```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    5973 ;TRUE            EQU             1
                    5974           ;EXT  VRAM_WRITE,REG_WRITE,VRAM_READ
                    5975           ;EXT  VDP_MODE_WORD
                    5976           ;EXT  MUX_SPRITES
                    5977           ;EXT  PARAM_
                    5978           ;EXT  LOCAL_SPR_TBL,SPRITE_ORDER
                    5979            GLB  INIT_TABLE_,GET_VRAM_,PUT_VRAM_,INIT_SPR_ORDER_,WR_SPR_NM_TBL_
                    5980            GLB  INIT_TABLEQ,GET_VRAMQ,PUT_VRAMQ,INIT_SPR_ORDERQ,WR_SPR_NM_TBLQ
                    5981
                    5982                 PROG
                    5983 * PROCEDURE INIT_TABLEQ (TABLE_CODE:BYTE;TABLE_ADDRESS:INTEGER)
                    5984
                    5985 * THIS IS THE PASCAL ENTRY POINT TO INIT_TABLE_
                    5986
 1808 00020001      5987 INIT_TABLE_P    DEFW            2,1,2
 180C 0002
                    5988 * THIS IS THE PARAMETER DESCRIPTOR FOR INIT_TABLEQ
                    5989
 180E               5990 INIT_TABLEQ
 180E 011B08        5991                 LD              BC,INIT_TABLE_P
 1811 11738A        5992                 LD              DE,PARAM_AREA
 1814 CD0098        5993                 CALL            PARAM_
 1817 3A738A        5994                 LD              A,[PARAM_AREA]
 181A 2A738B        5995                 LD              HL,[PARAM_AREA+1]
                    5996
 181D               5997 INIT_TABLE_                     ;INIT_TABLE_ INITIALIZES THE TABLE ADDRESSES
                    5998                                 ;FOR VRAM TABLES. IT ALSO WRITES THE APPROPRIATE
                    5999                                 ;BASE ADDRESS INTO THE RESPECTIVE VDP REGISTER.
                    6000                                 ;
                    6001                                 ;IN: VRAM ADDRESS IN HL
                    6002                                 ;    TABLE CODE IN 'A' : 0=SPRITE NAME TABLE
                    6003                                 ;    1=SPRITE GENERATOR TABLE, 2=PATTERN NAME
                    6004                                 ;    TABLE,3= PATTERN GENERATOR TABLE, 4=
                    6005                                 ;    COLOR TABLE
                    6006
 181D 4F            6007           LD   C,A
 181E 0600          6008           LD   B,0              ;USE TABLE CODE AS INDEX
 1820 DD2173F2      6009           LD   IX,VRAM_ADDR_TABLE
 1824 DD09          6010           ADD  IX,BC
 1826 DD09          6011           ADD  IX,BC
 1828 DD7500        6012           LD   [IX+0],L         ;SAVE VRAM ADDRESS IN TABLE
 182B DD7401        6013           LD   [IX+1],H
                    6014                                 ;CHECK VDP GRAPHICS MODE
 182E 3A73C3        6015           LD   A,[VDP_MODE_WORD]
 1831 CB4F          6016           BIT  1,A
 1833 2827          6017           JR   Z,INIT_TABLE80
                    6018           ;**  GRAPHICS MODE 2
 1835 79            6019           LD   A,C              ;GET TABLE CODE
 1836 FE03          6020           CP   3                ;CHECK IF GENERATOR TABLE
 1838 2806          6021           JR   Z,CASE_OF_GEN
 183A FE04          6022           CP   4                ;CHECK IF COLOR TABLE
 183C 2810          6023           JR   Z,CASE_OF_COLOR
 183E 181C          6024           JR   INIT_TABLE80
 1840               6025 CASE_OF_GEN
 1840 0604          6026           LD   B,4              ;VDP REGISTER TO WRITE
 1842 7D            6027           LD   A,L              ;CHECK FOR WHICH 8K BOUNDARY IN 16K VRAM
 1843 B4            6028           OR   H
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1844 2004        6029            JR   NZ,CASE_OF_GEN10
    1846 0E03        6030            LD   C,3              ;VALUE TO WRITE FOR VRAM ADDRESS OF 00H
    1848 1828        6031            JR   INIT_TABLE90
    184A             6032 CASE_OF_GEN10
    184A 0E07        6033            LD   C,7              ;VALUE TO WRITE FOR VRAM ADDRESS OF 2000H
    184C 1824        6034            JR   INIT_TABLE90
    184E             6035 CASE_OF_COLOR
    184E 0603        6036            LD   B,3              ;REGISTER TO WRITE
    1850 7D          6037            LD   A,L
    1851 B4          6038            OR   H
    1852 2004        6039            JR   NZ,CASE_OF_CLR10
    1854 0E7F        6040            LD   C,7FH            ;VALUE TO WRITE FOR VRAM ADDRESS OF 00H
    1856 181A        6041            JR    INIT_TABLE90
    1858             6042 CASE_OF_CLR10
    1858 0EFF        6043            LD   C,0FFH           ;VALUE TO WRITE FOR VRAM ADDRESS OF 2000H
    185A 1816        6044            JR   INIT_TABLE90
                     6045
    185C             6046 INIT_TABLE80
                     6047            ;**  COMPUTE BASE ADDRESS (BASE_ADDRESS=TABLE_ADDRESS/FACTOR
                     6048            ;**  GET BIT SHIFT COUNT
    185C FD211B76    6049            LD   IY,BASE_FACTORS
    1860 FD09        6050            ADD  IY,BC
    1862 FD09        6051            ADD  IY,BC
    1864 FD7E00      6052            LD   A,[IY+0]         ;SHIFT COUNT NOW IN 'A'
    1867 FD4601      6053            LD   B,[IY+1]         ;REGISTER NUMBER TO WRITE IN 'B'
    186A             6054 DIVIDE     ;**  COMPUTE BASE ADDRESS
    186A CB3C        6055            SRL  H                ;SHIFT HI BYTE
    186C CB1D        6056            RR   L                ;SHIFT LO BYTE
    186E 3D          6057            DEC  A                ;DECREMENT SHIFT COUNT
    186F 20F9        6058            JR   NZ,DIVIDE
                     6059            ;**  WRITE TO VDP REGISTER
    1871 4D          6060            LD   C,L              ;VALUE TO WRITE IN 'C'
    1872             6061 INIT_TABLE90
    1872 CD1CCA      6062            CALL REG_WRITE
    1875 C9          6063            RET
                     6064
    1876             6065 BASE_FACTORS    ;BASE_FACTOR,REGISTER_NUMBER
                     6066
    1876 070508060A  6067            DEFB 7,5,11,6,10,2,11,4,6,3
    187B 0208040603
                     6068
                     6069 * PROCEDURE GET_VRAMQ (TABLE_CODE:BYTE;START_INDEX:BYTE;SLICE:BYTE;
                     6070 *                      VAR DATA:BUFFER;ITEM_COUNT:INTEGER);
                     6071
                     6072 * THIS IS THE PASCAL ENTRY POINT TO INIT_TABLE_
                     6073
    1880 00050001    6074 GET_VRAM_P      DEFW            5,1,1,1,-2,2
    1884 00010001
    1888 FFFE0002
                     6075 * THIS IS THE PARAMETER DESCRIPTOR FOR INIT_TABLEQ
                     6076
    188C             6077 GET_VRAMQ
    188C 011B80      6078                 LD              BC,GET VRAM P
    188F 11738A      6079                 LD              DE,PARAM_AREA
    1892 CD0098      6080                 CALL            PARAM_
    1895 3A738A      6081                 LD              A,[PARAM AREA]
