXBUF
437 E9A4 1600            MVI   D,:00     Init D
438 E9A6 00              NOP
439 E9A7 4A              MOV   C,D       and C (BG free)
440 E9A8 7E        SBF88 MOV   A,M       Get byte from SCXBUF
441 E9A9 B8              CMP   B
442 E9AA C239E9          JNZ   :E939     Jump if blob not old BG
443                                      colour
444 E9AD 00              NOP
445 E9AE 23              INX   H
446 E9AF C3A8E9          JMP   :E9A8     Next blob
447                *
448                ************************
449                * UPDATE BUFFER SCXBUF *
450                ************************
451                *
452                * Takes a set of 'update instructions' in BC and
453                * sets various bytes in SCXBUF accordingly.
454                *
455                * Entry: BC: Instructions.
456                * Exit:  All registers corrupted.
457                *
458 E9B2 21A300    SUDCH LXI   H,:00A3   Addr SCXBUF
459 E9B5 78              MOV   A,B       Mask in A
460 E9B6 0608            MVI   B,:08     8 byte to be done
461 E9B8 07        SUD10 RLC             Bit from mask into CY
462 E9B9 D2BDE9          JNC   :E9BD     Jump if bit = 0
463 E9BC 71              MOV   M,C       Else: C into SCXBUF
464 E9BD 23        SUD20 INX   H
465 E9BE 05              DCR   B
466 E9BF C2B8E9          JNZ   :E9B8     Next byte if not ready
467 E9C2 C9              RET
468                *
469                ****************************
470                * SET UP COLOUR VARIABLES *
471                ****************************
472                *
473                * Entry: A: Colour.
474                * Exit:  CY=1: Colour not available.
475                *        CY=0: O.K.; ABCDEHL preserved.
476                *
477 E9C3 B7        COLSU ORA   A
478 E9C4 F5              PUSH  PSW
479 E9C5 C5              PUSH  B
480 E9C6 4F              MOV   C,A       Colour in C
481 E9C7 AF              XRA   A
482 E9C8 32C100          STA   :00C1     Reset animate flag
483 E9CB 3A9D00          LDA   :009D     Get current screen mode
484 E9CE 1F              RAR
485 E9CF 1F              RAR
486 E9D0 D2FDE9          JNC   :E9FD     Jump if 16-colour
487
488                * If 4-colour:
489
490 E9D3 79              MOV   A,C       Get colour
491 E9D4 FE10            CPI   :10
492 E9D6 DAE1E9          JC    :E9E1     Jump if < 16
493 E9D9 CD86D8          CALL  :D886     Check if >= 20. Set 00C1
494                                      for 4-colour animate if not
495 E9DC E603      CSU04 ANI   :03
496 E9DE C3E7E9          JMP   :E9E7     Bottom 3 bits only
497 E9E1 CD9BEB    CSU05 CALL  :EB9B     Find colour in COLORG reg
```

ClibPDF - www.fastio.com

```
498                                         (2 bit code)
499 E9E4 D207EA          JNC   :EA07        Jump if colour not av.
500 E9E7 0600      CSU08 MVI   B,:00
501 E9E9 FE02            CPI   :02
502 E9EB DAEFE9          JC    :E9EF        Jump if top bit 0
503 E9EE 05              DCR   B            Set 00/FF on top bit
504 E9EF E601      CSU10 ANI   :01
505 E9F1 2F              CMA
506 E9F2 3C              INR   A            Set 00/FF on bottom bit
507 E9F3 32C200    CSU30 STA   :00C2        )
508 E9F6 78              MOV   A,B          ) Store details for colour
509 E9F7 32C300          STA   :00C3        ) reqd
510 E9FA C1              POP   B
511 E9FB F1              POP   PSW
512 E9FC C9              RET
513
514                * If 16-colour:
515
516 E9FD 79        CSU40 MOV   A,C          Get colour
517 E9FE 87              ADD   A            )
518 E9FF 87              ADD   A            ) SHL 8
519 EA00 87              ADD   A            )
520 EA01 87              ADD   A            )
521 EA02 06FF            MVI   B,:FF
522 EA04 C3F3E9          JMP   :E9F3        Store details for colour
523                                         reqd
524
525                * If colour not found:
526
527 EA07 C1        CSU99 POP   B
528 EA08 F1              POP   PSW
529 EA09 3F              CMC                Quit; 'colour not available'
530 EA0A C9              RET
531                *
532                *
533                *
534 EA0B                 END

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

ARC10  E85D   ARC20  E86B   ARC90  E875   ARC98  E877
ARC99  E87F   ARGCHK E83A   COLSU  E9C3   CSU04  E9DC
CSU05  E9E1   CSU08  E9E7   CSU10  E9EF   CSU30  E9F3
CSU40  E9FD   CSU99  EA07   FIL10  E81D   FIL20  E82A
SBF00  E92D   SBF05  E939   SBF10  E945   SBF30  E94D
SBF40  E960   SBF45  E961   SBF50  E964   SBF80  E986
SBF83  E993   SBF85  E9A1   SBF88  E9A8   SBF90  E984
SBFM   E92D   SFILL  E818   SSC10  E8A9   SSC30  E8B8
SSC40  E8CC   SSC50  E8D2   SSC99  E8D8   SSCRN  E884
SSF10  E90F   SSF20  E91E   SSF30  E927   SSFM   E8F6
SUD10  E9B8   SUD20  E9BD   SUDCH  E9B2   SUPDTE E8DE
UPDTP  E7FC
```



```
002                      ORG   :EA0B
003                *
004                *
005                *
006                **************
007                * FILL BLOCK *
008                **************
009                *
010                * Fills a rectangular block of whole fields with
011                * one colour.
012                *
013                * Entry: HL:    Address bottom left corner.
014                *        DE:    Y,X-counts of size of block (E in
015                *               fields).
016                *        FCOLR: Colour info.
017                * Exit:  BCDEHL preserved. AF corrupted.
018                *
019 EA0B C5        FILBK  PUSH  B
020 EA0C D5               PUSH  D
021 EA0D E5               PUSH  H
022 EA0E 3AC200           LDA   :00C2      ) Get details for colour
023 EA11 4F               MOV   C,A        ) required in BC
024 EA12 3AC300           LDA   :00C3      )
025 EA15 47               MOV   B,A        )
026 EA16 1C               INR   E          Count range up by 1
027 EA17 D5        FBK10  PUSH  D
028 EA18 E5               PUSH  H
029 EA19 3AC100    FBK20  LDA   :00C1      Get animate flag
030 EA1C B7               ORA   A
031 EA1D C246EA           JNZ   :EA46      Jump if set
032 EA20 3A9D00           LDA   :009D      Get current screen mode
033 EA23 1F               RAR
034 EA24 1F               RAR
035 EA25 70               MOV   M,B        Colour details in screen RAM
036 EA26 2B               DCX   H
037 EA27 D23EEA           JNC   :EA3E      Jump if 16-colour mode
038
039                * If 4-colour mode:
040
041 EA2A 71               MOV   M,C        Colour details in screen RAM
042                *
043 EA2B 2B        FBK30  DCX   H
044 EA2C 1D               DCR   E
045 EA2D C219EA           JNZ   :EA19      Loop to do all fields
046 EA30 E1               POP   H          HL pnts to left of rectangle
047 EA31 D1               POP   D          Get Y-size count
048 EA32 15               DCR   D
049 EA33 14               INR   D
050 EA34 CA53EA           JZ    :EA53      Abort if ready
051 EA37 15               DCR   D          Update Y-count
052 EA38 CDC1EA           CALL  :EAC1      Update pntr to next line
053 EA3B C317EA           JMP   :EA17      Next line
054
055                * If 16-colour mode:
056
057 EA3E 7E        FBK40  MOV   A,M        Get data from screen RAM
058 EA3F E60F             ANI   :0F        Lonibble only (old BG)
059 EA41 B1               ORA   C          Add details
060 EA42 77               MOV   M,A        Preserve old background
061 EA43 C32BEA           JMP   :EA2B
062
063                * If animate:
```

ClibPDF - www.fastio.com

