```
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
                                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
```

```
126                                             frame start
127 E65E 09             DAD   B                 Calc new cursor pos addr
128 E65F CD30E3         CALL  :E330             Keep cursor on same pos on
129                                             line
130 E662 2A7800         LHLD  :0078             Get old start line pointer
131 E665 CDF2E6         CALL  :E6F2             HL=HL-DE
132 E668 09             DAD   B                 Calc new cursor pos
133 E669 CD98E6         CALL  :E698             Store addr line mode byte
134                                             current line and last addr
135                                             on that line
136 E66C E1             POP   H                 Get end preserved text area
137 E66D CDF2E6         CALL  :E6F2             HL=HL-DE
138 E670 09             DAD   B
139 E671 D1             POP   D
140 E672 C1             POP   B
141 E673 F1             POP   PSW
142 E674 C9             RET
143
144                 * Scroll frame 1 line if cursor outside frame:
145
146 E675 E1         SMV20   POP   H
147 E676 2A7800             LHLD  :0078         Get startaddr cursor line
148 E679 117AFF             LXI   D,:FF7A
149 E67C 19                 DAD   D             HL: start line after cursor
150 E67D E5                 PUSH  H
151 E67E 111802             LXI   D,:0218
152 E681 19                 DAD   D             Subtract 4 lines and get
153 E682 EB                 XCHG                line mode byte in DE
154 E683 E1                 POP   H             Get end reqd area
155 E684 C34FE6             JMP   :E64F
156                 *
157                 ************************************
158                 * PLACE CURSOR AT BEGIN OF LINE *
159                 ************************************
160                 *
161                 * Sets the cursor at the beginning of a line.
162                 * Several pointers are updated.
163                 *
164                 * SSETC: Given a pointer to the start of line,
165                 *        sets start and end line variables and
166                 *        places the cursor at the beginning of
167                 *        the line.
168                 * SSETL: Sets only start and end line positions.
169                 *
170                 * Entry: HL: Address line mode byte current line.
171                 * Exit:  ABCDEHL preserved.
172                 *
173 E687 F5         SSETC   PUSH  PSW
174 E688 D5                 PUSH  D
175 E689 E5                 PUSH  H
176 E68A 11F8FF             LXI   D,:FFF8
177 E68D 19                 DAD   D             Get addr 1st data byte
178                                             on current line
179 E68E CD30E3             CALL  :E330         Put cursor on screen
180 E691 AF                 XRA   A
181 E692 327B00             STA   :007B         No extended lines
182 E695 E1                 POP   H
183 E696 D1                 POP   D
184 E697 F1                 POP   PSW
185 E698 F5         SSETL   PUSH  PSW
186 E699 227800             SHLD  :0078         Store addr line mode byte
187                                             current line
```

```
188 E69C 3E80              MVI   A,:80     ) Calc lobyte last addr
189 E69E 85                ADD   L         ) on this line
190 E69F 327A00            STA   :007A     Store it in LNEND
191 E6A2 F1                POP   PSW
192 E6A3 C9                RET
193                  *
194                  ************************
195                  * SET GRAPHICS COLOURS *
196                  ************************
197                  *
198                  * Sets the colours available in a 4 colour mode
199                  * and the initial background in a 16 colour mode.
200                  *
201                  * Entry: HL: Points to colours vector.
202                  * Exit:  All registers preserved.
203                  *
204 E6A4 F5           SCOLG  PUSH  PSW
205 E6A5 C5                  PUSH  B
206 E6A6 D5                  PUSH  D
207 E6A7 E5                  PUSH  H
208 E6A8 119E00            LXI   D,:009E   Addr 1st COLORG byte
209 E6AB CD54E2            CALL  :E254     Set COLORG parameters
210 E6AE 3A9D00            LDA   :009D     Get current screen mode
211 E6B1 B7                ORA   A         Check mode type
212 E6B2 2A8200            LHLD  :0082     Get addr after header
213 E6B5 F467E2            CP    :E267     If not mode 0: Load COLORG
214                                          parameters in header
215 E6B8 1F                RAR             Check if char mode
216 E6B9 2A8E00            LHLD  :008E     Get addr after trailer
217 E6BC D467E2            CNC   :E267     If all graphics mode:
218                                          Load colours in trailer
219 E6BF C338E1      SGC10 JMP   :E138     Popall, ret
220                  *
221                  **********************
222                  * MOVE SCREEN AREA *
223                  **********************
224                  *
225                  * Moves a block of screen data from any position
226                  * to any other.
