```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1248 ;          .IDENT PLAYSON          ;Includes TONE_OUT
                     1249 ;****************************
                     1250 ;*          PLAY_SONGS_         *
                     1251 ;****************************
                     1252 ;.COMMENT )
                     1253 ;see Users' Manual for description
                     1254 ;)
                     1255            GLB PLAY_SONGS_,TONE_OUT
                     1256            ;EXT UPATNCTRL,UPFREQ
                     1257            ;INCLUDE OSSR_EQU:OS  ;equates
                     1258 ;          *** Sound chip register code EQUATES
                     1259 ;          Tone generator frequency and attenuation formatted register codes
                     1260 ;SR1FRQ  EQU 10000000B          ;BIT7 = 1, BIT6-4 = TONE GEN 1 FREQ CODE
                     1261 ;SR1ATN  EQU 10010000B          ;BIT7 = 1, BIT6-4 = TONE GEN 1 ATTN CODE
                     1262 ;SR2FRQ  EQU 10100000B          ;BIT7 = 1, BIT6-4 = TONE GEN 2 FREQ CODE
                     1263 ;SR2ATN  EQU 10110000B          ;BIT7 = 1, BIT6-4 = TONE GEN 2 ATTN CODE
                     1264 ;SR3FRQ  EQU 11000000B          ;BIT7 = 1, BIT6-4 = TONE GEN 3 FREQ CODE
                     1265 ;SR3ATN  EQU 11010000B          ;BIT7 = 1, BIT6-4 = TONE GEN 1 ATTN CODE
                     1266 ;          Noise generator control and attenuation formatted register codes
                     1267 ;SRNCTL  EQU 11100000B          ;BIT7 = 1, BIT6-4 = NOISE GEN CONTROL CODE
                     1268 ;SRNATN  EQU 11110000B          ;BIT7 = 1, BIT6-4 = NOISE GEN ATTN CODE
                     1269 ;          Noise generator formatted control codes
             <0004>  1270 WHITE    EQU 00000100B          ;BIT2 = 1, WHITE NOISE CODE
             <0000>  1271 PERIOD   EQU 00000000B          ;BIT2 = 0, PERIODIC NOISE CODE
             <0000>  1272 NSRHI    EQU 00000000B          ;BIT0-1 SET FOR HIGHEST NOISE SHIFT RATE [N/512]
             <0001>  1273 NSRMED   EQU 00000001B          ;BIT0-1 SET FOR MEDIUM NOISE SHIFT RATE [N/1024]
             <0002>  1274 NSRLOW   EQU 00000010B          ;BIT0-1 SET FOR LOWEST NOISE SHIFT RATE [N/2048]
             <0003>  1275 NSRTG3   EQU 00000011B          ;BIT0-1 SET FOR SHIFT FROM TONE GEN 3 OUTPUT
  0300               1276 PLAY_SONGS_
                     1277 ;          * output CH1 attenuation and frequency
  0300 3E9F          1278           LD A,OFF+SR1ATN        ;format CH1 OFF byte into A
  0302 0E90          1279           LD C,SR1ATN            ;format MSN C for CH1 attenuation
  0304 1680          1280           LD D,SR1FRQ            ;format MSN D for CH1 frequency
  0306 DD2A7024      1281           LD IX,[PTR_TO_S_ON_1]  ;point IX to byte 0 data area of song for CH1
  030A CD034E        1282           CALL TONE_OUT
                     1283 ;          * output CH2 attenuation and frequency
  030D 3EBF          1284           LD A,OFF+SR2ATN        ;format CH2 OFF byte into A
  030F 0EB0          1285           LD C,SR2ATN            ;format MSN C for CH2 attenuation
  0311 16A0          1286           LD D,SR2FRQ            ;format MSN D for CH2 frequency
  0313 DD2A7026      1287           LD IX,[PTR_TO_S_ON_2]  ;point IX to byte 0 data area of song for CH2
  0317 CD034E        1288           CALL TONE_OUT
                     1289 ;          * output CH3 attenuation and frequency
  031A 3EDF          1290           LD A,OFF+SR3ATN        ;format CH3 OFF byte into A
  031C 0ED0          1291           LD C,SR3ATN            ;format MSN C for CH3 attenuation
  031E 16C0          1292           LD D,SR3FRQ            ;format MSN D for CH3 frequency
  0320 DD2A7028      1293           LD IX,[PTR_TO_S_ON_3]  ;point IX to byte 0 data area of song for CH3
  0324 CD034E        1294           CALL TONE_OUT
                     1295 ;          * output CH0 [noise] ATN [and CTRL, if different from last time]
  0327 3EFF          1296           LD A,OFF+SRNATN        ;format CH0 OFF byte into A
  0329 0EF0          1297           LD C,SRNATN            ;format MSN C for CH0 attenuation
  032B DD2A7022      1298           LD IX,[PTR_TO_S_ON_0]  ;point IX to byte 0 data area of song for CH0
  032F DD5E00        1299           LD E,[IX+0]            ;look for inactive code, 0FFH