    1898 ED5B7388    6082                 LD              DE,[PARAM_AREA+1]
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    189C FD2A738F     6083                LD              IY,[PARAM_AREA+5]
    18A0 2A738D       6084                LD              HL,[PARAM_AREA+3]
                      6085
                      6086
    18A3              6087 GET_VRAM_                      ;GETS A CERTAIN NUMBER OF BYTES FROM VRAM
                      6088                                ;AND PUTS THEM IN A BUFFER
                      6089                                ;IN: TABLE CODE IN A
                      6090                                ;    0=SPRITE NAME TABLE
                      6091                                ;    1=SPRITE GENERATOR TABLE, 2=PATTERN NAME
                      6092                                ;    TABLE,3= PATTERN GENERATOR TABLE, 4=
                      6093                                ;    COLOR TABLE
                      6094                                ; START_INDEX IN DE,
                      6095                                ; DATA BUFFER IN HL, AND COUNT IN IY.
                      6096
    18A3 CD18AA       6097            CALL SET_COUNT
    18A6 CD1D3E       6098            CALL VRAM_READ
    18A9 C9           6099            RET
                      6100
                      6101
    18AA              6102 SET_COUNT                      ;CALLED BY PUT_VRAM_ AND GET_VRAM_
                      6103                                ;SETS BYTE COUNT AND INDEX FOR
                      6104                                ;WRITES AND READS TO AND FROM VRAM
                      6105                                ; TABLE          BYTES/ITEM
                      6106                                ; SPRITE_NAME        4
                      6107                                ; SPRITE_GEN         8
                      6108                                ; PATTERN_NAME       1
                      6109                                ; PATTERN_GEN        8
                      6110                                ; COLOR (MODE 1)     1
                      6111                                ; COLOR (MODE 2)     8
                      6112
    18AA FD2273FE     6113            LD   [SAVED_COUNT],IY               ;SAVE COUNT
    18AE DD2173F2     6114            LD   IX,VRAM_ADDR_TABLE             ;POINTER TO SAVED VRAM ADDRESSES
    18B2 4F           6115            LD   C,A            ;SAVE TABLE CODE
    18B3 0600         6116            LD   B,0            ;BC USED AS INDEX
    18B5 FE04         6117            CP   4                              ;CHECK FOR CASE OF COLOR TABLE
    18B7 2007         6118            JR   NZ,SET_COUNT10
                      6119            ;**  COLOR TABLE, CHECK IF MODE 1
    18B9 3A73C3       6120            LD   A,[VDP_MODE_WORD]              ;CHECK FOR VDP GRAPHICS MODE
    18BC CB4F         6121            BIT  1,A
    18BE 282C         6122            JR   Z,SET_COUNT20
                      6123            ;**  NOT MODE 1, ADJUST ITEM COUNT AND INDEX
    18C0              6124 SET_COUNT10
    18C0 FD211BFF     6125            LD   IY,SHIFT_CT    ;GET ITEM COUNT CONVERSION FACTORS
    18C4 FD09         6126            ADD  IY,BC
    18C6 FD7E00       6127            LD   A,[IY+0]       ;SHIFT COUNT FOR MULTIPLICATION
    18C9 FE00         6128            CP   0
    18CB 281F         6129            JR   Z,SET_COUNT20
    18CD              6130 ADJUST_INDEX                   ;MULTIPLY ITEM_INDEX TO GET BYTE_INDEX
    18CD CB23         6131            SLA  E
    18CF CB12         6132            RL   D
    18D1 3D           6133            DEC  A
    18D2 20F9         6134            JR   NZ,ADJUST_INDEX
    18D4              6135 END_ADJ_INDEX
    18D4 C5           6136            PUSH BC             ;SAVE TABLE_CODE/INDEX
    18D5 ED4B73FE     6137            LD   BC,[SAVED_COUNT]
    18D9 FD7E00       6138            LD   A,[IY+0]       ;SHIFT COUNT FOR MULTIPLICATION
    18DC FE00         6139            CP   0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1BDE 2808        6140           JR   Z,END_ADJ_COUNT
    1BE0             6141 ADJUST_COUNT                    ;MULTIPLY ITEM_COUNT TO GET BYTE COUNT
    1BE0 CB21        6142           SLA  C
    1BE2 CB10        6143           RL   B
    1BE4 3D          6144           DEC  A
    1BE5 20F9        6145           JR   NZ,ADJUST_COUNT
    1BE7 ED4373FE    6146           LD   [SAVED_COUNT],BC          ;SAVE ADJUSTED COUNT
    1BEB             6147 END_ADJ_COUNT
                     6148
    1BEB C1          6149           POP  BC              ;RESTORE TABLE_CODE/INDEX
    1BEC             6150 SET_COUNT20
    1BEC E5          6151           PUSH HL
    1BED DD09        6152           ADD  IX,BC           ;GET TABLE ADDRESS IN VRAM
    1BEF DD09        6153           ADD  IX,BC
    1BF1 DD6E00      6154           LD   L,[IX+0]        ;LOW ORDER OF VRAM ADDRESS
    1BF4 DD6601      6155           LD   H,[IX+1]        ;HIGH ORDER OF VRAM ADDRESS
    1BF7 19          6156           ADD  HL,DE           ;ADD BYTE_INDEX TO GET VRAM START ADDRESS
    1BF8 EB          6157           EX   DE,HL           ;VRAM DESTINATION NOW IN DE
    1BF9 E1          6158           POP  HL              ;RESTORE DATA POINTER
    1BFA ED4B73FE    6159           LD   BC,[SAVED_COUNT] ;RESTORE ADJUSTED COUNT
    1BFE             6160 SET_COUNTX
    1BFE C9          6161           RET
                     6162
                     6163
    1BFF             6164 SHIFT_CT
    1BFF 0203000303  6165                DEFB           2,3,0,3,3
                     6166
                     6167
                     6168 * PROCEDURE PUT_VRAMQ (TABLE_CODE:BYTE;START_INDEX:BYTE;SLICE:BYTE;
                     6169 *                     VAR DATA:BUFFER;ITEM_COUNT:INTEGER);
                     6170
                     6171 * THIS IS THE PASCAL ENTRY POINT TO INIT_TABLE_
                     6172
    1C04 00050001    6173 PUT_VRAM_P      DEFW           5,1,1,1,-2,2
    1C08 00010001
    1C0C FFFE0002
                     6174 * THIS IS THE PARAMETER DESCRIPTOR FOR INIT_TABLEQ
                     6175
    1C10             6176 PUT_VRAMQ
    1C10 011C04      6177                LD             BC,PUT_VRAM_P
    1C13 11738A      6178                LD             DE,PARAM_AREA
    1C16 CD0098      6179                CALL           PARAM_
    1C19 3A738A      6180                LD             A,[PARAM_AREA]
    1C1C ED5B7388    6181                LD             DE,[PARAM_AREA+1]
    1C20 FD2A738F    6182                LD             IY,[PARAM_AREA+5]
    1C24 2A738D      6183                LD             HL,[PARAM_AREA+3]
                     6184
    1C27             6185 PUT_VRAM_                      ;WRITES A CERTAIN NUMBER OF BYTES TO VRAM
                     6186                               ;FROM A BUFFER.