```
064
065 EA46 E5          FBK50   PUSH  H
066 EA47 05                  DCR   B
067 EA48 04                  INR   B
068 EA49 C24DEA              JNZ   :EA4D
069 EA4C 2B                  DCX   H
070 EA4D 71          FBK60   MOV   M,C       Change whole field
071 EA4E E1                  POP   H
072 EA4F 2B                  DCX   H
073 EA50 C32BEA              JMP   :EA2B     Next field
074                  *
075 EA53 E1          FBK90   POP   H
076 EA54 D1                  POP   D
077 EA55 C1                  POP   B
078 EA56 C9                  RET
079                  *
080                  **************
081                  * FILL STRIP *
082                  **************
083                  *
084                  * Fills a vertical strip on the screen with one
085                  * colour.
086                  *
087                  * Entry: HL: Points to bottom field of strip.
088                  *        B:  Mask bits to change.
089                  *        D:  Heigth -1 of strip.
090                  * Exit:  AF currupted, BCDEHL preserved.
091                  *
092 EA57 C5          FILST   PUSH  B
093 EA58 D5                  PUSH  D
094 EA59 E5                  PUSH  H
095 EA5A 3A9D00              LDA   :009D     Get current screen mode
096 EA5D 1F                  RAR
097 EA5E 1F                  RAR
098 EA5F D2A9EA              JNC   :EAA9     Jump if 16-colour mode
099
100                  * If 4-colour mode:
101
102 EA62 D5                  PUSH  D
103 EA63 EB                  XCHG
104 EA64 2AC200              LHLD  :00C2     Get details for colour reqd
105 EA67 3AC100              LDA   :00C1     Get animate flag
106 EA6A B7                  ORA   A
107 EA6B C291EA              JNZ   :EA91     Jump if set
108 EA6E 78                  MOV   A,B       )
109 EA6F A4                  ANA   H         )
110 EA70 4F                  MOV   C,A       ) Mask for bits to be update
111 EA71 78                  MOV   A,B       )
112 EA72 A5                  ANA   L         )
113 EA73 6F                  MOV   L,A       )
114 EA74 78                  MOV   A,B
115 EA75 2F                  CMA             Bits to be preserved
116 EA76 47                  MOV   B,A
117 EA77 67                  MOV   H,A
118 EA78 EB          FST05   XCHG
119 EA79 78          FST10   MOV   A,B
120 EA7A A6                  ANA   M         Pick up old colours
121 EA7B B1                  ORA   C
122 EA7C 77                  MOV   M,A       Update top bits
123 EA7D 2B                  DCX   H
124 EA7E 7A                  MOV   A,D
125 EA7F A6                  ANA   M
```

ClibPDF - www.fastio.com


```
126 EA80 B3                  ORA   E
127 EA81 77                  MOV   M,A       Update bottom bits
128 EA82 23                  INX   H
129 EA83 E3                  XTHL
130 EA84 7C                  MOV   A,H
131 EA85 25                  DCR   H
132 EA86 B7                  ORA   A
133 EA87 CABCEA              JZ    :EABC     Jump if ready
134 EA8A E3                  XTHL
135 EA8B CDC1EA              CALL  :EAC1     Update pointer
136 EA8E C379EA              JMP   :EA79     Next line
137
138                  * If animate:
139
140 EA91 E5          FST15   PUSH  H         Preserve colour details
141 EA92 78                  MOV   A,B
142 EA93 A5                  ANA   L
143 EA94 4F                  MOV   C,A       Bits to be set in C
144 EA95 78                  MOV   A,B
145 EA96 2F                  CMA
146 EA97 B5                  ORA   L
147 EA98 47                  MOV   B,A       To be set
148 EA99 2E00                MVI   L,:00
149 EA9B 26FF                MVI   H,:FF     For other byte
150 EA9D F1                  POP   PSW       Get colour details
151 EA9E B7                  ORA   A
152 EA9F C2A6EA              JNZ   :EAA6
153 EAA2 C5                  PUSH  B
154 EAA3 E5                  PUSH  H
155 EAA4 C1                  POP   B
156 EAA5 E1                  POP   H
157 EAA6 C378EA      FST18   JMP   :EA78
158
159                  * If 16-colour mode:
160
161 EAA9 3AC200      FST20   LDA   :00C2     Get 1 byte of details colour
162                                          required
163 EAAC 4F                  MOV   C,A       Set colour as reqd
164 EAAD CDDEE8      FST30   CALL  :E8DE     Update
165 EAB0 7A                  MOV   A,D
166 EAB1 15                  DCR   D
167 EAB2 B7                  ORA   A
168 EAB3 CABDEA              JZ    :EABD
169 EAB6 CDC1EA              CALL  :EAC1     Next line
170 EAB9 C3ADEA              JMP   :EAAD     Next field
171
172                  * If ready:
173
174 EABC E1          FST90   POP   H
175 EABD E1          FST91   POP   H
176 EABE D1                  POP   D
177 EABF C1                  POP   B
178 EAC0 C9                  RET
179                  *
180                  ***************************
181                  * MOVE AND CHECK POINTER *
182                  ***************************
183                  *
184                  * DADCK: Moves a pointer 1 line up screen and
185                  *        offsets to alternate area if necessary.
186                  * PTRCK: Checks a memory pointer and moves it
187                  *        into or out of the archive area if
```

```
                *       necessary.
                *
                * Exit:  HL: New pointer.
                *
EAC1 3A9800     DADCK   LDA   :0098     Get nr bytes/line
EAC4 CD01E7             CALL  :E701     Add 1 line length
EAC7 C5         PTRCK   PUSH  B
EAC8 D5                 PUSH  D
EAC9 EB                 XCHG
EACA 2A8800             LHLD  :0088     Get addr after end graphics
                                        area
EACD CDFBE6             CALL  :E6FB     Compare HL-DE
EAD0 2A8400             LHLD  :0084     Get bottom archive area
EAD3 44                 MOV   B,H       ) in BC
EAD4 4D                 MOV   C,L       )
EAD5 2A8200             LHLD  :0082     Get top visible area
EAD8 D2E5EA             JNC   :EAE5     Jump if pntr is below
                                        visible screen
EADB CDFBE6             CALL  :E6FB     Compare HL-DE
EADE DA0FD8             JC    :D80F     Jump if pntr is off top
                                        visible screen
EAE1 EB         PCK10   XCHG
EAE2 D1         PCK15   POP   D
EAE3 C1                 POP   B
EAE4 C9                 RET

                * If pntr is below visible screen:

EAE5 E5         PCK20   PUSH  H         )
EAE6 C5                 PUSH  B         ) Swop BC and HL
EAE7 E1                 POP   H         )
EAE8 C1                 POP   B         )
EAE9 CDFBE6             CALL  :E6FB     Compare HL-DE
EAEC DAE1EA             JC    :EAE1     Jump if within archive area
EAEF EB         PCK30   XCHG
EAF0 CDF2E6             CALL  :E6F2     Subtract nearest boundary
EAF3 09                 DAD   B         Add other
EAF4 C3E2EA             JMP   :EAE2
                *
                ****************************
                * FILL A RECTANGULAR AREA *
                ****************************
                *
                * Entry: DE: Width.
                *        B : Height.
                *        C : Offset.
                *        HL: Address.
                * Exit: All registers preserved.
                *
EAF7 F5         FILRT   PUSH  PSW
EAF8 C5                 PUSH  B
EAF9 D5                 PUSH  D
EAFA E5                 PUSH  H
EAFB E5                 PUSH  H
EAFC 79                 MOV   A,C
EAFD 83                 ADD   E
EAFE 67                 MOV   H,A       H = C + E
EAFF 3E00               MVI   A,:00
EB01 8A                 ADC   D
EB02 50                 MOV   D,B
EB03 1F                 RAR
EB04 7C                 MOV   A,H
```



```
EB05 DA0DEB             JC    :EB0D
EB08 FE08               CPI   :08
EB0A DA40EB             JC    :EB40     If > 8: Fill only one strip
EB0D 1F         L2E152  RAR
EB0E 0F                 RRC
EB0F E67E               ANI   :7E
EB11 E3                 XTHL
EB12 F5                 PUSH  PSW
EB13 0F                 RRC
EB14 D602               SUI   :02
EB16 5F                 MOV   E,A
EB17 2B                 DCX   H
EB18 2B                 DCX   H
EB19 D40BEA             CNC   :EA0B     Fill block
EB1C 23                 INX   H
EB1D 23                 INX   H
EB1E 79                 MOV   A,C
EB1F D609               SUI   :09
EB21 2F                 CMA
EB22 47                 MOV   B,A
EB23 CDE1EB             CALL  :EBE1     Set mask for bits
EB26 CD57EA             CALL  :EA57     Fill strip
EB29 F1                 POP   PSW
EB2A 2F                 CMA
EB2B 4F                 MOV   C,A
EB2C 06FF               MVI   B,:FF
EB2E 03                 INX   B
EB2F 09                 DAD   B
EB30 F1                 POP   PSW
EB31 E607               ANI   :07
EB33 3C                 INR   A
EB34 47                 MOV   B,A
EB35 0E00               MVI   C,:00
EB37 CDE1EB     L2E153  CALL  :EBE1     Set mask for bits
EB3A CD57EA             CALL  :EA57     Fill strip
EB3D C338E1             JMP   :E138     Popall, ret

                * If < 1 field:

EB40 43         L2E154  MOV   B,E
EB41 04                 INR   B
EB42 E1                 POP   H
EB43 C337EB             JMP   :EB37     Fill a strip only
                *
                ************************
                * CALCULATE HL = A * HL *
                ************************
                *
                * Exit: AFBCDE preserved.
                *
EB46 F5         HLMUL   PUSH  PSW
EB47 D5                 PUSH  D
EB48 EB                 XCHG            Original HL in DE
EB49 210000             LXI   H,:0000   Init result
EB4C B7         L2E156  ORA   A
EB4D CA5DEB             JZ    :EB5D     Abort if ready
EB50 1F                 RAR             )
EB51 F5                 PUSH  PSW       )
EB52 D256EB             JNC   :EB56     ) Calc HL = A * HL
EB55 19                 DAD   D         )
EB56 EB         L2E157  XCHG            )
EB57 29                 DAD   H         )
```

ClibPDF - www.fastio.com

```
312 EB58 EB                 XCHG            )
313 EB59 F1                 POP   PSW       )
314 EB5A C34CEB             JMP   :EB4C     Cont multiplication
315 EB5D D1           L2E158 POP   D
316 EB5E F1                 POP   PSW
317 EB5F C9                 RET
318                   *
319                   *************************
320                   * CALCULATE HL = HL / A *
321                   *************************
322                   *
323                   * Exit: AFBCDE preserved.
324                   *
325 EB60 F5           HLDIV  PUSH  PSW
326 EB61 B7                 ORA   A
327 EB62 CA78EB             JZ    :EB78     Abort if A=0
328 EB65 C5                 PUSH  B
329 EB66 D5                 PUSH  D
330 EB67 01FFFF             LXI   B,:FFFF
331 EB6A 2F                 CMA
332 EB6B 5F                 MOV   E,A
333 EB6C 16FF               MVI   D,:FF
334 EB6E 13                 INX   D
335 EB6F 19           L2E160 DAD   D
336 EB70 03                 INX   B
337 EB71 DA6FEB             JC    :EB6F
338 EB74 69                 MOV   L,C       ) Result in HL
339 EB75 60                 MOV   H,B       )
340 EB76 D1                 POP   D
341 EB77 C1                 POP   B
342 EB78 F1           L2E161 POP   PSW
343 EB79 C9                 RET
344                   *
345                   *********************************************
346                   * CHECK IF SUFFICIENT ROOM FOR GRAPHIC MODE *
347                   *********************************************
348                   *
349                   * Exit: CY=1: Insufficient room for mode.
350                   *       CY=0: O.K.
351                   *       ABCDEHL preserved.
352                   *
353 EB7A C5           TPOSN  PUSH  B
354 EB7B F5                 PUSH  PSW
355 EB7C D5                 PUSH  D
356 EB7D 3A9D00             LDA   :009D     Get current screen mode
357 EB80 C601               ADI   :01
358 EB82 DA96EB             JC    :EB96     If mode 0: Abort CY=1
359 EB85 EB                 XCHG
360 EB86 2A9400             LHLD  :0094     Get nr of hor. blobs
361 EB89 2B                 DCX   H         minus 1
362 EB8A CDFBE6             CALL  :E6FB     Compare HL-DE
363 EB8D EB                 XCHG
364 EB8E DA96EB             JC    :EB96     Abort CY=1 if insufficient
365                                         room
366 EB91 3A9600             LDA   :0096     Get nr of graphics lines
367 EB94 3D                 DCR   A         minus 1
368 EB95 B9                 CMP   C         Set flags on difference
369 EB96 D1           L2E163 POP   D
370 EB97 C1                 POP   B
371 EB98 78                 MOV   A,B
372 EB99 C1                 POP   B
373 EB9A C9                 RET
```



```
374                   *
375                   ************************************
376                   * FIND COLOUR IN COLORG REGISTERS *
377                   ************************************
378                   *
379                   * Entry: C: Requested colour.
380                   * Exit:  CY=1: 'Serialnr' of colour in A.
381                   *        CY=0: Colour not available.
382                   *        BCDEHL preserved.
383                   *
384 EB9B C5           STR164 PUSH  B
385 EB9C E5                 PUSH  H
386 EB9D 219E00             LXI   H,:009E   Addr 1st colour byte graph
387 EBA0 0603               MVI   B,:03     Total 4 colour data
388 EBA2 7E           L2E165 MOV   A,M       Get colour
389 EBA3 E60F               ANI   :0F       Colour nibble only
390 EBA5 B9                 CMP   C         Ident to reqd one ?
391 EBA6 CAB2EB             JZ    :EBB2     Then colour found
392 EBA9 23                 INX   H         Pnts to next one
393 EBAA 05                 DCR   B
394 EBAB F2A2EB             JP    :EBA2     Get next colour
395 EBAE B7                 ORA   A         CY=0
396 EBAF E1           L2E166 POP   H
397 EBB0 C1                 POP   B
398 EBB1 C9                 RET
399
400                   * If colour found:
401
402 EBB2 3E03         L2E167 MVI   A,:03
403 EBB4 90                 SUB   B         Colour'nr' in A
404 EBB5 37                 STC
405 EBB6 C3AFEB             JMP   :EBAF     Quit, CY=1
406                   *
407                   ***********************
408                   * GET MEMORY POINTER *
409                   ***********************
410                   *
411 EBB9 D5           SMEMMK PUSH  D
412 EBBA F5                 PUSH  PSW
413 EBBB 78                 MOV   A,B
414 EBBC 0F                 RRC
415 EBBD 79                 MOV   A,C
416 EBBE 1F                 RAR
417 EBBF 1F                 RAR
418 EBC0 E67E               ANI   :7E
419 EBC2 C604               ADI   :04
420 EBC4 5F                 MOV   E,A
421 EBC5 1600               MVI   D,:00
422 EBC7 E1                 POP   H
423 EBC8 E5                 PUSH  H
424 EBC9 6C                 MOV   L,H       Entry A in L
425 EBCA 2600               MVI   H,:00
426 EBCC 23                 INX   H         +1
427 EBCD 3A9800             LDA   :0098     Get nr bytes/line
428 EBD0 CD46EB             CALL  :EB46     Calc HL=A*HL
429 EBD3 CDF2E6             CALL  :E6F2     HL=HL-DE
430 EBD6 EB                 XCHG            Result in DE
431 EBD7 2A8800             LHLD  :0088     Get addr after end graph
432                                         area
433 EBDA 19                 DAD   D         Add offset
434 EBDB CDC7EA             CALL  :EAC7     Check and move pntr
435 EBDE F1                 POP   PSW
```

ClibPDF - www.fastio.com

```
436 EBDF D1                 POP   D
437 EBE0 C9                 RET
438                 *
439                 **********************
440                 * SET MASK FOR BITS *
441                 **********************
442                 *
443                 * Entry: B and C are data for mask.
444                 * Exit:  B: Result.
445                 *        AFCDEHL preserved.
446                 *
447 EBE1 F5           SMKMSK PUSH  PSW
448 EBE2 C5                  PUSH  B
449 EBE3 AF                  XRA   A          A=0
450 EBE4 37           L2E170 STC               CY=1
451 EBE5 1F                  RAR
452 EBE6 05                  DCR   B
453 EBE7 C2E4EB              JNZ   :EBE4      RAR (B) times
454 EBEA 1F           L2E171 RAR
455 EBEB 0D                  DCR   C
456 EBEC F2EAEB              JP    :EBEA      RAR until C <= 7F
457 EBEF 17                  RAL
458 EBF0 C1                  POP   B
459 EBF1 47                  MOV   B,A        Result in B
460 EBF2 F1                  POP   PSW
461 EBF3 C9                  RET
462                 *
463                 *
464                 *
465 EBF4                     END
```

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

DADCK   EAC1    FBK10   EA17    FBK20   EA19    FBK30   EA2B
FBK40   EA3E    FBK50   EA46    FBK60   EA4D    FBK90   EA53
FILBK   EA0B    FILRT   EAF7    FILST   EA57    FST05   EA78
FST10   EA79    FST15   EA91    FST18   EAA6    FST20   EAA9
FST30   EAAD    FST90   EABC    FST91   EABD    HLDIV   EB60
HLMUL   EB46    L2E152  EB0D    L2E153  EB37    L2E154  EB40
L2E156  EB4C    L2E157  EB56    L2E158  EB5D    L2E160  EB6F
L2E161  EB78    L2E163  EB96    L2E165  EBA2    L2E166  EBAF
L2E167  EBB2    L2E170  EBE4    L2E171  EBEA    PCK10   EAE1
PCK15   EAE2    PCK20   EAE5    PCK30   EAEF    PTRCK   EAC7
SMEMMK  EBB9    SMKMSK  EBE1    STR164  EB9B    TPOSN   EB7A
```



```
002                         ORG   :EBF4
003                 *
004                 *
005                 *
006                 *   ====================
007                 *** SCREEN EDITOR PACKAGE ***
008                 *   ====================
009                 *
010                 *
011                 **********************
012                 * INITIALISE EDITOR *
013                 **********************
014                 *
015                 * Sets up a mode 0 screen, clears 00A8-00AD, 00AF.
016                 * Prints text from 00A2 onwards; cursor at top left.
017                 *
018                 * Exit: HL: Cursor position (top left).
019                 *       AF corrupted, BCDE preserved.
020                 *
021 EBF4 3EFF       EINIT   MVI   A,:FF      Init mode 0
022 EBF6 CDD9E3             CALL  :E3D9      Change to mode 0
023 EBF9 3E0C               MVI   A,:0C
024 EBFB CD02E1             CALL  :E102      Clear screen
025 EBFE 210000             LXI   H,:0000    )
026 EC01 22A900             SHLD  :00A9      )
027 EC04 22AC00             SHLD  :00AC      )
028 EC07 7C                 MOV   A,H        ) Clear edit pointers
029 EC08 32A800             STA   :00A8      )
030 EC0B 32AB00             STA   :00AB      )
031 EC0E 32AF00             STA   :00AF      )
032 EC11 2A7200             LHLD  :0072      Get cursor pos addr
033 EC14 CD6BE3             CALL  :E36B      Delete cursor
034 EC17 CD3AEF             CALL  :EF3A      Print full screen from
035                                          start of edit buffer
036 EC1A CD30E3             CALL  :E330      Put cursor on screen
037 EC1D C9                 RET
038                 *
039                 *************
040                 * OBEY EDIT *
041                 *************
042                 *
043                 * A given character is inserted in the buffer,
044                 * except:  #08  deletes a character
045                 *          #10  moves cursor up
046                 *          #11  moves cursor down
047                 *          #12  moves cursor left
048                 *          #13  moves cursor right
049                 *          #14  moves window up
050                 *          #15  moves window down
051                 *          #16  moves window left
052                 *          #17  moves window right
053                 *
054                 * Entry: A:    Character given.
055                 * Exit:  CY=1: If done.
056                 *        CY=0: Not done.
057                 *        ABCDEHL + rest of F preserved.
058                 *
059 EC1E FE08       EOBEY   CPI   :08
060 EC20 CACCEF             JZ    :EFCC      Delete character
061 EC23 FE10               CPI   :10
062 EC25 DA4BEF             JC    :EF4B      Insert character
063 EC28 CA88ED             JZ    :ED88      Cursor up
```