  0332 1C            1300           INC E                  ;this sets Z flag if E = 0FFH
                     1301 ;          IF [PSW,IS,ZERO]       ; song data area is inactive
  0333 2004          1302           JR NZ,L5
  0335 D3FF          1303             OUT [SOUND_PORT],A      ;turn off CH0
  0337 1814          1304             JR L6
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  03A1 C30461        1391               JP MODB0                 ;to load byte 0
                     1392 ;          ENDIF
                     1393 ;          - test for special sound effect
  03A4 E63C          1394 L14       AND 00111100B              ;mask irrelevant bits
  03A6 FE04          1395           CP 00000100B               ;test for B5 - B2 = 0001
                     1396 ;         IF [PSW,IS,ZERO]           ;note is a special effect
  03A8 2028          1397           JR NZ,L15
                     1398             ;--CASE-- special effect
  03AA FDE1          1399 EFFECT     POP IY                    ;IY := SONGNO
  03AC FDE5          1400            PUSH IY                   ;put SONGNO back on stack
  03AE C5            1401            PUSH BC                   ;save header on stack; NEXT_NOTE_PTR := SFX, DE := SFX
  03AF 23            1402            INC HL                    ;-pt HL to next byte [LSB addr SFX]
  03B0 5E            1403            LD E,[HL]                 ;-E := LSB SFX
  03B1 DD7301        1404            LD [IX+1],E               ;-put LSB of SFX in byte 1 of SxDATA [NEXT_NOTE_PTR]
  03B4 23            1405            INC HL                    ;-pt HL to MSB SFX
  03B5 56            1406            LD D,[HL]                 ;-D := MSB SFX
  03B6 DD7202        1407            LD [IX+2],D               ;-put MSB SFX in byte 2 of SxDATA
  03B9 23            1408            INC HL                    ;point HL to next note [after this new note]
  03BA FDE5          1409            PUSH IY                   ;A := SONGNO
  03BC F1            1410            POP AF
  03BD D5            1411            PUSH DE                   ;IY := SFX
  03BE FDE1          1412            POP IY
  03C0 1103C6        1413            LD DE,PASS1               ;create "CALL [IY]" with RET to PASS1 by storing
  03C3 D5            1414            PUSH DE                   ;PASS1 on the stack
  03C4 FDE9          1415            JP [IY]                   ;1st 7 bytes SFX will save addr next note & SONGNO
  03C6 1600          1416 PASS1      LD D,0                    ;in same fashion, create a "CALL (IY+7)"
  03C8 1E07          1417            LD E,7                    ;to allow SFX to load initial values
  03CA FD19          1418            ADD IY,DE
  03CC 110461        1419            LD DE,MODB0               ;RET to MODB0
  03CF D5            1420            PUSH DE
  03D0 FDE9          1421            JP [IY]
                     1422 ;          ENDIF
                     1423 ;          - if here, note is type 0 - 3
  03D2 C5            1424 L15       PUSH BC                    ;save header on stack
  03D3 78            1425           LD A,B                     ;A := fresh copy header
  03D4 E603          1426           AND 00000011B              ;mask all but type number
  03D6 FE00          1427           CP 0                       ;test for type 0
                     1428 ;         IF [PSW,IS,ZERO]           ;note is type 0: fixed freq and atn
  03D8 2020          1429           JR NZ,L16
                     1430 ;            --CASE-- note type 0
                     1431 ;            * set up NEXT_NOTE_PTR
  03DA 23            1432 TYPE0      INC HL                    ;next note [after this new note] is 4 bytes away,
  03DB 23            1433            INC HL                    ;point HL to it
  03DC 23            1434            INC HL
  03DD 23            1435            INC HL
  03DE DD7501        1436            LD [IX+1],L               ;put addr in NEXT_NOTE_PTR
  03E1 DD7402        1437            LD [IX+2],H
                     1438 ;          * move new note data and fill in bytes where necessary
  03E4 2B            1439            DEC HL                    ;point HL back to 1st ROM data to move, NLEN
  03E5 110005        1440            LD DE,05                  ;point DE to destination: bytes 5, 4, and 3
  03E8 CD0478        1441            CALL DE_TO_DEST