                     6187                               ;IN: TABLE CODE IN A
                     6188                               ;    0=SPRITE NAME TABLE
                     6189                               ;    1=SPRITE GENERATOR TABLE, 2=PATTERN NAME
                     6190                               ;    TABLE,3= PATTERN GENERATOR TABLE, 4=
                     6191                               ;    COLOR TABLE
                     6192                               ; START INDEX IN DE,
                     6193                               ; DATA BUFFER IN HL, AND COUNT IN IY.
                     6194
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1C27 F5          6195           PUSH  AF
                     6196 *   IF (TABLE_CODE = SPRITE_NAME_TABLE) AND (MUX_SPRITES = TRUE) THEN
    1C28 FE00        6197           CP 0
    1C2A 2022        6198           JR NZ,ELSEZZ
    1C2C 3A73C7      6199           LD A,[MUX_SPRITES]
    1C2F FE01        6200           CP 1
    1C31 201B        6201           JR NZ,ELSEZZ
                     6202
                     6203 *     WRITE ENTRY TO LOCAL TABLE
    1C33 F1          6204           POP  AF                  ; CLEAR STACK
    1C34 E5          6205           PUSH  HL  ; [SP] = DATA BUFFER
                     6206
    1C35 2A8002      6207           LD  HL,[LOCAL_SPR_TBL] ; CALCULATE ADDRESS IN TABLE
    1C38 7B          6208           LD  A,E
    1C39 CB27        6209           SLA  A
    1C3B CB27        6210           SLA  A
    1C3D 5F          6211           LD  E,A
    1C3E 19          6212           ADD HL,DE
    1C3F EB          6213           EX  DE,HL
                     6214
    1C40 FDE5        6215           PUSH  IY                 ; CALCULATE BYTE COUNT
    1C42 C1          6216           POP  BC
    1C43 79          6217           LD  A,C
    1C44 CB27        6218           SLA  A
    1C46 CB27        6219           SLA  A
    1C48 4F          6220           LD  C,A
                     6221
    1C49 E1          6222           POP  HL                  ; STACK CLEAR
    1C4A EDB0        6223           LDIR                     ; PERFORM WRITE FROM BUFFER
                     6224
    1C4C 1807        6225           JR END_IFZZ
                     6226 *   ELSE
    1C4E             6227 ELSEZZ
    1C4E F1          6228           POP  AF
    1C4F CD1BAA      6229           CALL SET_COUNT
    1C52 CD1D01      6230           CALL VRAM_WRITE
                     6231 *   END IF
    1C55             6232 END_IFZZ
    1C55 C9          6233           RET
                     6234
                     6235
                     6236 * PROCEDURE INIT_SPR_ORDERQ  (SPRITE_COUNT:BYTE);
                     6237
                     6238 * THIS IS THE PASCAL ENTRY POINT TO THE INIT_SPR_ORDER_ ROUTINE.
                     6239
    1C56 00010001    6240 INIT_SPR_P      DEFW          1,1
                     6241
          <1C5A>     6242 INIT_SPR_ORDERQ EQU           $
    1C5A 011C56      6243                 LD            BC,INIT_SPR_P
    1C5D 1173BA      6244                 LD            DE,PARAM_AREA
    1C60 CD0098      6245                 CALL          PARAM
    1C63 3A73BA      6246                 LD            A,[PARAM_AREA]
                     6247
    1C66             6248 INIT_SPR_ORDER_              ;INITIALIZES THE SPRITE DISPLAY ORDER
                     6249                              ;LIST IN RAM TO DEFAULT ORDER (0...31)
                     6250                              ;IN: NUMBER OF SPRITES TO ORDER IN 'A'
                     6251
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

   1C66 47          6252            LD   B,A             ;SAVE SPRITE COUNT
   1C67 AF          6253            XOR  A
   1C68 2A8004      6254            LD   HL,[SPRITE_ORDER]
   1C6B             6255 INIT_SPR10
   1C6B 77          6256            LD   [HL],A
   1C6C 23          6257            INC  HL
   1C6D 3C          6258            INC  A
   1C6E B8          6259            CP   B
   1C6F 20FA        6260            JR   NZ,INIT_SPR10
   1C71 C9          6261            RET
                    6262
                    6263 * PROCEDURE WR_SPR_NM_TBLQ  (SPRITE_COUNT:BYTE);
                    6264
                    6265 * THIS IS THE PASCAL ENTRY POINT TO THE WR_SPR_NM_TBL_ ROUTINE.