ClibPDF - www.fastio.com

```
064 EC2B FE12               CPI   :12
065 EC2D DAABED             JC    :EDAB     Cursor down
066 EC30 CAD3ED             JZ    :EDD3     Cursor left
067 EC33 FE14               CPI   :14
068 EC35 DAF6ED             JC    :EDF6     Cursor right
069 EC38 CA4BEC             JZ    :EC4B     Window up
070 EC3B FE16               CPI   :16
071 EC3D DAB3EC             JC    :ECB3     Window down
072 EC40 CA50ED             JZ    :ED50     Window left
073 EC43 FE17               CPI   :17
074 EC45 CAF8EC             JZ    :ECF8     Window right
075 EC48 C34BEF             JMP   :EF4B     Insert character
076                 *
077                 *************
078                 * WINDOW UP *
079                 *************
080                 *
081                 * Moves window up one line.
082                 *
083                 * Exit: All registers preserved.
084                 *       CY=0: Window unchanged.
085                 *       CY=1: Window changed.
086                 *
087 EC4B D5          EWUP    PUSH  D
088 EC4C E5                  PUSH  H
089 EC4D F5                  PUSH  PSW
090 EC4E 2AA900              LHLD  :00A9     Get offset of top of window
091 EC51 7C                  MOV   A,H
092 EC52 B5                  ORA   L
093 EC53 CADAEC              JZ    :ECDA     Window unchanged if
094                                          offset = 0; CY=0
095 EC56 AF                  XRA   A
096 EC57 32AF00              STA   :00AF     Clear cursor pos in buffer
097 EC5A 3AAC00              LDA   :00AC     Get Y-offset cursor in
098                                          in document
099 EC5D 95                  SUB   L         Minus offset top of window
100 EC5E FE17                CPI   :17       Nr of lines in window -1
101 EC60 CC88ED              CZ    :ED88     Cursor up if at
102                                          bottom of window
103 EC63 2A7200              LHLD  :0072     Get cursor pos addr
104 EC66 E5                  PUSH  H
105 EC67 CD6BE3              CALL  :E36B     Delete cursor
106 EC6A CD74EC              CALL  :EC74     Move window, correct pntrs
107 EC6D E1                  POP   H
108 EC6E 117AFF      L2E172  LXI   D,:FF7A   Length one line
109 EC71 C3D5EC              JMP   :ECD5     Put cursor on next line,
110                                          quit with CY=1
111                 *
112                 * SCROLL EDIT DISPLAY DOWN ONE LINE:
113                 *
114                 * Exit: BC preserved. AFDEHL corrupted.
115                 *
116 EC74 C5          L2E173  PUSH  B
117 EC75 018600              LXI   B,:0086   Length one screen line
118 EC78 CD86EC              CALL  :EC86     Move window up in text
119                                          1 line
120 EC7B D5                  PUSH  D
121 EC7C 2AA900              LHLD  :00A9     Get offset of top of window
122 EC7F 2B                  DCX   H         -1
123 EC80 22A900              SHLD  :00A9     And preserve it
124 EC83 C3EFEC              JMP   :ECEF     Print new line of text at
125                                          window bottom
```



```
126                 *
127                 ************************
128                 * MOVE WINDOW IN TEXT *
129                 ************************
130                 *
131                 * Moves window up or left.
132                 *
133                 * Entry: BC: Offset (#86 for 1 line, #02 for
134                 *            1 position).
135                 * Exit:  All registers corrupted.
136                 *
137 EC86 2A8C00      L2E174  LHLD  :008C     Get addr after end char area
138 EC89 54                 MOV   D,H       ) in DE
139 EC8A 5D                 MOV   E,L       )
140 EC8B 09                 DAD   B         Add offset
141 EC8C EB                 XCHG            HL: end char area;
142                                          DE: end char area + offset
143                 * Move full screen:
144
145 EC8D 010600      L2E175  LXI   B,:0006   3 not used locations
146                                          at line end
147 EC90 09                 DAD   B         )
148 EC91 EB                 XCHG            ) Add it to both HL and DE
149 EC92 09                 DAD   B         ) and exchange DE and HL
150 EC93 EB                 XCHG            )
151 EC94 063C               MVI   B,:3C     60 char/line visible
152
153                 * Move one line:
154
155 EC96 23          L2E176  INX   H
156 EC97 23                 INX   H         New destination pntr
157 EC98 13                 INX   D
158 EC99 13                 INX   D         New origin pntr
159 EC9A 1A                 LDAX  D         Get one char
160 EC9B 77                 MOV   M,A       Move it to new screen loc
161 EC9C 05                 DCR   B         char count -1
162 EC9D C296EC             JNZ   :EC96     Next char if not ready
163                 *
164 ECA0 010800             LXI   B,:0008   4 not-useable pos at line
165                                          boundary
166 ECA3 09                 DAD   B         HL pnts to end previous line
167                                          (destination)
168 ECA4 E5                 PUSH  H
169 ECA5 2A8A00             LHLD  :008A     Get addr start char area
170 ECA8 EB                 XCHG            in DE
171 ECA9 09                 DAD   B         Update origin too
172 ECAA CDFBE6             CALL  :E6FB     Compare HL-DE
173 ECAD EB                 XCHG            New origin in DE
174 ECAE E1                 POP   H         Destination in HL
175 ECAF DA8DEC             JC    :EC8D     Evt another line
176 ECB2 C9                 RET
177                 *
178                 ***************
179                 * WINDOW DOWN *
180                 ***************
181                 *
182                 * Moves window down one line.
183                 *
184                 * Exit: ABCDEHL preserved.
185                 *       CY=0: Window unchanged.
186                 *       CY=1: Window changed.
187                 *
```