  03EB 010003        1442            LD BC,3                   ;move 3 bytes
  03EE EDB8          1443            LDDR
  03F0 DD360700      1444            LD [IX+FSTEP],0           ;set for no freq sweep
  [illegible] DD360[illegible] 14[illegible]  [illegible] [IX+[illegible]],0  ;[illegible] no [illegible] sweep
  [illegible] 1867   14[illegible]            [illegible] MODB[illegible]
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  03FA FE01        1448 L16     CP 1                      ;test for type 1
                   1449 ;       IF [PSW,IS,ZERO]          ;note is type 1: swept freq, fixed attenuation
  03FC 201B        1450         JR NZ,L17
                   1451 ;         --CASE-- note type 1
                   1452 ;         * set up NEXT_NOTE_PTR
  03FE 1E06        1453 TYPE1     LD E,6                    ;note after this note is 6 bytes away,
  0400 1600        1454           LD D,0                    ;pt HL to it
  0402 19          1455           ADD HL,DE
  0403 DD7501      1456           LD [IX+1],L               ;store in NEXT_NOTE_PTR
  0406 DD7402      1457           LD [IX+2],H
                   1458 ;         * move new note data and fill in bytes where necessary
  0409 2B          1459           DEC HL                    ;point HL back to 1st ROM data to move, FSTEP
  040A 1C          1460           INC E                     ;E := 7; point DE to destination: bytes 7 - 3
  040B CD0478      1461           CALL DE_TO_DEST
  040E 010005      1462           LD BC,5                   ;move 5 bytes
  0411 EDB8        1463           LDDR
  0413 DD360800    1464           LD [IX+ASTEP],0           ;set for no atn sweep
  0417 1848        1465           JR MODB0
                   1466 ;       ENDIF
  0419 FE02        1467 L17     CP 2                      ;test for type 2
                   1468 ;       IF [PSW,IS,ZERO]          ;note is type 2: fixed freq, swept attenuation
  041B 2028        1469         JR NZ,TYPE3
                   1470 ;         --CASE-- note type 2
                   1471 ;         * set up NEXT_NOTE_PTR
  041D 1E06        1472 TYPE2     LD E,6                    ;pt HL to note after this note; since it's 6 bytes away,
  041F 1600        1473           LD D,0                    ;pt HL to it by adding 6
  0421 19          1474           ADD HL,DE
  0422 F1          1475           POP AF                    ;A := header this note [CH# | SONGNO]
  0423 F5          1476           PUSH AF                   ;put back on stack
  0424 E6C0        1477           AND 11000000B             ;mask SONGNO, leaving CH#
                   1478 ;         IF [PSW,IS,ZERO]          ;this is a noise note, which is only 5 ROM bytes long
  0426 2001        1479           JR NZ,L18
  0428 2B          1480             DEC HL                    ;so move HL back 1 byte
                   1481 ;         ENDIF
  0429 DD7501      1482 L18       LD [IX+1],L               ;put addr in NEXT_NOTE_PTR
  042C DD7402      1483           LD [IX+2],H
                   1484 ;         * move new note data and fill in bytes where necessary
  042F 2B          1485           DEC HL                    ;point HL back to 1st ROM data to move, APS
  0430 1E09        1486           LD E,9                    ;point DE to destination: bytes 9,8,5 - 3
  0432 CD0478      1487           CALL DE_TO_DEST
  0435 010002      1488           LD BC,2                   ;move 2 bytes
  0438 EDB8        1489           LDDR                      ;when done, DE points to FSTEP, HL to ROM NLEN
  043A 3E00        1490           LD A,0
  043C 12          1491           LD [DE],A                 ;FSTEP := 0 for no freq sweep
  043D 1B          1492           DEC DE                    ;pt DE to RAM NLEN
  043E 1B          1493           DEC DE
  043F 0E03        1494           LD C,3                    ;move last 3 ROM bytes; if this is a noise note, garbage
  0441 EDB8        1495           LDDR                      ;will be loaded into byte 3, buts that's OK
  0443 181C        1496           JR MODB0
                   1497 ;       ENDIF
                   1498 ;       if here, note is type 3: swept freq, swept attenuation
                   1499 ;       --CASE-- note type 3
                   1500 ;       * set up NEXT_NOTE_PTR
  0445 1E08        1501 TYPE3   LD E,8                    ;note after this note is 8 bytes away,