                    6266
   1C72 00010001    6267 WR_SPR_P        DEFW            1,1
                    6268
         <1C76>     6269 WR_SPR_NM_TBLQ  EQU             $
   1C76 011C72      6270                 LD              BC,WR_SPR_P
   1C79 11738A      6271                 LD              DE,PARAM_AREA
   1C7C CD0098      6272                 CALL            PARAM_
   1C7F 3A738A      6273                 LD              A,[PARAM_AREA]
                    6274
   1C82             6275 WR_SPR_NM_TBL_                   ;WRITES SPRITE_NAME_TABLE TO VRAM
                    6276                                 ;USING THE SPRITE ORDER LIST.
                    6277                                 ;
                    6278                                 ;    NUMBER OF SPRITES TO WRITE IN 'A'
                    6279
                    6280 ;MODE_0_PORT     EQU             0BEH
                    6281 ;MODE_1_PORT     EQU             0BFH
                    6282
   1C82 DD2A8004    6283            LD   IX,[SPRITE_ORDER]  ;LIST OF DISPLAY ORDERS
                    6284 ;          LD   [SPRITE_CT],A      ;SAVE COUNT
   1C86 F5          6285            PUSH AF                 ;SAVE COUNT
   1C87 FD2173F2    6286            LD   IY,VRAM_ADDR_TABLE
   1C8B FD5E00      6287            LD   E,[IY+0]
   1C8E FD5601      6288            LD   D,[IY+1]           ;VRAM SPRITE_NAME_TABLE ADDRESS NOW IN DE.
                    6289
                    6290            ;**  SET UP VDP TO RECEIVE DATA
                    6291
   1C91 7B          6292            LD   A,E
   1C92 D3BF        6293            OUT  [MODE_1_PORT],A
   1C94 7A          6294            LD   A,D
   1C95 F640        6295            OR   40H
   1C97 D3BF        6296            OUT  [MODE_1_PORT],A
                    6297
   1C99 F1          6298            POP  AF                 ;RESTORE COUNT
   1C9A             6299 OUTPUT_LOOP_TABLE_MA
   1C9A 2A8002      6300            LD   HL,[LOCAL_SPR_TBL] ;RESTORE RAM SPRITE_NAME_TABLE POINTER
   1C9D DD4E00      6301            LD   C,[IX+0]           ;DISPLAY ORDER
   1CA0 DD23        6302            INC  IX                 ;ADVANCE DISPLAY ORDER POINTER
   1CA2 0600        6303            LD   B,0
   1CA4 09          6304            ADD  HL,BC
   1CA5 09          6305            ADD  HL,BC
   1CA6 09          6306            ADD  HL,BC
   1CA7 09          6307            ADD  HL,BC
                    6308
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6309            ;**  OUTPUT TO VRAM THROUGH VDP
                     6310
    1CA8 0604        6311            LD   B,4              ;ELEMENT COUNT FOR ONE SPRITE
    1CAA 0EBE        6312            LD   C,MODE_0_PORT    ;OUTPUT PORT IN 'C'
    1CAC             6313 OUTPUT_LOOP10
    1CAC EDA3        6314            OUTI
    1CAE 00          6315            NOP                   ;DELAY
    1CAF 00          6316            NOP
    1CB0 20FA        6317            JR   NZ,OUTPUT_LOOP10
                     6318 ;          LD   A,[SPRITE_CT]
                     6319 ;          DEC  A
                     6320 ;          LD   [SPRITE_CT],A
    1CB2 3D          6321            DEC  A
    1CB3 20E5        6322            JR   NZ,OUTPUT_LOOP_TABLE_MA
    1CB5 C9          6323            RET
                     6324
                     6325                 GLB              VRAM_ADDR_TABLE
                     6326                 GLB              SPRITENAMETBL
                     6327                 GLB              SPRITEGENTBL
                     6328                 GLB              PATTRNNAMETBL
                     6329                 GLB              PATTRNGENTBL
                     6330                 GLB              COLORTABLE
                     6331                 DATA
    73F2             6332 VRAM_ADDR_TABLE
    73F2             6333 SPRITENAMETBL   DEFS             2
    73F4             6334 SPRITEGENTBL    DEFS             2
    73F6             6335 PATTRNNAMETBL   DEFS             2
    73F8             6336 PATTRNGENTBL    DEFS             2
    73FA             6337 COLORTABLE      DEFS             2
                     6338 * THIS TABLE HOLDS THE BASE ADDRESSES OF ALL THE VRAM TABLES.
                     6339
                     6340
    73FC             6341 SAVE_TEMP       DEFS 2
    73FE             6342 SAVED_COUNT     DEFS 2
                     6343
                     6344 ;               COMN
                     6345
                     6346 ;PARAM_AREA      DEFS             6
                     6347 * THIS IS THE PARAMETER PASSING AREA FOR THE PASCAL ENTRY POINTS TO
                     6348 * ROUTINES IN THIS MODULE. IT IS HELD IN COMMON WITH OTHER SUCH ENTRY
                     6349 * POINTS FOR OTHER MODULES.