ClibPDF - www.fastio.com

```
ECB3 D5          EWDN    PUSH  D
ECB4 E5                  PUSH  H
ECB5 F5                  PUSH  PSW
ECB6 2AAC00              LHLD  :00AC      Get Y-offset cursor in
                                          document
ECB9 3AA900              LDA   :00A9      Get offset of top of window
ECBC BD                  CMP   L          Cursor at top of screen ?
ECBD CCABED              CZ    :EDAB      Then cursor down
ECC0 D2DAEC              JNC   :ECDA      Quit with CY=0 if
                                          window unchanged
ECC3 AF                  XRA   A
ECC4 32AF00              STA   :00AF      Clear cursor pos in buffer
ECC7 2A7200              LHLD  :0072      Get cursor pos addr
ECCA E5                  PUSH  H
ECCB CD6BE3              CALL  :E36B      Delete cursor
ECCE CDDFEC              CALL  :ECDF      Scroll edit display up
                                          one line
ECD1 E1                  POP   H
ECD2 118600      L2E177  LXI   D,:0086    Length one screen line
ECD5 19          L2E178  DAD   D          One line further
ECD6 CD30E3      L2E179  CALL  :E330      Put new cursor on screen
ECD9 37                  STC              Exit CY=1 if cursor moved
ECDA E1          L2E180  POP   H
ECDB 7C                  MOV   A,H
ECDC E1                  POP   H
ECDD D1                  POP   D
ECDE C9                  RET
                 *
                 * SCROLL EDIT DISPLAY UP ONE LINE:
                 *
                 * Exit: BC preserved, AFDEHL corrupted.
                 *
ECDF C5          L2E181  PUSH  B
ECE0 CDCBE1              CALL  :E1CB      Scroll window up 1 line
ECE3 E5                  PUSH  H
ECE4 2AA900              LHLD  :00A9      Get offset of top of window
ECE7 23                  INX   H          +1
ECE8 22A900              SHLD  :00A9      Preserve it
ECEB 111700              LXI   D,:0017    Nr of lines for window
ECEE 19                  DAD   D          HL pnts to new line at
                                          window bottom
ECEF CD1CEE      L2E182  CALL  :EE1C      Skip lines
ECF2 D1                  POP   D
ECF3 CDC0EE              CALL  :EEC0      Print new line of text
ECF6 C1                  POP   B
ECF7 C9                  RET
                 *
                 ****************
                 * WINDOW RIGHT *
                 ****************
                 *
                 * Moves window right one position.
                 *
                 * Exit: ABCDEHL preserved.
                 *       CY=0: Window unchanged.
                 *       CY=1: Window changed.
                 *
ECF8 D5          EWRT    PUSH  D
ECF9 E5                  PUSH  H
ECFA F5                  PUSH  PSW
ECFB 3AA800              LDA   :00A8      Get offset of left side
                                          of window
```



```
ECFE FEC4                CPI   :C4        256-60 ?
ED00 CADAEC              JZ    :ECDA      Abort if window at right en
                                          of text
ED03 6F                  MOV   L,A        Offset in L
ED04 3AAB00              LDA   :00AB      Get X-offset cursor in
                                          document
ED07 BD                  CMP   L          Cursor at left side of
                                          window ?
ED08 CCF6ED              CZ    :EDF6      Then move cursor right
ED0B 2A7200              LHLD  :0072      Get cursor pos addr
ED0E E5                  PUSH  H
ED0F CD6BE3              CALL  :E36B      Delete cursor
ED12 CD1BED              CALL  :ED1B      Scroll display left 1 pos
ED15 E1                  POP   H
ED16 23          L2E184  INX   H
ED17 23                  INX   H
ED18 C3D6EC              JMP   :ECD6      Put cursor on screen;
                                          quit with CY=1
                 *
                 * SCROLL EDIT DISPAY LEFT ONE POSITION:
                 *
                 * Exit: BC preserved, AFDEHL corrupted.
                 *
ED1B C5          L2E185  PUSH  B
ED1C 01FEFF              LXI   B,:FFFE    -2
ED1F CDCEE1              CALL  :E1CE      Scroll window
ED22 3AA800              LDA   :00A8      Get offset of left side
                                          of window
ED25 3C                  INR   A          +1
ED26 32A800              STA   :00A8      Preserve it
ED29 C63B                ADI   :3B        +60 (offset right side
                                          of window)
ED2B 1182FF              LXI   D,:FF82    -7E
ED2E 47          L2E186  MOV   B,A        Offset right side of
                                          window in B
ED2F 0E18                MVI   C,:18      Nr of lines in window
ED31 2A8A00              LHLD  :008A      Get pntr to start char area
ED34 19                  DAD   D          Pos 60th char on screen
ED35 EB                  XCHG             in DE
ED36 2AA900              LHLD  :00A9      Get offset of top of window
ED39 CD1CEE              CALL  :EE1C      Skip lines

                 * Put newly visible 60th character on screen:

ED3C CD7BEE      L2E187  CALL  :EE7B      Skip to 8th pos on line
ED3F 12                  STAX  D          Next visible char in window
ED40 CD38EE              CALL  :EE38      Next line
ED43 E5                  PUSH  H
ED44 217AFF              LXI   H,:FF7A    -86 (length one line)
ED47 19                  DAD   D          Pnts to 60th char on
                                          next line
ED48 EB                  XCHG
ED49 E1                  POP   H
ED4A 0D                  DCR   C          Update line count
ED4B C23CED      L2E188  JNZ   :ED3C      Scroll next line if not
                                          ready
ED4E C1                  POP   B
ED4F C9                  RET
                 *
                 ****************
                 * WINDOW LEFT *
                 ****************
```

ClibPDF - www.fastio.com

```
                        *
                        * Moves window left one position.
                        *
                        * Exit: ABCDEHL preserved.
                        *       CY=0: Window unchanged.
                        *       CY=1: Window changed.
                        *
ED50 D5         EWLF    PUSH  D
ED51 E5                 PUSH  H
ED52 F5                 PUSH  PSW
ED53 3AA800             LDA   :00A8     Get offset of left side
                                        of window
ED56 B7                 ORA   A
ED57 CADAEC             JZ    :ECDA     Abort if offset = 0
ED5A 6F                 MOV   L,A       Offset in L
ED5B 3AAB00             LDA   :00AB     Get X-offset of cursor
                                        in document
ED5E 95                 SUB   L
ED5F FE3B               CPI   :3B       Cursor at right side of
                                        window ?
ED61 CCD3ED             CZ    :EDD3     Then cursor left
ED64 2A7200             LHLD  :0072     Get cursor pos addr
ED67 E5                 PUSH  H
ED68 CD6BE3             CALL  :E36B     Delete cursor
ED6B CD74ED             CALL  :ED74     Move window, correct pntrs
ED6E E1                 POP   H
ED6F 2B         L2E189  DCX   H
ED70 2B                 DCX   H         New cursor pos
ED71 C3D6EC             JMP   :ECD6     Put cursor on screen;
                                        quit with CY=1
                        *
                        * SCROLL EDIT DISPLAY RIGHT ONE POSITION:
                        *
                        * Exit: BC preserved; AFDEHL corrupted.
                        *
ED74 C5         L2E190  PUSH  B
ED75 010200             LXI   B,:0002
ED78 CD86EC             CALL  :EC86     Move window in text
ED7B 3AA800             LDA   :00A8     Get offset of left side
                                        of window
ED7E 3D                 DCR   A         -1
ED7F 32A800             STA   :00A8     Preserve it
ED82 11F8FF             LXI   D,:FFF8   Gives in ED2E 1st pos
                                        on 1st screen line
ED85 C32EED             JMP   :ED2E     Scroll display left
                        *
                        **************
                        * CURSOR UP *
                        **************
                        *
                        * Moves cursor up one position.
                        *
                        * Exit: ABCDEHL preserved.
                        *       CY=0: Cursor not moved.
                        *       CY=1: Cursor moved.
                        *
ED88 D5         ECUP    PUSH  D
ED89 E5                 PUSH  H
ED8A F5                 PUSH  PSW
ED8B 2AAC00             LHLD  :00AC     Get Y-offset cursor in
                                        document
ED8E 7C                 MOV   A,H
```



```
ED8F B5                 ORA   L
ED90 CADAEC             JZ    :ECDA     Quit if cursor at top
                                        left; CY=0
ED93 AF                 XRA   A
ED94 32AF00             STA   :00AF     Clear cursor pos in buffer
ED97 3AA900             LDA   :00A9     Get offset of top of window
ED9A BD                 CMP   L         Cursor at top of window ?
ED9B CC4BEC             CZ    :EC4B     Then window up
ED9E 2B                 DCX   H
ED9F 22AC00             SHLD  :00AC     Store Y-offset cursor in
                                        document
EDA2 2A7200             LHLD  :0072     Get cursor pos addr
EDA5 CD6BE3             CALL  :E36B     Delete cursor
EDA8 C3D2EC             JMP   :ECD2     Put cursor on new pos;
                                        quit with CY=1
                        *
                        **************
                        * CURSOR DOWN *
                        **************
                        *
                        * Moves cursor down one position.
                        *
                        * Exit: ABCDEHL preserved.
                        *       CY=0: Cursor not moved.
                        *       CY=1: Cursor moved.
                        *
EDAB D5         ECDN    PUSH  D
EDAC E5                 PUSH  H
EDAD F5                 PUSH  PSW
EDAE 2AAC00             LHLD  :00AC     Get Y-offset cursor in
                                        in document
EDB1 23                 INX   H         +1
EDB2 E5                 PUSH  H         Preserve it
EDB3 CD1CEE             CALL  :EE1C     Skip lines
EDB6 E1                 POP   H
EDB7 D2DAEC             JNC   :ECDA     No move if cursor at last
                                        line of text; CY=0
EDBA AF                 XRA   A
EDBB 32AF00             STA   :00AF     Clear cursor pos in buffer
EDBE 3AA900             LDA   :00A9     Get offset of top of window
EDC1 C618               ADI   :18       24 lines in window
EDC3 BD                 CMP   L         Cursor at window bottom ?
EDC4 CCB3EC             CZ    :ECB3     Then window down
EDC7 22AC00             SHLD  :00AC     Store Y-offset cursor in
                                        document
EDCA 2A7200             LHLD  :0072     Get cursor pos addr
EDCD CD6BE3             CALL  :E36B     Delete cursor
EDD0 C36EEC             JMP   :EC6E     Put cursor on new pos;
                                        quit with CY=1
                        *
                        **************
                        * CURSOR LEFT *
                        **************
                        *
                        * Moves cursor left one position.
                        *
                        * Exit: ABCDEHL preserved.
                        *       CY=0: Cursor not moved.
                        *       CY=1: Cursor moved.
                        *
EDD3 D5         ECLF    PUSH  D
EDD4 E5                 PUSH  H
```