  0447 1600        1502         LD D,0                    ;pt HL to it
  0449 19          1503         ADD HL,DE
  044A DD7501      1504         LD [IX+1],L                 ;put addr in NEXT_NOTE_PTR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   0440 DD7402        1505            LD [IX+2],H
                      1506 ;          * move new note data and fill in bytes where necessary
   0450 2B            1507            DEC HL                      ;point HL back to 1st ROM data to move, APS
   0451 DDE5          1508            PUSH IX                     ;point DE to destination: bytes 9 - 3
   0453 FDE1          1509            POP IY                      ;IY := addr byte 0 [and DE = 6]
   0455 1E09          1510            LD E,9                      ;DE := 9
   0457 FD19          1511            ADD IY,DE                   ;IY := addr byte 9 [APS]
   0459 FDE5          1512            PUSH IY
   045B D1            1513            POP DE                      ;DE := addr APS
   045C 010007        1514            LD BC,7                     ;move 7 bytes
   045F EDB8          1515            LDDR
                      1516 ;          * modify byte 0 basis header new note
   0461 DDE5          1517 MODB0      PUSH IX                     ;pt HL to byte 0
   0463 E1            1518            POP HL
   0464 F1            1519            POP AF                      ;A := header new note
   0465 C1            1520            POP BC                      ;B := SONGNO
   0466 FEFF          1521            CP INACTIVE                 ;test for inactive [song over, as detected above]
   0468 C8            1522              RET Z
   0469 57            1523            LD D,A                      ;save header in D
   046A E63F          1524            AND 3FH                     ;Rid channel bits
   046C FE04          1525            CP 04                       ;Special effect
   046E 2002          1526            JR NZ,L20_LOAD_NEX
   0470 063E          1527              LD B,62
   0472               1528 L20_LOAD_NEX:
   0472 7A            1529            LD A,D                      ;restore A to header
   0473 E6C0          1530            AND 0C0H                    ;A := CH# 0 0 0 0 0 0
   0475 B0            1531            OR B                        ;A := new CH# | SONGNO
   0476 77            1532            LD [HL],A                   ;store back in byte 0
                      1533 ;          ENDIF
   0477 C9            1534 L19        RET
   0478               1535 DE_TO_DEST                            ;DE passed = offset from byte 0, RETed w addr byte offset
   0478 DDE5          1536            PUSH IX
   047A FDE1          1537            POP IY                      ;IY := addr byte 0 [and DE = offset]
   047C FD19          1538            ADD IY,DE                   ;IY := addr byte 0 + offset
   047E FDE5          1539            PUSH IY
   0480 D1            1540            POP DE                      ;DE := addr of destination byte in SxDATA
   0481 C9            1541            RET
                      1542 ;          END ;LOADNEX
                      1543  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

     ;        .IDENT  ACTIVATE
     ;        .ZOP
     ;        .EPOP
     ;        .COMMENT )
     ;********************* ACTIVATE *************************************************
     ;
     ;                                                              4/22/82
     ;                                                              13:50:00
     ;
     ;THE FOLLOWING CHANGES/REVISIONS WERE MADE:
     ;
     ;        1. ELIMINATE CODE PLACING OLD.SCREEN ADDRESS IN STATUS AREA
     ;        2. INIT X.PAT.POS IN OLD.SCREEN WHEN IN VRAM AS WELL AS WHEN IN CRAM
     ;        3. USE VDP.MODE.WORD TO TEST GRAPHICS MODE
     ;        4. ADD CODE TO EXPAND ONE COLOR GENERATOR BYTE TO 8
     ;        5. ADDED C_BUFF DEFS 8 FOR COLOR EXPANDING CODE
     ; 5/02   6. FIX COLOR GEN MOVE IN MODE I
     ;        7. USE CONTROLER_MAP FOR BUFFER AREA
     ;
     *  ACTIVATE is used to initialize the RAM status area for the passed
     *object and move its pattern and color generators to the PATTERN and
     *COLOR GENERATOR tables in VRAM_  The second function is enabled or
     *disabled by setting or resetting the carry flag in the PSW_  this is