                     6350  PROG
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

                    6465 * THIS IS THE PARAMETER DESCRIPTOR FOR REG_WRITEQ
                    6466
                    6467 * BEGIN REG_WRITE
                    6468                    GLB             REG_WRITEQ
          <1CBC>    6469 REG_WRITEQ         EQU             $
  1CBC 011CB6       6470                    LD              BC,REG_WRITE_P
  1CBF 11738A       6471                    LD              DE,PARAM_AREA
  1CC2 CD0098       6472                    CALL            PARAM_
  1CC5 2A738A       6473                    LD              HL,[PARAM_AREA]
  1CC8 4C           6474                    LD              C,H
  1CC9 45           6475                    LD              B,L
                    6476
                    6477                    GLB             REG_WRITE
          <1CCA>    6478 REG_WRITE          EQU             $
                    6479
                    6480 *   VALUE =; CTRL_PORT
  1CCA 79           6481                    LD              A,C
  1CCB D38F         6482                    OUT             [CTRL_PORT],A
                    6483
                    6484 *   REGISTER + 80H =; CTRL_PORT
  1CCD 78           6485                    LD              A,B
  1CCE C680         6486                    ADD             A,80H
  1CD0 D38F         6487                    OUT             [CTRL_PORT],A
                    6488
                    6489 *   IF REGISTER = 0 THEN VDP_MODE_WORD[0] := VALUE
  1CD2 78           6490                    LD              A,B
  1CD3 FE00         6491                    CP              0
  1CD5 2004         6492                    JR              NZ,NOT_REG_0
  1CD7 79           6493                    LD              A,C
  1CD8 3273C3       6494                    LD              [VDP_MODE_WORD],A
          <1CDB>    6495 NOT_REG_0          EQU             $
                    6496
                    6497 *   IF REGISTER = 1 THEN VDP_MODE_WORD[1] := VALUE
  1CDB 78           6498                    LD              A,B
  1CDC FE01         6499                    CP              1
  1CDE 2004         6500                    JR              NZ,NOT_REG_1
  1CE0 79           6501                    LD              A,C
  1CE1 3273C4       6502                    LD              [VDP_MODE_WORD+1],A
          <1CE4>    6503 NOT_REG_1          EQU             $
                    6504
                    6505 * END REG_WRITE
  1CE4 C9           6506                    RET
                    6507
                    6508 * PROCEDURE  VRAM_WRITE (VAR DATA:BUFFER;DEST:INTEGER;COUNT:INTEGER)
                    6509
                    6510 * VAR DATA (POINTER TO DATA BUFFER) IS PASSED IN HL
                    6511 * DEST IS PASSED IN DE
                    6512 * COUNT IS PASSED IN BC
                    6513 * DESTROYS: ALL
                    6514
  1CE5 0003FFFE     6515 VRAM_WRITE_P       DEFW            3,-2,2,2
  1CE9 00020002
                    6516 * THIS IS THE PARAMETER DESCRIPTOR FOR VRAM_WRITEQ
                    6517
                    6518 * BEGIN VRAM_WRITE
                    6519                    GLB             VRAM_WRITEQ
          <1CED>    6520 VRAM_WRITEQ        EQU             $
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1CED 011CE5      6521                 LD              BC,VRAM_WRITE_P
    1CF0 11738A      6522                 LD              DE,PARAM_AREA
    1CF3 CD0098      6523                 CALL            PARAM_
    1CF6 2A738A      6524                 LD              HL,[PARAM_AREA]
    1CF9 ED5B738C    6525                 LD              DE,[PARAM_AREA+2]
    1CFD ED4B738E    6526                 LD              BC,[PARAM_AREA+4]
                     6527
                     6528                 GLB             VRAM_WRITE
          <1D01>     6529 VRAM_WRITE      EQU             $
                     6530
                     6531 * DEST := DEST + 4000H
    1D01 E5          6532                 PUSH            HL
    1D02 D5          6533                 PUSH            DE
    1D03 E1          6534                 POP             HL
    1D04 114000      6535                 LD              DE,4000H
    1D07 19          6536                 ADD             HL,DE
                     6537
                     6538 * LOW BYTE OF DEST =; CTRL_PORT
    1D08 7D          6539                 LD              A,L
    1D09 D38F        6540                 OUT             [CTRL_PORT],A
                     6541
                     6542 * HIGH BYTE OF DEST =; CTRL_PORT
    1D0B 7C          6543                 LD              A,H
    1D0C D38F        6544                 OUT             [CTRL_PORT],A
                     6545
                     6546 * DATA =; DATA_PORT
    1D0E C5          6547                 PUSH            BC
    1D0F D1          6548                 POP             DE
    1D10 E1          6549                 POP             HL
    1D11 0EBE        6550                 LD              C,DATA_PORT
    1D13 43          6551                 LD              B,E
          <1D14>     6552 OUTPUT_LOOP     EQU             $
    1D14 EDA3        6553                 OUTI
    1D16 00          6554                 NOP
    1D17 00          6555                 NOP
    1D18 C21D14      6556                 JP              NZ,OUTPUT_LOOP
    1D1B 15          6557                 DEC             D
    1D1C FA1D21      6558                 JP              M,END_OUTPUT
    1D1F 20F3        6559                 JR              NZ,OUTPUT_LOOP
                     6560
                     6561 * END VRAM_WRITE
          <1D21>     6562 END_OUTPUT      EQU             $
    1D21 C9          6563                 RET
                     6564
                     6565 * PROCEDURE  VRAM_READ  (VAR DATA:BUFFER;SRCE:INTEGER;COUNT:INTEGER)
                     6566
                     6567 * VAR DATA (POINTER TO DATA BUFFER) IS PASSED IN HL
                     6568 * SRCE IS PASSED IN DE
                     6569 * COUNT IS PASSED IN BC
                     6570 * DESTROYS: ALL
                     6571
    1D22 0003FFFE    6572 VRAM_READ_P     DEFW            3,-2,2,2
    1D26 00020002
                     6573 * THIS IS THE PARAMETER DESCRIPTOR FOR VRAM_READQ
                     6574
                     6575 * BEGIN VRAM_READ
                     6576                 GLB             VRAM_READQ
```

```
.OCATION OBJECT CODE LINE     SOURCE LINE

          <1D2A>  6577 VRAM_READQ      EQU             $
    1D2A 011D22   6578                 LD              BC,VRAM_READ_P
    1D2D 11738A   6579                 LD              DE,PARAM_AREA
    1D30 CD0098   6580                 CALL            PARAM_
    1D33 2A738A   6581                 LD              HL,[PARAM_AREA]
    1D36 ED5B738C 6582                 LD              DE,[PARAM_AREA+2]
    1D3A ED4B738E 6583                 LD              BC,[PARAM_AREA+4]
                  6584
                  6585                 GLB             VRAM_READ
          <1D3E>  6586 VRAM_READ       EQU             $
                  6587
                  6588 * LOW BYTE OF SRCE =; CTRL_PORT
    1D3E 7B       6589                 LD              A,E
    1D3F D38F     6590                 OUT             [CTRL_PORT],A
                  6591
                  6592 * HIGH BYTE OF SRCE =; CTRL_PORT
    1D41 7A       6593                 LD              A,D
    1D42 D38F     6594                 OUT             [CTRL_PORT],A
                  6595
                  6596 * DATA =; DATA_PORT
    1D44 C5       6597                 PUSH            BC
    1D45 D1       6598                 POP             DE
    1D46 0E8E     6599                 LD              C,DATA_PORT
    1D48 43       6600                 LD              B,E
          <1D49>  6601 INPUT_LOOP      EQU             $
    1D49 EDA2     6602                 INI
    1D4B 00       6603                 NOP
    1D4C 00       6604                 NOP
    1D4D C2 1D49  6605                 JP              NZ,INPUT_LOOP
    1D50 15       6606                 DEC             D
    1D51 FA1D56   6607                 JP              M,END_INPUT
    1D54 20F3     6608                 JR              NZ,INPUT_LOOP
                  6609
                  6610 * END VRAM_READ
          <1D56>  6611 END_INPUT       EQU             $
    1D56 C9       6612                 RET
                  6613
                  6614 * FUNCTION   REG_READ:BYTE
                  6615
                  6616 * FUNCTION OUTPUT RETURNED IN A
                  6617 * DESTROYS A ONLY
                  6618
                  6619 * BEGIN REG_READ
                  6620                 GLB             REG_READ
          <1D57>  6621 REG_READ        EQU             $
                  6622
                  6623 *   REG_READ ;= CTRL_PORT
    1D57 DB8F     6624                 IN              A,[CTRL_PORT]
                  6625
                  6626 * END REG_READ
    1D59 C9       6627                 RET
                  6628
                  6629  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6631
                     6632 * THIS IS A PACKAGE OF ROUTINES THAT ALLOW APPLICATIONS PROGRAMMERS TO
                     6633 * OPERATE ON SHAPE GENERATORS. EACH OF THEM TAKES, AS INPUTS, AN AREA
                     6634 * IN ONE OF THE GENERATOR TABLES IN WHICH THE GENERATORS TO BE OPERATED
                     6635 * UPON RESIDE, A COUNT OF THE GENERATORS TO BE USED, AND AN AREA OF THE
                     6636 * SAME TABLE INTO WHICH THE RESULTS ARE TO BE PUT.THE ONLY RAM AREA THEY
                     6637 * IS IN THE WORK_BUFFER A POINTER TO WHICH IS DECLARED AS AN EXTERNAL
                     6638 * AND DEFINED IN THE CARTRIDGE.