ClibPDF - www.fastio.com

```
436 EDD5 F5                 PUSH  PSW
437 EDD6 3AA800             LDA   :00A8     Get offset of left side
438                                          of window
439 EDD9 6F                 MOV   L,A        in L
440 EDDA 3AAB00             LDA   :00AB     Get X-offset cursor in
441                                          document
442 EDDD B7                 ORA   A
443 EDDE CADAEC             JZ    :ECDA     Abort if X-offset = 0
444 EDE1 BD                 CMP   L         Cursor at left side of
445                                          window ?
446 EDE2 CC50ED             CZ    :ED50     Then window left
447 EDE5 3D                 DCR   A         X-offset -1
448 EDE6 32AB00             STA   :00AB     Preserve it
449 EDE9 AF                 XRA   A
450 EDEA 32AF00             STA   :00AF     Clear cursor pos in buffer
451 EDED 2A7200             LHLD  :0072     Get cursor pos addr
452 EDF0 CD6BE3             CALL  :E36B     Delete cursor
453 EDF3 C316ED             JMP   :ED16     Put cursor on new pos;
454                                          quit with CY=1
455                 *
456                 *
457                 *
458 EDF6                    END
```

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

ECDN   EDAB   ECLF   EDD3   ECUP   ED88   EINIT  EBF4
EOBEY  EC1E   EWDN   ECB3   EWLF   ED50   EWRT   ECF8
EWUP   EC4B   L2E172 EC6E   L2E173 EC74   L2E174 EC86
L2E175 EC8D   L2E176 EC96   L2E177 ECD2   L2E178 ECD5
L2E179 ECD6   L2E180 ECDA   L2E181 ECDF   L2E182 ECEF
L2E184 ED16   L2E185 ED1B   L2E186 ED2E   L2E187 ED3C
L2E188 ED4B   L2E189 ED6F   L2E190 ED74
```



```
002                         ORG   :EDF6
003                 *
004                 *
005                 *
006                 ****************
007                 * CURSOR RIGHT *
008                 ****************
009                 *
010                 * Moves cursor right one position.
011                 *
012                 * Exit: ABCDEHL preserved.
013                 *       CY=0: Cursor not moved.
014                 *       CY=1: Cursor moved.
015                 *
016 EDF6 D5         ECRT    PUSH  D
017 EDF7 E5                 PUSH  H
018 EDF8 F5                 PUSH  PSW
019 EDF9 3AA800             LDA   :00A8     Get offset of left side
020                                          of window
021 EDFC C63B               ADI   :3B       Calc end of window
022 EDFE 6F                 MOV   L,A       Result in L
023 EDFF 3AAB00             LDA   :00AB     Get X-offset of cursor
024                                          in document
025 EE02 FEFF               CPI   :FF       Max nr char/line reached ?
026 EE04 CADAEC             JZ    :ECDA     Then quit with CY=0
027 EE07 BD                 CMP   L         End of window reached ?
028 EE08 CCF8EC             CZ    :ECF8     Then window right
029 EE0B 3C                 INR   A         Incr cursor pos on line
030 EE0C 32AB00             STA   :00AB     Store X-offset of cursor
031                                          in document
032 EE0F AF                 XRA   A
033 EE10 32AF00             STA   :00AF     Clear cursor pos in buffer
034 EE13 2A7200             LHLD  :0072     Get cursor pos addr
035 EE16 CD6BE3             CALL  :E36B     Delete old cursor
036 EE19 C36FED             JMP   :ED6F     Put new cursor on screen;
037                                          quit with CY=1
038                 *
039                 ****************************
040                 * SKIP LINES IN EDIT BUFFER *
041                 ****************************
042                 *
043                 * Entry: HL: Number of lines to be skipped.
044                 * Exit:  HL: Points to 1st not skipped line.
045                 *        CY=1: O.K.
046                 *        CY=0: End of text reached before count
047                 *              is zero.
048                 *        ABCDE preserved.
049                 *
050 EE1C D5         L2E191  PUSH  D
051 EE1D F5                 PUSH  PSW
052 EE1E EB                 XCHG            Count in DE
053 EE1F 2AA200             LHLD  :00A2     Get startaddr editbuf
054 EE22 7A         L2E192  MOV   A,D       )
055 EE23 B3                 ORA   E         ) Abort with CY=1 if ready
056 EE24 CA33EE             JZ    :EE33     )
057 EE27 CD38EE             CALL  :EE38     Next line
058 EE2A 00                 NOP
059 EE2B B7                 ORA   A
060 EE2C CA34EE             JZ    :EE34     Abort with CY=0 if end of
061                                          buffer reached before DE=0
062 EE2F 1B                 DCX   D         Count -1
063 EE30 C322EE             JMP   :EE22     Skip next line
```

ClibPDF - www.fastio.com

```
064 EE33 37          L2E193  STC
065 EE34 D1          L2E194  POP   D
066 EE35 7A                  MOV   A,D
067 EE36 D1                  POP   D
068 EE37 C9                  RET
069                  *
070                  **************
071                  * NEXT LINE *
072                  **************
073                  *
074                  * Entry: HL: points to text, each line ended by
075                  *             a car.ret, last line followed by 0.
076                  * Exit:  HL: Points to next line.
077                  *        A:  0 if end of text reached, else CR.
078                  *        BCDE preserved.
079                  *
080 EE38 7E          L2E195  MOV   A,M       Get char from text
081 EE39 B7                  ORA   A
082 EE3A CA43EE              JZ    :EE43     Ready if end of text reached
083 EE3D 23                  INX   H
084 EE3E FE0D                CPI   :0D       Car.ret ?
085 EE40 C238EE              JNZ   :EE38     Loop till end of line found
086 EE43 C9          L2E196  RET
087                  *
088                  ****************************************
089                  * FIND CURRENT POSITION IN EDIT BUFFER *
090                  ****************************************
091                  *
092                  * Exit: CY=1: HL: (00AF/AE) = cursor position in
093                  *                 buffer.
094                  *       CY=0: HL: Points to cursor line in the
095                  *                 edit buffer.
096                  *       ABCDE preserved.
097                  *
098 EE44 2AAE00      L2E197  LHLD  :00AE     Get pntr to cursor pos
099                                          in buffer
100 EE47 25                  DCR   H
101 EE48 24                  INR   H
102 EE49 37                  STC
103 EE4A C0                  RNZ           Quit (CY=1) if (AF) <> 0
104 EE4B D5                  PUSH  D
105 EE4C C5                  PUSH  B
106 EE4D F5                  PUSH  PSW
107 EE4E 2AAC00              LHLD  :00AC     Get Y-offset of cursor
108                                          in document (no lines to
109                                          be skipped)
110 EE51 54                  MOV   D,H       ) in DE
111 EE52 5D                  MOV   E,L       )
112 EE53 CD1CEE              CALL  :EE1C     Skip lines
113 EE56 22B200              SHLD  :00B2     Store pntr to start cursor
114                                          line in buffer
115 EE59 E5                  PUSH  H         and preserve it
116 EE5A 2AA900              LHLD  :00A9     Get offset of top of window
117 EE5D CDF2E6              CALL  :E6F2     Calc difference
118 EE60 3E86                MVI   A,:86     Length one line
119 EE62 CD46EB              CALL  :EB46     Calc total length
120 EE65 EB                  XCHG            Result in DE
121 EE66 2A8A00              LHLD  :008A     Get startaddr char area
122 EE69 19                  DAD   D         Add offset
123 EE6A 22B000              SHLD  :00B0     Preserve pntr to start
124                                          cursor line on screen
125 EE6D E1                  POP   H         Get pntr to start cursor
```