     *necessary to prevent sending the same graphics data to VRAM more than
     *once when creating identical objects_  The calling sequence for act-
     *ivating an object is as follows:
     *
     *             LD           HL,OBJ_n       ;->OBJ TO ACTIVATE
     *             SCF                         ;SIGNAL MV TO VRAM
     *             CALL         ACTIVATE
     *
     *
     *OR
     *
     *             LD           HL,OBJ_n       ;->OBJ TO ACTIVATE
     *             OR           A              ;DON'T MV TO VRAM
     *             CALL         ACTIVATE
     *
     *)
                  ;EXT          PUT_VRAM_,VRAM_WRITE,VDP_MODE_WORD
                  ;EXT          WORK_BUFFER
     ;
                   GLB          ACTIVATE_
     ;
     ;REGISTER USAGE: FOLLOWING WILL BE CHANGED BY ACTIVATE, ADDITIONAL
     ;MAY BE CHANGED BY CALLED SUBR
     ;             AF,HL,DE,BC,IY
     ;


     ; PROCEDURE ACTIVATEQ(VAR OBJ:OBJECT;MOVE:BOOLEAN);

     ; ACTIVATEQ IS THE PASCAL ENTRY POINT TO ACTIVATE

                  ;EXT     PARAM_
     ; THE PASCAL PARAMETER PASSING PROCEDURE
     ;             COMN
```

```
                    ;PRM_AREA:      DEFS            3          ;Moved to OS
                    ; THIS IS THE COMMON PARAMETER PASSING AREA

                                    PROG
 0482 0002FFFE      ACTIVATE_P:     DEFW            2,-2,1
 0486 0001

                                    GLB             ACTIVATEQ
         <0488>     ACTIVATEQ       EQU             $
 0488 010482                        LD              BC,ACTIVATE_P
 048B 11738A                        LD              DE,PRM_AREA
 048E CD0098                        CALL            PARAM_
 0491 2A738A                        LD              HL,[PRM_AREA]
 0494 5E                            LD              E,[HL]
 0495 23                            INC             HL
 0496 56                            LD              D,[HL]
 0497 EB                            EX              DE,HL
 0498 3A738C                        LD              A,[PRM_AREA+2]
 049B FE00                          CP              0
 049D 2803                          JR              Z,NTZZZ_
 049F 37                            SCF
 04A0 1801                          JR              TZZZ_
 04A2 B7            NTZZZ_:          OR              A
 04A3               TZZZ_:

         <04A3>     ACTIVATE_       EQU             $
                    ;SUP POINTERS ETC_ COMMON TO ALL SUBCASES
                    ;               HL->OBJ DEF CROM
                    ;               C FLG=SUP VRAM FLG
 04A3 5E                            LD              E,[HL]          ;->OBJ GEN CROM
 04A4 23                            INC             HL
 04A5 56                            LD              D,[HL]
 04A6 23                            INC             HL
 04A7 4E                            LD              C,[HL]          ;->OBJ CRAM
 04A8 23                            INC             HL
 04A9 46                            LD              B,[HL]
 04AA 23                            INC             HL
 04AB 3E00                          LD              A,0             ;ZERO FRAME
 04AD 02                            LD              [BC],A
 04AE 1A                            LD              A,[DE]          ;GET OBJ_TYPE
 04AF F5                            PUSH            AF              ;SV OBJ_TYPE & FLG
 04B0 E60F                          AND             0FH             ;GET OBJ_TYPE NUM
 04B2 CA04E7                        JP              Z,ACT_SEMI                ;TYPE=0
 04B5 3D                            DEC             A
 04B6 CA05F1                        JP              Z,ACT_MOBILE              ;TYPE=1
 04B9 3D                            DEC             A
 04BA CA0600                        JP              Z,ACT_OSPRT               ;TYPE=2
 04BD 3D                            DEC             A
 04BE CA0600                        JP              Z,ACT_1SPRT               ;TYPE=3
 04C1 3D                            DEC             A
 04C2 2802                          JR              Z,ACT_CMPLX               ;TYPE=4
 04C4 F1                            POP             AF              ;SUBCASE ELSE
 04C5 C9                            RET
                    ;ON ENTRY TO SUBCASES:
                    ;               STACK=OBJ TYPE & SUP VRAM FLG
                    ;               ->OB