                     6639
                     6640 ************ NOTE:                                    ******************
                     6641 ************      THESE ROUTINES WRITE TO AND READ  ******************
                     6642 ************      WITHOUT POSSIBILITY OF DEFERAL    ******************
                     6643 ************      AND SHOULD NOT BE USED IN ANY     ******************
                     6644 ************      SITUATION WHERE THEY MAY BE       ******************
                     6645 ************      INTERRUPTED.                      ******************
                     6646 ***************************************************************************
                     6647
                     6648 ;               EXT             WORK_BUFFER
                     6649 * POINTER TO THE WORK BUFFER DEFINED BY THE CARTRIDGE PROGRAMMER
                     6650
                     6651 ;               EXT             VDP_MODE_WORD
                     6652 * THE WORD IN OS RAM THAT DESCRIBES THE CURRENT GRAPHICS MODE.
                     6653
                     6654 ;               EXT             GET_VRAM_
                     6655 ;               EXT             PUT_VRAM_
                     6656 * EXTERNAL OS ROUTINES IN TABLE_MANAGER MODULE
                     6657
                     6658 ;               EXT             MIRROR_L_R
                     6659 ;               EXT             MIRROR_U_D
                     6660 ;               EXT             ROTATE
                     6661 ;               EXT             MAGNIFY
                     6662 ;               EXT             QUADRUPLE
                     6663 * EXTERNAL ROUTINES THAT PERFORM BLOCK OPERATIONS
                     6664
                     6665 ;TRUE            EQU             1
                     6666 ;FALSE           EQU             0
                     6667 * VALUES FOR BOOLEAN FLAGS
                     6668
          <0003>     6669 PATTERN_GEN     EQU             3
          <0004>     6670 COLOR_TABLE     EQU             4
                     6671 * VALUES FOR TABLE CODE
                     6672
                     6673 * PROCEDURE REFLECT_VERTICAL (TABLE_CODE(A),SOURCE(DE),DESTINATION(HL),COUNT(BC))
                     6674
                     6675 * REFLECT REFLECTS EACH OF A BLOCK OF GENERATORS FROM VRAM AROUND
                     6676 * THE VERTICAL AXIS. IF THE GENERATORS ARE FROM THE PATTERN PLANE
                     6677 * AND THE GRAPHICS MODE IS 2, THEN THE ROUTINE ALSO COPIES THE
                     6678 * CORRESPONDING COLOR GENERATORS., OTHERWISE IS ASSUMES THAT THE COLOR
                     6679 * DATA HAS ALREADY BEEN SET UP.
                     6680
                     6681 * BEGIN REFLECT_VERTICAL
                     6682                 GLB             RFLCT_VERT
  1D5A               6683 RFLCT_VERT                      ; ACTUAL ROUTINE NAME EXISTS IN OS
                     6684                                 ; JUMP TABLE ONLY
                     6685
                     6686 *  SET OPERATION CODE
  1D5A DD211D96      6687                 LD              IX,RFLCT_VERT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6688
                     6689 *  CONTINUE BELOW
     1D5E 1810       6690                    JR              CONTINUE_GRAPHICS
                     6691
                     6692
                     6693 * PROCEDURE REFLECT_HORIZONTAL (TABLE_CODE(A),SOURCE(DE),DESTINATION(HL),COUNT(BC))
                     6694
                     6695 * REFLECT_HORIZONTAL REFLECTS EACH OF A BLOCK OF GENERATORS FROM VRAM
                     6696 * AROUND THE HORIZONTAL AXIS. IF THE GENERATORS ARE FROM THE PATTERN
                     6697 * PLANE ANDTHE GRAPHICS MODE IS 2 THEN IT REFLECTS THE CORRESPONDING
                     6698 * COLOR GENERATORS AS WELL.
                     6699
                     6700 * BEGIN REFLECT_HORIZONTAL
                     6701                    GLB             RFLCT_HOR
     1D60            6702 RFLCT_HOR                          ; ACTUAL ROUTINE NAME EXISTS IN OS
                     6703                                    ; JUMP TABLE ONLY
                     6704
                     6705 *  SET OPERATION CODE
     1D60 DD211DB7   6706                    LD              IX,RFLCT_HOR_
                     6707
                     6708 *  CONTINUE BELOW
     1D64 180A       6709                    JR              CONTINUE_GRAPHICS
                     6710
                     6711
                     6712
                     6713
                     6714 * PROCEDURE ROTATE_90 (TABLE_CODE(A),SOURCE(DE),DESTINATION(HL),COUNT(BC))
                     6715
                     6716 * ROTATE_90 ROTATES EACH OF A BLOCK OF GENERATORS FROM VRAM 90 DEGREES
                     6717 * CLOCKWISE. IF THE GENERATORS ARE FROM THE PATTERN PLANE AND THE
                     6718 * GRAPHICS MODE IS 2 THEN THE ROUTINE COPIES THE CORRESPONDING COLOR
                     6719 * ENTRIES AS WELL.