227                  *
228                  * Entry: BC: Points to highaddress target area.
229                  *        DE: Points to highaddress source area.
230                  *        HL: Points to lowaddress -1 source area.
231                  * Exit:  All registers preserved.
232                  *
233 E6C2 F5           MOVES  PUSH  PSW
234 E6C3 C5                  PUSH  B
235 E6C4 D5                  PUSH  D
236 E6C5 E5                  PUSH  H
237 E6C6 CDF2E6            CALL  :E6F2     Calc length of block
238                                          (neg.value)
239 E6C9 7B                MOV   A,E
240 E6CA 91                SUB   C
241 E6CB 7A                MOV   A,D
242 E6CC 98                SBB   B
243 E6CD DAE2E6            JC    :E6E2     Jump if move up
244
245                  * Move down:
246
247 E6D0 54                MOV   D,H       ) Length in DE
248 E6D1 5D                MOV   E,L       )
249 E6D2 09                DAD   B         HL = lowest targetaddr -1
```

```
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
```

```
                        *
                        ***************************
                        * ADD OFFSET TO ADDRESS *
                        ***************************
                        *
                        * Sets HL = HL + A.
                        *
                        * Entry: HL: baseaddress.
                        *        A : offset.
                        * Exit:  HL = HL + A.
                        *        BCDE preserved.
                        *
E701 85                 DADA    ADD   L          Add lobyte addr to offset
E702 6F                         MOV   L,A        and store it in L
E703 D0                         RNC
E704 24                         INR   H          Incr hibyte if overflow
E705 C9                         RET
                        *
                        ************************************
                        * TWO COMPLEMENT OF 16-BITS DATA *
                        ************************************
                        *
                        * Sets HL = - HL.
                        *
                        * Entry: Data to be complemented in HL.
                        * Exit:  HL contains two-complement.
                        *        AFBCDE preserved.
                        *
E706 F5                 CMPHL   PUSH  PSW
E707 7D                         MOV   A,L
E708 2F                         CMA              Compl. L
E709 6F                         MOV   L,A        and store it
E70A 7C                         MOV   A,H
E70B 2F                         CMA              Compl. H
E70C 67                         MOV   H,A        and store it
E70D 23                         INX   H          Add 1
E70E F1                         POP   PSW
E70F C9                         RET
                        *
                        ******************************
                        * DRAW A DOT ON THE SCREEN *
                        ******************************
                        *
                        * Draws a single blob of a colour anywhere
                        * on the screen.
                        *
                        * Entry: C,HL: Y,X coordinate of the dot.
                        *        A:    Colour of the dot.
                        * Exit:  CY=0: O.K.
                        *        CY=1: Error code in A.
                        *        ABCDEHL preserved.
                        *
E710 B7                 SDOT    ORA   A
E711 F5                         PUSH  PSW
E712 C5                         PUSH  B
E713 D5                         PUSH  D
E714 E5                         PUSH  H
E715 41                         MOV   B,C        Y-coord in B
E716 54                         MOV   D,H        ) X-coord in DE
E717 5D                         MOV   E,L        )
E718 C31DE8                     JMP   :E81D      Into 'SFILL'
                        *
```

```
                        ******************************
                        * 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:

E72C EB                         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
```

```
436 E757 CD46EB             CALL  :EB46     HL = SECT*DY
437 E75A 09                 DAD   B         HL = SECT*DY-DX
438 E75B 29                 DAD   H         HL = 2*(SECT*DY-DX)
439 E75C 22B500             SHLD  :00B5     Store amount to add into
440                                         count
441 E75F E1                 POP   H         Get SECT
442 E760 7C                 MOV   A,H
443 E761 1F                 RAR
444 E762 7D                 MOV   A,L
445 E763 1F                 RAR             A=INT(SECT/2)
446 E764 32BE00             STA   :00BE     Store amount to trim off
447                                         last sector
448 E767 3C                 INR   A
449 E768 6F                 MOV   L,A
450 E769 2600               MVI   H,:00     HL = INIT
451 E76B E5                 PUSH  H
452 E76C 29                 DAD   H         HL = 2*INIT
453 E76D 7B                 MOV   A,E
454 E76E CD46EB             CALL  :EB46     HL = 2*INIT*DY
455 E771 09                 DAD   B         HL = 2*INIT*DY-DX
456 E772 22B700             SHLD  :00B7     Set INIT running total
457 E775 E1                 POP   H         Get length 1st sector
458 E776 F1                 POP   PSW
459 E777 4F                 MOV   C,A       C is initial offset
460 E778 7B                 MOV   A,E
461 E779 B7                 ORA   A
462 E77A C284E7             JNZ   :E784     If more than 1 sector
463 E77D 32BE00             STA   :00BE     Store amount to trim off
464                                         last sector
465 E780 2ABB00             LHLD  :00BB     Get lower of 2 possible
466                                         sectors
467 E783 23                 INX   H         Frig length if only 1 sector
468 E784 11BD00     DRL40   LXI   D,:00BD   Addr of nr of sectors
469 E787 1A                 LDAX  D         Get count of sectors
470 E788 D601               SUI   :01       -1
471 E78A 12                 STAX  D         Store it again
472 E78B D297E7             JNC   :E797     Jump if not last sector
473
474                 * Trim off last sector:
475
476 E78E 3ABE00             LDA   :00BE     Get amount to trim off
477                                         last sector
478 E791 2F                 CMA
479 E792 5F                 MOV   E,A
480 E793 16FF               MVI   D,:FF
481 E795 13                 INX   D
482 E796 19                 DAD   D
483                 *
484 E797 110100     DRL50   LXI   D,:0001   Init Y-size is 1
485 E79A 3AC000             LDA   :00C0     ) Check if swop X,Y dir.
486 E79D B7                 ORA   A         ) (line > 45 degrees)
487 E79E CAA2E7             JZ    :E7A2     Jump if not
488 E7A1 EB                 XCHG            Swop X,Y direction
489 E7A2 43         DRL60   MOV   B,E       Get Y-size in B
490 E7A3 EB                 XCHG            X-size in DE
491 E7A4 E1                 POP   H         Get memory pointer
492 E7A5 05                 DCR   B
493 E7A6 3ABF00             LDA   :00BF     ) Check for Y-invert
494 E7A9 B7                 ORA   A         )
495 E7AA F5                 PUSH  PSW       Save condition
496 E7AB C4FCE7             CNZ   :E7FC     Move pntr to bottom of
497                                         sector
```

```
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
```

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

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

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		

```
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
```

```
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
```

```
                        * If 16-colour mode:
                        *
E89E CDF6E8                     CALL  :E8F6     Colours to buffer
E8A1 21A300                     LXI   H,:00A3   Addr SCXBUF
E8A4 F1                         POP   PSW
E8A5 CD01E7                     CALL  :E701     Calc addr in buffer
E8A8 7E                         MOV   A,M       Get colour for reqd blob
E8A9 2A9600             SSC10   LHLD  :0096     Get nr of graphics lines
E8AC 45                         MOV   B,L
E8AD 05                         DCR   B         lobyte -1 in B
E8AE 2A9400                     LHLD  :0094     Get nr of hor. blobs
E8B1 2B                         DCX   H         -1
E8B2 EB                         XCHG            in DE
E8B3 E1                         POP   H
E8B4 4D                         MOV   C,L       Y-coord in C
E8B5 E1                         POP   H         X-coord in HL
E8B6 B7                         ORA   A         CY=0
E8B7 C9                         RET
                        *
                        * If 4-colour mode:
                        *
E8B8 56                 SSC30   MOV   D,M       )
E8B9 2B                         DCX   H         ) Get screen data of
E8BA 5E                         MOV   E,M       ) point in DE
E8BB F1                         POP   PSW
E8BC 4F                         MOV   C,A       Field offset in C
E8BD 0601                       MVI   B,:01
E8BF CDE1EB                     CALL  :EBE1     Set mask for bits
E8C2 219E00                     LXI   H,:009E   Pntr to COLORG colours
E8C5 78                         MOV   A,B
E8C6 A2                         ANA   D         Test top bit result
E8C7 CACCE8                     JZ    :E8CC     Skip if 0
E8CA 23                         INX   H
E8CB 23                         INX   H
E8CC 78                 SSC40   MOV   A,B
E8CD A3                         ANA   E         Test bottom bit result
E8CE CAD2E8                     JZ    :E8D2     Skip if 0
E8D1 23                         INX   H
E8D2 7E                 SSC50   MOV   A,M       Get result from table
E8D3 E60F                       ANI   :0F       Colour bits only
E8D5 C3A9E8                     JMP   :E8A9
                        *
                        * If off screen:
                        *
E8D8 E1                 SSC99   POP   H
E8D9 C1                         POP   B
E8DA 3E01                       MVI   A,:01     Error 'off screen'
E8DC 37                         STC
E8DD C9                         RET
                        *
                        ********************
                        * UPDATE A FIELD *
                        ********************
                        *
                        * Given a mask of bits to be changed, a colour to
                        * set them to and a memory address where the field
                        * starts. This routine reads, updates and replaces
                        * a field.