```
126                                          line in buffer
127 EE6E 3AAB00              LDA   :00AB     Get X-offset of cursor
128                                          in document
129 EE71 47                  MOV   B,A       in B
130 EE72 CD7BEE              CALL  :EE7B     Skip to Bth pos on line
131                                          exit: HL pnts to cursor pos
132 EE75 C1                  POP   B
133 EE76 78                  MOV   A,B
134 EE77 C1                  POP   B
135 EE78 D1                  POP   D
136 EE79 3F                  CMC             Abort with CY=0
137 EE7A C9                  RET
138                  *
139                  ************************************
140                  * SKIP TO Bth POSITION ON TEXT LINE *
141                  ************************************
142                  *
143                  * Looks through a textline until Bth position is
144                  * found. On exit, character on this position is
145                  * in A.
146                  *
147                  * Entry: HL: Start text line.
148                  *        B:  Number of position in line.
149                  * Exit:  CY=0: Position found:
150                  *              A:  Character on that position.
151                  *              HL: Points to Bth position.
152                  *              BCDE preserved.
153                  *        CY=1: Bth position is beyond car.ret,
154                  *              tab or end of text.
155                  *              HL: Points to CR, tab or 0.
156                  *              A:  Space.
157                  *
158 EE7B C5          L2E198  PUSH  B
159 EE7C D5                  PUSH  D
160 EE7D 0E00                MVI   C,:00     Init count
161 EE7F C3FDD1      L2E199  JMP   :D1FD     Evaluate character
162 EE82 CAA3EE      L2E200  JZ    :EEA3     Abort if Bth pos reached
163 EE85 B7                  ORA   A         Set flags on char
164 EE86 CAA0EE              JZ    :EEA0     Jump if end of text reached
165 EE89 FE0D                CPI   :0D
166 EE8B CAA0EE              JZ    :EEA0     Jump if char is car.ret
167 EE8E FE09                CPI   :09
168 EE90 CA98EE              JZ    :EE98     Jump if char is tab
169 EE93 0C                  INR   C         Incr count
170 EE94 23          L2E201  INX   H         Pnts to next pos on line
171 EE95 C37FEE              JMP   :EE7F     Loop untill ready
172
173                  * If character is tab:
174
175 EE98 CDA6EE      L2E202  CALL  :EEA6     Tabulate
176 EE9B 78                  MOV   A,B       Reqd pos in A
177 EE9C B9                  CMP   C         Compare with tab stops
178 EE9D D294EE              JNC   :EE94     Continu if not past tab
179
180                  * If end of line or end of text reached or if
181                  * past tab:
182
183 EEA0 3E20        L2E203  MVI   A,:20     Set char is space
184 EEA2 37                  STC             Abort with CY=1
185 EEA3 D1          L2E204  POP   D
186 EEA4 C1                  POP   B
187 EEA5 C9                  RET
```

ClibPDF - www.fastio.com

```
                        *
                        ************
                        * TABULATE *
                        ************
                        *
                        * Routine goes through tab-table for 1st
                        * tab-stop > C.
                        *
                        * Entry: 00B4/B5:  Pointer to tab-table.
                        *        C:        Line position.
                        * Exit:  If found: Tab in C.
                        *        Else:     C = C + 1.
                        *        AFBDEHL preserved.
                        *
EEA6 F5         L2E205  PUSH  PSW
EEA7 C5                 PUSH  B
EEA8 E5                 PUSH  H
EEA9 2AB400             LHLD  :00B4     Get addr tab table
EEAC 41         L2E206  MOV   B,C       Line pos in B
EEAD 04                 INR   B         +1
EEAE 7E                 MOV   A,M       Get tab from table
EEAF B7                 ORA   A
EEB0 CABAEE             JZ    :EEBA     If end tab table reached
EEB3 23                 INX   H         Pnts to next tab
EEB4 47                 MOV   B,A       Tab in B
EEB5 79                 MOV   A,C       Line pos in A
EEB6 B8                 CMP   B
EEB7 D2ACEE             JNC   :EEAC     Get next tab if line
                                        pos > tab stop
EEBA 48         L2E207  MOV   C,B       Replace line pos by tab stop
                                        or by line pos +1 if no tab
                                        found
EEBB E1                 POP   H
EEBC F1                 POP   PSW
EEBD 47                 MOV   B,A
EEBE F1                 POP   PSW
EEBF C9                 RET
                        *
                        ********************************
                        * PRINT LINE OF TEXT IN WINDOW *
                        ********************************
                        *
                        * Entry: HL:   Address of textline in buffer.
                        *        DE:   Line control byte of screen line
                        *              to print text on.
                        *        00A8: Number of non-printing positions.
                        *
                        * During execution loop:
                        *        B:    Nr of non-printing pos left +1.
                        *        C:    Nr of screen line pos left.
                        *        D:    Current text line position.
                        *        E:    Count of blanks inserted.
                        *        HL:   Points to text.
                        *        TOS:  Points to screen.
                        *
EEC0 C5         L2E208  PUSH  B
EEC1 EB                 XCHG
EEC2 01F8FF             LXI   B,:FFF8
EEC5 09                 DAD   B         1st printable pos on screen
EEC6 EB                 XCHG            in DE
EEC7 3AA800             LDA   :00A8     Get offset of left side
                                        of window
```



```
EECA 3C                 INR   A
EECB 47                 MOV   B,A       B is offset +1
EECC 0E3C               MVI   C,:3C     60 visible characters
EECE D5                 PUSH  D         Preserve start screen line
EECF 1600               MVI   D,:00     ) Init current text pos
EED1 5A                 MOV   E,D       ) and blanks count

                        * Loop:

EED2 3E20       L2E209  MVI   A,:20     Space
EED4 1D                 DCR   E
EED5 1C                 INR   E
EED6 C2ECEE             JNZ   :EEEC     E<>0: Update screen pos pntr
EED9 7E                 MOV   A,M       Else: Get char from buffer
EEDA B7                 ORA   A         Set flags on char
EEDB C3D1D1             JMP   :D1D1     Test character
EEDE FE0D       L2E210  CPI   :0D
EEE0 CAECEE             JZ    :EEEC     Skip tab-handling if CR
EEE3 1E00               MVI   E,:00     Reset blanks count
EEE5 23                 INX   H         Pnts to next char in text
EEE6 FE09               CPI   :09       Tab ?
EEE8 CA08EF             JZ    :EF08     Then tab-handling
EEEB 1C         L2E211  INR   E
EEEC 1D         L2E212  DCR   E         Update blanks count
EEED 14                 INR   D         Update line position
EEEE 05                 DCR   B         and nr of pos left in window
EEEF C2D2EE             JNZ   :EED2     No printing, cont with
                                        next char
EEF2 04                 INR   B
EEF3 E3                 XTHL            Screen pos in HL
EEF4 77                 MOV   M,A       Display char on screen
EEF5 2B                 DCX   H
EEF6 2B                 DCX   H         Next screen pos
EEF7 E3                 XTHL            Preserve it
EEF8 0D                 DCR   C         End of screen line reached ?
EEF9 C2D2EE             JNZ   :EED2     Next char if not
EEFC CD38EE             CALL  :EE38     Else: next line
EEFF E3                 XTHL            HL is screen pos
EF00 11FAFF             LXI   D,:FFFA
EF03 19                 DAD   D         HL is 1st useable pos
                                        on next line
EF04 EB                 XCHG            into DE
EF05 E1                 POP   H
EF06 C1                 POP   B
EF07 C9                 RET

                        * Tab-handling:

EF08 C5         L2E213  PUSH  B
EF09 4A                 MOV   C,D       C is pos on current line
EF0A CDA6EE             CALL  :EEA6     Tabulate
EF0D 79                 MOV   A,C       Next tab stop in A
EF0E 92                 SUB   D         ) Calc blanks to be inserted
EF0F 3D                 DCR   A         )
EF10 5F                 MOV   E,A       Update blanks count
EF11 C1                 POP   B
EF12 3E09               MVI   A,:09     Restore tab char in A
EF14 C3EBEE             JMP   :EEEB
                        *
                        ****************************
                        * PRINT A COMPLETE WINDOW *
                        ****************************
```

ClibPDF - www.fastio.com