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                  1658 ;                DE->OBJ GRAPHICS+0
                  1659 ;                BC->OBJ STATUS+0
                  1660 ;                A=0
                  1661 ;
          <04C6>  1662 ACT_CMPLX  EQU        $
                  1663 ; SUBCASE Complex
  04C6 1A         1664                  LD        A,[DE]          ;GET COMP_CNT
  04C7 1F         1665                  RRA
  04C8 1F         1666                  RRA
  04C9 1F         1667                  RRA
  04CA 1F         1668                  RRA
  04CB E60F       1669                  AND       0FH
  04CD 47         1670                  LD        B,A             ;SET CNTR
  04CE 5E         1671                  LD        E,[HL]          ;DE->COMP PTRS LIST
  04CF 23         1672                  INC       HL
  04D0 56         1673                  LD        D,[HL]
  04D1 23         1674                  INC       HL
  04D2 B7         1675                  OR        A               ;? EMPTY
  04D3 2810       1676                  JR        Z,CMPLX9
          <04D5>  1677 CMPLX4          EQU       $
  04D5 F1         1678                  POP       AF              ;SUP CALL, COMP OBJ
  04D6 F5         1679                  PUSH      AF
  04D7 E5         1680                  PUSH      HL
  04D8 C5         1681                  PUSH      BC
  04D9 EB         1682                  EX        DE,HL
  04DA CD04A3     1683                  CALL      ACTIVATE_
  04DD C1         1684                  POP       BC              ;RESTORE PNTRS
  04DE E1         1685                  POP       HL
  04DF 5E         1686                  LD        E,[HL]
  04E0 23         1687                  INC       HL
  04E1 56         1688                  LD        D,[HL]
  04E2 23         1689                  INC       HL
  04E3 10F0       1690                  DJNZ      CMPLX4         ;? MORE, RELOOP
  04E5 F1         1691 CMPLX9:          POP       AF              ;CLEAR STACK FOR RTN
  04E6 C9         1692                  RET       ;TECHNICALLY SHOULD JMP TO RTN
                  1693 ;
          <04E7>  1694 ACT_SEMI   EQU        $
                  1695 ; SUBCASE Semi_Moblile
  04E7 CD0572     1696                  CALL      INIT_XP_OS      ;X_PAT_POS := 80H
  04EA 1A         1697                  LD        A,[DE]          ;A := FIRST_GEN_NAME
  04EB 6F         1698                  LD        L,A
  04EC 13         1699                  INC       DE
  04ED 1A         1700                  LD        A,[DE]          ;A := NUMGEN
  04EE 85         1701                  ADD       A,L
  04EF FD7705     1702                  LD        [IY+5],A        ;NEXT_GEN := FIRST_GEN_NAME + NUMGEN
  04F2 2600       1703                  LD        H,0             ;HL=FIRST_GEN_NAME
                  1704 ;AT THIS POINT:
                  1705 ;                 STACK=OBJ_TYPE & SUP VRAM FLG
                  1706 ;                 HL=FIRST_GEN_NAME
                  1707 ;                 DE->NUMGEN
                  1708 ;                 BC:FREE
                  1709 ; SUP FOR VRAM INIT
  04F4 F1         1710                  POP       AF              ;IF SUP VRAM FLG ON
  04F5 3038       1711                  JR        NC,SEMI_EXIT
  04F7 F5         1712                  PUSH      AF
  04F8 3A73C3     1713                  LD        A,[VDP_MODE_WORD] ;SEE WHICH GRAPHICS MODE
  04FB CB4F       1714                  BIT       1,A             ;IF GR II MODE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  04FD 2831          1715                 JR          Z,SEMI_GRI      ;·., GO GRI
  04FF EB            1716                 EX          DE,HL           ;DE=FIRST_GEN_NAME
  0500 44            1717                 LD          B,H             ;SV -> NUMGEN
  0501 4D            1718                 LD          C,L
  0502 6E            1719                 LD          L,[HL]          ;CALC SOURCE OFFSET
  0503 2600          1720                 LD          H,0
  0505 E5            1721                 PUSH        HL
  0506 29            1722                 ADD         HL,HL
  0507 29            1723                 ADD         HL,HL
  0508 29            1724                 ADD         HL,HL
  0509 E5            1725                 PUSH        HL