                     6720
                     6721 * BEGIN ROTATE_90
                     6722                    GLB             ROT_90
     1D66            6723 ROT_90                             ; ACTUAL ROUTINE NAME EXISTS IN OS
                     6724                                    ; JUMP TABLE ONLY
                     6725
                     6726 *  SET OPERATION CODE
     1D66 DD211DE5   6727                    LD              IX,ROT_90_
                     6728
                     6729 *  CONTINUE BELOW
     1D6A 1804       6730                    JR              CONTINUE_GRAPHICS
                     6731
                     6732
                     6733
                     6734
                     6735 * PROCEDURE ENLARGE (TABLE_CODE(A),SOURCE(DE),DESTINATION(HL),COUNT(BC))
                     6736
                     6737 * ENLARGE TAKES EACH OF A BLOCK OF GENERATORS AND ENLARGES IT INTO
                     6738 * A BLOCK OF FOUR GENERATORS WHEREIN EACH PIXEL OF THE ORIGINAL
                     6739 * GENERATOR IS EXPANDED TO FOUR PIXELS IN THE NEW ONES. IF THE
                     6740 * GENERATORS ARE FROMTHE PATTERN PLANE AND THE GRAPHICS MODE IS 2
                     6741 * THE THE ROUTINE ALSO QUADRUPLES EACH OF THE CORRESPONDING COLOR
                     6742 * GENERATORS AS WELL.
                     6743
                     6744 * BEGIN ENLARGE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6745                 GLB             ENLRG
1D6C                 6746 ENLRG                           ; ACTUAL ROUTINE NAME EXISTS IN OS
                     6747                                 ; JUMP TABLE ONLY
                     6748
                     6749
                     6750 *  SET OPERATION CODE
1D6C DD211E07        6751                 LD              IX,ENLRG_
                     6752
                     6753
                     6754 *  CONTINUE EXECUTION HERE
1D70                 6755 CONTINUE_GRAPHICS
                     6756
                     6757 *  SAVE ALL ENTRY PARAMETERS
1D70 D9              6758                 EXX
1D71 08              6759                 EX              AF,AF'
1D72 DDE5            6760                 PUSH            IX              ; [SP] = OPERATION CODE
                     6761
                     6762 *  REPEAT
1D74                 6763 MAIN_LOOP
                     6764
                     6765 *    GET_VRAM_ (TABLE_CODE,SOURCE,WORK_BUFFER[0..7],1)
1D74 08              6766                 EX              AF,AF'
1D75 F5              6767                 PUSH            AF
1D76 08              6768                 EX              AF,AF'
1D77 F1              6769                 POP             AF
1D78 D9              6770                 EXX
1D79 D5              6771                 PUSH            DE
1D7A D9              6772                 EXX
1D7B D1              6773                 POP             DE
1D7C FD210001        6774                 LD              IY,1
1D80 2A8006          6775                 LD              HL,[WORK_BUFFER]
1D83 CD18A3          6776                 CALL            GET_VRAM_
                     6777
                     6778 *    EXECUTE ENCODED OPERATION BELOW
1D86 DDE1            6779                 POP             IX
1D88 DDE5            6780                 PUSH            IX
1D8A DDE9            6781                 JP              [IX]
        <1D8C>       6782 RETURN_HERE     EQU             $
                     6783
                     6784 *    SOURCE : SUCC (SOURCE)
1D8C 13              6785                 INC             DE
                     6786
                     6787 *    COUNT := PRED (COUNT)
1D8D 0B              6788                 DEC             BC
                     6789
                     6790 *  UNTIL COUNT = 0
1D8E 78              6791                 LD              A,B
1D8F B1              6792                 OR              C
1D90 D9              6793                 EXX
1D91 20E1            6794                 JR              NZ,MAIN_LOOP
                     6795
                     6796 * END (ALL)
1D93                 6797 ALL_X
1D93 DDE1            6798                 POP             IX              ; CLEAR STACK
1D95 C9              6799                 RET
                     6800
1D96                 6801 RELCT_VERT_
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

                    6802 * OPERATIONS SPECIFIC TO REFLECT_VERTICAL ROUTINE
                    6803
                    6804 *   MIRROR_L_R(WORK_BUFFER[0..7],WORK_BUFFER[8..15])
  1D96 2A8006       6805                 LD              HL,[WORK_BUFFER]
  1D99 010008       6806                 LD              BC,8
  1D9C E5           6807                 PUSH            HL
  1D9D D1           6808                 POP             DE
  1D9E 09           6809                 ADD             HL,BC
  1D9F EB           6810                 EX              DE,HL
  1DA0 CD1F00       6811                 CALL            MIRROR_L_R
                    6812
                    6813 *   PUT_VRAM (TABLE_CODE,DESTINATION,WORK_BUFFER[8..15],1)
  1DA3 CD1E72       6814                 CALL            PUT_TABLE
                    6815
                    6816 *   IF COLOR_TEST THEN
  1DA6 CD1E5D       6817                 CALL            COLOR_TEST
  1DA9 FE01         6818                 CP              TRUE
  1DAB 2006         6819                 JR              NZ,END_IF_1_GRAPHICS
                    6820
                    6821 *     GET_VRAM(COLOR_TABLE,SOURCE,WORK_BUFFER[0..7],1)
  1DAD CD1E89       6822                 CALL            GET_COLOR
                    6823
                    6824 *     PUT_VRAM(COLOR_TABLE,DESTINATION,WORK_BUFFER[0..7],1)
  1DB0 CD1E9A       6825                 CALL            PUT_COLOR
                    6826
                    6827 *   END_IF
  1DB3              6828 END_IF_1_GRAPHICS
                    6829
                    6830 *   DESTINATION := SUCC (DESTINATION)
  1DB3 D9           6831                 EXX
  1DB4 23           6832                 INC             HL
                    6833
                    6834 * END
  1DB5 18D5         6835                 JR              RETURN_HERE
                    6836
                    6837
                    6838
  1DB7              6839 RFLCT_HOR
                    6840 * OPERATIONS SPECIFIC TO REFLECT_HORIZONTAL ROUTINE
                    6841
                    6842 *   MIRROR_U_D(WORK_BUFFER[0..7],WORK_BUFFER[8..15])
  1DB7 2A8006       6843                 LD              HL,[WORK_BUFFER]
  1DBA 010008       6844                 LD              BC,8
  1DBD E5           6845                 PUSH            HL
  1DBE D1           6846                 POP             DE
  1DBF 09           6847                 ADD             HL,BC