```
                        *
                        * Text is printed in window from (DE) to bottom
                        * text screen if text ends. A ASCII 0 (vertical
                        * bar) is printed at the beginning of all
                        * following lines.
                        *
                        * Entry: HL: Points to text.
                        *        DE: Points to line control byte screen line
                        * Exit:  HL: Points to next line to print.
                        *        DE: Bottom text screen.
                        *        AF corrupted, BC preserved.
                        *
EF17 E5         L2E214  PUSH  H
EF18 2A8C00             LHLD  :008C     Get bottom text screen
EF1B CDFBE6             CALL  :E6FB     Compare HL-DE
EF1E E1                 POP   H
EF1F CA28EF             JZ    :EF28     Quit if window full
EF22 CDC0EE             CALL  :EEC0     Print a textline
EF25 C317EF             JMP   :EF17     Loop until ready
EF28 C9         L2E215  RET
                        *
                        ****************************************
                        * PRINT A TEXT LINE POINTED BY (00B2) *
                        * ON SCREEN POSITION (00B0) IN WINDOW *
                        ****************************************
                        *
                        * Exit: All registers preserved.
                        *
EF29 F5         L2E216  PUSH  PSW
EF2A C5                 PUSH  B
EF2B D5                 PUSH  D
EF2C E5                 PUSH  H
EF2D 2AB000             LHLD  :00B0     Get pntr to start cursor
                                        line on screen
EF30 EB                 XCHG            in DE
EF31 2AB200             LHLD  :00B2     Get pntr to start cursor
                                        line in buffer
EF34 CDC0EE             CALL  :EEC0     Print a text line
EF37 C338E1             JMP   :E138     Popall, ret
                        *
                        **********************
                        * PRINT FULL SCREEN *
                        **********************
                        *
                        * Prints from Nth line in full screen window.
                        * N is the offset of the top of the window from
                        * the start of the edit buffer.
                        *
                        * Exit: All registers preserved.
                        *
EF3A F5         L2E217  PUSH  PSW
EF3B C5                 PUSH  B
EF3C D5                 PUSH  D
EF3D E5                 PUSH  H
EF3E 2A8A00             LHLD  :008A     Get startaddr char area
EF41 EB                 XCHG            in DE
EF42 2AA900             LHLD  :00A9     Get offset of top of window
EF45 CD1CEE             CALL  :EE1C     Skip lines
EF48 C385CE             JMP   :CE85     Print complete window
                        *
                        *
                        *
```



```
                        ******************************
                        * INSERT CHARACTER IN BUFFER *
                        ******************************
                        *
                        * Entry: Character in A.
                        * Exit:  ABCDEHL preserved.
                        *        CY=0: Buffer full, no action.
                        *        CY=1: Character inserted.
                        *
EF4B C5         EINCH   PUSH  B
EF4C F5                 PUSH  PSW
EF4D D5                 PUSH  D
EF4E E5                 PUSH  H
EF4F 47                 MOV   B,A       Char in B
EF50 2AA600             LHLD  :00A6     Get end available space
EF53 EB                 XCHG            in DE
EF54 2AA400             LHLD  :00A4     Get input pointer
EF57 CDFBE6             CALL  :E6FB     Compare HL-DE
EF5A EB                 XCHG            Input pntr in E
EF5B 78                 MOV   A,B       Char in A
EF5C DC44EE             CC    :EE44     Space available: Find
                                        current pos in buffer
EF5F D296EF             JNC   :EF96     Buffer full: quit, char in A
EF62 E5                 PUSH  H         Preserve end available space
EF63 2A7200             LHLD  :0072     Get cursor pos addr
EF66 E3                 XTHL            Put it on stack
EF67 E5                 PUSH  H
EF68 CD6BE3             CALL  :E36B     Delete cursor
EF6B EB                 XCHG            Old input pntr in HL
EF6C 00                 NOP
EF6D 23                 INX   H         +1
EF6E 22A400             SHLD  :00A4     Update input pointer
EF71 2B                 DCX   H
EF72 44                 MOV   B,H       ) Old input pntr in BC
EF73 4D                 MOV   C,L       )
EF74 2B                 DCX   H
EF75 EB                 XCHG            Old input pntr -1 in DE
EF76 E1                 POP   H         Get end available space
EF77 2B                 DCX   H         -1
EF78 CDC2E6             CALL  :E6C2     Move screen area 1 pos
EF7B 23                 INX   H
EF7C 77                 MOV   M,A       Insert char in buffer
EF7D 23                 INX   H
EF7E 22AE00             SHLD  :00AE     Preserve pntr to cursor
                                        in buffer
EF81 E1                 POP   H         Get end available space
EF82 FE0D               CPI   :0D
EF84 CA9CEF             JZ    :EF9C     Jump if char is car.ret
EF87 CD29EF             CALL  :EF29     Reprint text line in window
EF8A CD30E3             CALL  :E330     Put cursor on screen
EF8D FE09               CPI   :09
EF8F CAB9EF             JZ    :EFB9     Jump if char is tab
EF92 CDF6ED             CALL  :EDF6     Move cursor right 1 pos
EF95 37         L2E218  STC
EF96 E1         L2E219  POP   H
EF97 D1                 POP   D
EF98 C1                 POP   B
EF99 78                 MOV   A,B
EF9A C1                 POP   B
EF9B C9                 RET

                        * If car.ret:
```

ClibPDF - www.fastio.com

```
436
437 EF9C AF          L2E220  XRA   A
438 EF9D 32A800             STA   :00A8     )
439 EFA0 32AB00             STA   :00AB     ) Reset pointers
440 EFA3 32AF00             STA   :00AF     )
441 EFA6 CD3AEF             CALL  :EF3A     Reprint full screen
442 EFA9 2AB000             LHLD  :00B0     Get pntr to start cursor
443                                         line on screen
444 EFAC 11F8FF             LXI   D,:FFF8
445 EFAF 19                 DAD   D         New cursor addr
446 EFB0 CD30E3             CALL  :E330     Put cursor on screen
447 EFB3 CDABED             CALL  :EDAB     Move cursor down
448 EFB6 C395EF             JMP   :EF95     Quit with CY=1
449
450                 * If tab:
451
452 EFB9 CDF6ED     L2E230  CALL  :EDF6     Move cursor right
453 EFBC 3AAB00             LDA   :00AB     Get X-offset of cursor
454                                         in document
455 EFBF 47                 MOV   B,A       in B
456 EFC0 2AB200             LHLD  :00B2     Get pntr to start cursor
457                                         line in buffer
458 EFC3 CD7BEE             CALL  :EE7B     Skip to Bth pos on line
459 EFC6 DAB9EF             JC    :EFB9     Again
460 EFC9 C395EF             JMP   :EF95     Abort with CY=1
461                 *
462                 ******************************
463                 * DELETE CHARACTER IN BUFFER *
464                 ******************************
465                 *
466                 * Deletes character at cursor position and moves
467                 * the text above this position. No action if past
468                 * car.ret, tab or at or beyond end of text.
469                 *
470                 * Exit: ABCDEHL preserved.
471                 *       CY=1: O.K.
472                 *       CY=0: If past car.ret, tab or beyond 0.
473                 *
474 EFCC C5          EDLCH   PUSH  B
475 EFCD F5                 PUSH  PSW
476 EFCE D5                 PUSH  D
477 EFCF E5                 PUSH  H
478 EFD0 CD44EE             CALL  :EE44     Find current pos in buffer
479 EFD3 D296EF             JNC   :EF96     Quit if past CR, tab
480                                         or 0 with CY=0
481 EFD6 7E                 MOV   A,M       Get char
482 EFD7 B7                 ORA   A
483 EFD8 CA96EF             JZ    :EF96     If at end of text:
484                                         quit with CY=0
485 EFDB 00                 NOP
486 EFDC 00                 NOP
487 EFDD 00                 NOP
488 EFDE E5                 PUSH  H
489 EFDF 2A7200             LHLD  :0072     Get cursor pos addr
490 EFE2 E3                 XTHL            preserve it on stack
491 EFE3 CD6BE3             CALL  :E36B     Delete cursor
492 EFE6 EB                 XCHG            Original HL in DE
493 EFE7 2AA400             LHLD  :00A4     Get input pointer
494 EFEA 2B                 DCX   H         Decrement it
495 EFEB 22A400             SHLD  :00A4     And store update input pntr
496 EFEE 44                 MOV   B,H       ) Move updated pointer
497 EFEF 4D                 MOV   C,L       ) into BC
```



```
498 EFF0 0B                 DCX   B         Decrement it once again
499 EFF1 EB                 XCHG            restore original HL
500 EFF2 CDC2E6             CALL  :E6C2     Move screen area above
501                                         downwards
502 EFF5 FE0D               CPI   :0D       Deleted car.ret ?
503 EFF7 C429EF             CNZ   :EF29     If not: reprint text line
504 EFFA CC3AEF             CZ    :EF3A     Else: reprint full screen
505 EFFD C3F2CE             JMP   :CEF2     Put cursor on screen, abort
506                                         with CY=1
507                     *
508                     *
509                     *
510 F000                    END

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

ECRT   EDF6   EDLCH  EFCC   EINCH  EF4B   L2E191 EE1C
L2E192 EE22   L2E193 EE33   L2E194 EE34   L2E195 EE38
L2E196 EE43   L2E197 EE44   L2E198 EE7B   L2E199 EE7F
L2E200 EE82   L2E201 EE94   L2E202 EE98   L2E203 EEA0
L2E204 EEA3   L2E205 EEA6   L2E206 EEAC   L2E207 EEBA
L2E208 EEC0   L2E209 EED2   L2E210 EEDE   L2E211 EEEB
L2E212 EEEC   L2E213 EF08   L2E214 EF17   L2E215 EF28
L2E216 EF29   L2E217 EF3A   L2E218 EF95   L2E219 EF96
L2E220 EF9C   L2E230 EFB9
```

ClibPDF - www.fastio.com