                        *
                        * Entry: HL: Memory address of start of field.
                        *         B: Mask of bits to be changed.
                        *         C: Colour to change to (hinibble).
```

```
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             ) Foreground colour
298 E905 0F                     RRC             ) in lonibble
299 E906 0F                     RRC             )
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 ?
```

```
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
```

```
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:
```

```
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
```

```
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
```

```
188                     *       necessary.
189                     *
190                     * Exit:  HL: New pointer.
191                     *
192 EAC1 3A9800         DADCK   LDA   :0098     Get nr bytes/line
193 EAC4 CD01E7                 CALL  :E701     Add 1 line length
194 EAC7 C5             PTRCK   PUSH  B
195 EAC8 D5                     PUSH  D
196 EAC9 EB                     XCHG
197 EACA 2A8800                 LHLD  :0088     Get addr after end graphics
198                                             area
199 EACD CDFBE6                 CALL  :E6FB     Compare HL-DE
200 EAD0 2A8400                 LHLD  :0084     Get bottom archive area
201 EAD3 44                     MOV   B,H       ) in BC
202 EAD4 4D                     MOV   C,L       )
203 EAD5 2A8200                 LHLD  :0082     Get top visible area
204 EAD8 D2E5EA                 JNC   :EAE5     Jump if pntr is below
205                                             visible screen
206 EADB CDFBE6                 CALL  :E6FB     Compare HL-DE
207 EADE DA0FD8                 JC    :D80F     Jump if pntr is off top
208                                             visible screen
209 EAE1 EB             PCK10   XCHG
210 EAE2 D1             PCK15   POP   D
211 EAE3 C1                     POP   B
212 EAE4 C9                     RET
213
214                     * If pntr is below visible screen:
215
216 EAE5 E5             PCK20   PUSH  H         )
217 EAE6 C5                     PUSH  B         ) Swop BC and HL
218 EAE7 E1                     POP   H         )
219 EAE8 C1                     POP   B         )
220 EAE9 CDFBE6                 CALL  :E6FB     Compare HL-DE
221 EAEC DAE1EA                 JC    :EAE1     Jump if within archive area
222 EAEF EB             PCK30   XCHG
223 EAF0 CDF2E6                 CALL  :E6F2     Subtract nearest boundary
224 EAF3 09                     DAD   B         Add other
225 EAF4 C3E2EA                 JMP   :EAE2
226                     *
227                     ****************************
228                     * FILL A RECTANGULAR AREA *
229                     ****************************
230                     *
231                     * Entry: DE: Width.
232                     *        B : Height.
233                     *        C : Offset.
234                     *        HL: Address.
235                     * Exit: All registers preserved.
236                     *
237 EAF7 F5             FILRT   PUSH  PSW
238 EAF8 C5                     PUSH  B
239 EAF9 D5                     PUSH  D
240 EAFA E5                     PUSH  H
241 EAFB E5                     PUSH  H
242 EAFC 79                     MOV   A,C
243 EAFD 83                     ADD   E
244 EAFE 67                     MOV   H,A       H = C + E
245 EAFF 3E00                   MVI   A,:00
246 EB01 8A                     ADC   D
247 EB02 50                     MOV   D,B
248 EB03 1F                     RAR
249 EB04 7C                     MOV   A,H
```

```
250 EB05 DA0DEB             JC    :EB0D
251 EB08 FE08               CPI   :08
252 EB0A DA40EB             JC    :EB40     If > 8: Fill only one strip
253 EB0D 1F         L2E152  RAR
254 EB0E 0F                 RRC
255 EB0F E67E               ANI   :7E
256 EB11 E3                 XTHL
257 EB12 F5                 PUSH  PSW
258 EB13 0F                 RRC
259 EB14 D602               SUI   :02
260 EB16 5F                 MOV   E,A
261 EB17 2B                 DCX   H
262 EB18 2B                 DCX   H
263 EB19 D40BEA             CNC   :EA0B     Fill block
264 EB1C 23                 INX   H
265 EB1D 23                 INX   H
266 EB1E 79                 MOV   A,C
267 EB1F D609               SUI   :09
268 EB21 2F                 CMA
269 EB22 47                 MOV   B,A
270 EB23 CDE1EB             CALL  :EBE1     Set mask for bits
271 EB26 CD57EA             CALL  :EA57     Fill strip
272 EB29 F1                 POP   PSW
273 EB2A 2F                 CMA
274 EB2B 4F                 MOV   C,A
275 EB2C 06FF               MVI   B,:FF
276 EB2E 03                 INX   B
277 EB2F 09                 DAD   B
278 EB30 F1                 POP   PSW
279 EB31 E607               ANI   :07
280 EB33 3C                 INR   A
281 EB34 47                 MOV   B,A
282 EB35 0E00               MVI   C,:00
283 EB37 CDE1EB     L2E153  CALL  :EBE1     Set mask for bits
284 EB3A CD57EA             CALL  :EA57     Fill strip
285 EB3D C338E1             JMP   :E138     Popall, ret
286
287                 * If < 1 field:
288
289 EB40 43         L2E154  MOV   B,E
290 EB41 04                 INR   B
291 EB42 E1                 POP   H
292 EB43 C337EB             JMP   :EB37     Fill a strip only
293                 *
294                 ***************************
295                 * CALCULATE HL = A * HL *
296                 ***************************
297                 *
298                 * Exit: AFBCDE preserved.