  1DC0 EB           6848                 EX              DE,HL
  1DC1 CD1F4E       6849                 CALL            MIRROR_U_D
                    6850
                    6851 *   PUT_VRAM (TABLE_CODE,DESTINATION,WORK_BUFFER[8..15],1)
  1DC4 CD1E72       6852                 CALL            PUT_TABLE
                    6853
                    6854 *   IF COLOR_TEST THEN
  1DC7 CD1E5D       6855                 CALL            COLOR_TEST
  1DCA FE01         6856                 CP              TRUE
  1DCC 2013         6857                 JR              NZ,END_IF_2_GRAPHICS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6859 *     GET_VRAM(COLOR_TABLE,SOURCE,WORK_BUFFER[0..7],1)
     1DCE CD1E89     6860                CALL            GET_COLOR
                     6861
                     6862 *     MIRROR_U_D(WORK_BUFFER[0..7],WORK_BUFFER[8..15])
     1DD1 2A8006     6863                LD              HL,[WORK_BUFFER]
     1DD4 010008     6864                LD              BC,8
     1DD7 E5         6865                PUSH            HL
     1DD8 D1         6866                POP             DE
     1DD9 09         6867                ADD             HL,BC
     1DDA EB         6868                EX              DE,HL
     1DDB CD1F4E     6869                CALL            MIRROR_U_D
                     6870
                     6871 *     PUT_VRAM(COLOR_TABLE,DESTINATION,WORK_BUFFER[8..15],1)
     1DDE CD1E9A     6872                CALL            PUT_COLOR
                     6873
                     6874 *   END_IF
     1DE1            6875 END_IF_2_GRAPHICS
                     6876
                     6877 *   DESTINATION := SUCC (DESTINATION)
     1DE1 D9         6878                EXX
     1DE2 23         6879                INC             HL
                     6880
                     6881 * END
     1DE3 18A7       6882                JR              RETURN_HERE
                     6883
                     6884
                     6885
     1DE5            6886 ROT_90_
                     6887 * OPERATIONS SPECIFIC TO THE ROTATE_90 ROUTINE
                     6888
                     6889
                     6890 *   ROTATE(WORK_BUFFER[0..7],WORK_BUFFER[8..15])
     1DE5 2A8006     6891                LD              HL,[WORK_BUFFER]
     1DE8 010008     6892                LD              BC,8
     1DEB E5         6893                PUSH            HL
     1DEC D1         6894                POP             DE
     1DED 09         6895                ADD             HL,BC
     1DEE EB         6896                EX              DE,HL
     1DEF CD1F12     6897                CALL            ROTATE
                     6898
                     6899 *   PUT_VRAM (TABLE_CODE,DESTINATION,WORK_BUFFER[8..15],1)
     1DF2 CD1E72     6900                CALL            PUT_TABLE
                     6901
                     6902 *   IF COLOR_TEST THEN
     1DF5 CD1E5D     6903                CALL            COLOR_TEST
     1DF8 FE01       6904                CP              TRUE
     1DFA 2006       6905                JR              NZ,END_IF_3_GRAPHICS
                     6906
                     6907 *     GET_VRAM(COLOR_TABLE,SOURCE,WORK_BUFFER[0..7],1)
     1DFC CD1E89     6908                CALL            GET_COLOR
                     6909
                     6910 *     PUT_VRAM(COLOR_TABLE,DESTINATION,WORK_BUFFER[0..7],1)
     1DFF CD1E9A     6911                CALL            PUT_COLOR
                     6912
                     6913 *   END_IF
     1E02            6914 END_IF_3_GRAPHICS
                     6915
```

```
XCATION OBJECT CODE LINE     SOURCE LINE

                    6916 *   DESTINATION := SUCC (DESTINATION)
   1E02 D9          6917                EXX
   1E03 23          6918                INC             HL
                    6919
                    6920 * END
   1E04 C31D8C      6921                JP              RETURN_HERE
                    6922
                    6923
                    6924
   1E07             6925 ENLRG_
                    6926 * OPERATIONS SPECIFIC TO THE ENLARGE ROUTINE
                    6927
                    6928 *   MAGNIFY(WORK_BUFFER[0..7],WORK_BUFFER[8..39])
   1E07 2A8006      6929                LD              HL,[WORK_BUFFER]
   1E0A 010008      6930                LD              BC,8
   1E0D E5          6931                PUSH            HL
   1E0E D1          6932                POP             DE
   1E0F 09          6933                ADD             HL,BC
   1E10 EB          6934                EX              DE,HL
   1E11 CD1EAB      6935                CALL            MAGNIFY
                    6936
                    6937 *   PUT_VRAM (TABLE_CODE,DESTINATION,WORK_BUFFER[8..39],4)
   1E14 08          6938                EX              AF,AF'
   1E15 F5          6939                PUSH            AF
   1E16 08          6940                EX              AF,AF'
   1E17 F1          6941                POP             AF
   1E18 D9          6942                EXX
   1E19 E5          6943                PUSH            HL
   1E1A D9          6944                EXX
   1E1B D1          6945                POP             DE
   1E1C 2A8006      6946                LD              HL,[WORK_BUFFER]
   1E1F 010008      6947                LD              BC,8
   1E22 09          6948                ADD             HL,BC
   1E23 FD210004    6949                LD              IY,4
   1E27 CD1C27      6950                CALL            PUT_VRAM_
                    6951
                    6952 *   IF COLOR_TEST THEN
   1E2A CD1E5D      6953                CALL            COLOR_TEST
   1E2D FE01        6954                CP              TRUE
   1E2F 2024        6955                JR              NZ,END_IF_4_GRAPHICS
                    6956
                    6957 *     GET_VRAM(COLOR_TABLE,SOURCE,WORK_BUFFER[0..7],1)
   1E31 CD1E89      6958                CALL            GET_COLOR
                    6959
                    6960 *     QUADRUPLE(WORK_BUFFER[0..7],WORK_BUFFER[8..39])
   1E34 2A8006      6961                LD              HL,[WORK_BUFFER]
   1E37 010008      6962                LD              BC,8
   1E3A E5          6963                PUSH            HL
   1E3B D1          6964                POP             DE
   1E3C 09          6965                ADD             HL,BC
   1E3D EB          6966                EX              DE,HL
   1E3E CD1EEA      6967                CALL            QUADRUPLE
                    6968
                    6969 *     PUT_VRAM(COLOR_TABLE,DESTINATION,WORK_BUFFER[8..39],4)
   1E41 3E04        6970                LD              A,COLOR_TABLE
   1E43             6971                EXX
   1E44 E5          6972                PUSH            HL
```