299                 *
300 EB46 F5         HLMUL   PUSH  PSW
301 EB47 D5                 PUSH  D
302 EB48 EB                 XCHG            Original HL in DE
303 EB49 210000             LXI   H,:0000   Init result
304 EB4C B7         L2E156  ORA   A
305 EB4D CA5DEB             JZ    :EB5D     Abort if ready
306 EB50 1F                 RAR             )
307 EB51 F5                 PUSH  PSW       )
308 EB52 D256EB             JNC   :EB56     ) Calc HL = A * HL
309 EB55 19                 DAD   D         )
310 EB56 EB         L2E157  XCHG            )
311 EB57 29                 DAD   H         )
```

```
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
```

```
                        *
                        ************************************
                        * FIND COLOUR IN COLORG REGISTERS *
                        ************************************
                        *
                        * Entry: C: Requested colour.
                        * Exit:  CY=1: 'Serialnr' of colour in A.
                        *        CY=0: Colour not available.
                        *        BCDEHL preserved.
                        *
EB9B C5         STR164  PUSH  B
EB9C E5                 PUSH  H
EB9D 219E00             LXI   H,:009E    Addr 1st colour byte graph
EBA0 0603               MVI   B,:03      Total 4 colour data
EBA2 7E         L2E165  MOV   A,M        Get colour
EBA3 E60F               ANI   :0F        Colour nibble only
EBA5 B9                 CMP   C          Ident to reqd one ?
EBA6 CAB2EB             JZ    :EBB2      Then colour found
EBA9 23                 INX   H          Pnts to next one
EBAA 05                 DCR   B
EBAB F2A2EB             JP    :EBA2      Get next colour
EBAE B7                 ORA   A          CY=0
EBAF E1         L2E166  POP   H
EBB0 C1                 POP   B
EBB1 C9                 RET

                        * If colour found:

EBB2 3E03       L2E167  MVI   A,:03
EBB4 90                 SUB   B          Colour'nr' in A
EBB5 37                 STC
EBB6 C3AFEB             JMP   :EBAF      Quit, CY=1
                        *
                        ************************
                        * GET MEMORY POINTER *
                        ************************
                        *
EBB9 D5         SMEMMK  PUSH  D
EBBA F5                 PUSH  PSW
EBBB 78                 MOV   A,B
EBBC 0F                 RRC
EBBD 79                 MOV   A,C
EBBE 1F                 RAR
EBBF 1F                 RAR
EBC0 E67E               ANI   :7E
EBC2 C604               ADI   :04
EBC4 5F                 MOV   E,A
EBC5 1600               MVI   D,:00
EBC7 E1                 POP   H
EBC8 E5                 PUSH  H
EBC9 6C                 MOV   L,H        Entry A in L
EBCA 2600               MVI   H,:00
EBCC 23                 INX   H          +1
EBCD 3A9800             LDA   :0098      Get nr bytes/line
EBD0 CD46EB             CALL  :EB46      Calc HL=A*HL
EBD3 CDF2E6             CALL  :E6F2      HL=HL-DE
EBD6 EB                 XCHG             Result in DE
EBD7 2A8800             LHLD  :0088      Get addr after end graph
                                         area
EBDA 19                 DAD   D          Add offset
EBDB CDC7EA             CALL  :EAC7      Check and move pntr
EBDE F1                 POP   PSW
```

```
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