  050A 03            1726                 INC         BC              ;HL->SOURCE BUFFER
  050B 0A            1727                 LD          A,[BC]
  050C 6F            1728                 LD          L,A
  050D 03            1729                 INC         BC
  050E 0A            1730                 LD          A,[BC]
  050F 67            1731                 LD          H,A
  0510 C1            1732                 POP         BC
  0511 FDE1          1733                 POP         IY
  0513 F1            1734                 POP         AF
                     1735 ;AT THIS POINT:
                     1736 ;               HL->SOURCE BUFFER, PTRN GNRTRS
                     1737 ;               DE=INDEX TO START OF VRAM ENTRIES
                     1738 ;               IY=NUMBER OF ITEMS TO READ FROM VRAM
                     1739 ;               BC=OFFSET TO COLOR SOURCE BUFFER @
                     1740 ;               AF=OBJ_TYPE [& SUP VRAM FLG, UNNEEDED]
                     1741 ; FILL AS NEEDED TOP, MID, AND BOT PTRN_GNRTRS & DITTO FOR COLOR_GNRTRS
  0514 CB7F          1742                 BIT         7,A             ;IF BIT 7 OBJ_TYPE ON (TOP)
  0516 2803          1743                 JR          Z,SEMI_MID      ;·., GO HNDL MID
  0518 CD0594        1744                 CALL        SUP_GEN_CLR
          <051B>     1745 SEMI_MID        EQU         $
  051B CD05E8        1746                 CALL        SUP_UPDATE
  051E CB77          1747                 BIT         6,A             ;IF BIT 6 OBJ_TYPE ON (MID)
  0520 2803          1748                 JR          Z,SEMI_BOT
  0522 CD0594        1749                 CALL        SUP_GEN_CLR
          <0525>     1750 SEMI_BOT        EQU         $
  0525 CD05E8        1751                 CALL        SUP_UPDATE
  0528 CB6F          1752                 BIT         5,A             ;IF BIT 5 OBJ_TYPE ON (BOT)
  052A 2803          1753                 JR          Z,SEMI_EXIT
  052C CD0594        1754                 CALL        SUP_GEN_CLR
          <052F>     1755 SEMI_EXIT       EQU         $
  052F C9            1756                 RET
                     1757 ;
                     1758 ; Handle GRAPHICS MODE I
          <0530>     1759 SEMI_GRI        EQU         $
  0530 EB            1760                 EX          DE,HL           ;HL->NUMGEN
  0531 4E            1761                 LD          C,[HL]          ;IY=NUMGEN
  0532 0600          1762                 LD          B,0
  0534 C5            1763                 PUSH        BC
  0535 FDE1          1764                 POP         IY
  0537 23            1765                 INC         HL              ;HL->PTRN_GNRTRS
  0538 7E            1766                 LD          A,[HL]
  0539 23            1767                 INC         HL
  053A 66            1768                 LD          H,[HL]
  053B 6F            1769                 LD          L,A
  053C E5            1770                 PUSH        HL              ;SAVE FOR RESTORE
  053D C5            1771                 PUSH        BC
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  053E D5            1772                   PUSH          DE
  053F FDE5          1773                   PUSH          IY
  0541 3E03          1774                   LD            A,3             ;SIGNAL PTRN GEN FILL
  0543 CD1C27        1775                   CALL          PUT_VRAM_
  0546 C1            1776                   POP BC  ;BC := NUMGEN
  0547 E1            1777                   POP HL  ;HL := FIRST_GEN_NAME
  0548 50            1778                   LD E,L
  0549 54            1779                   LD D,H  ;DE := FIRST_GEN_NAME
  054A 09            1780                   ADD HL,BC ;HL := FIRST_GEN_NAME + NUMGEN
  054B 2B            1781                   DEC HL
  054C CB3C          1782                   SRL H
  054E CB1D          1783                   RR L
  0550 CB3C          1784                   SRL H
  0552 CB1D          1785                   RR L
  0554 CB3C          1786                   SRL H
  0556 CB1D          1787                   RR L    ;HL := (FIRST_GEN_NAME + NUMGEN - 1)/8
  0558 CB2B          1788                   SRA E
  055A CB2B          1789                   SRA E
  055C CB2B          1790                   SRA E   ;DE := FIRST_GEN_NAME/8
  055E B7            1791                   OR A    ;CLEAR CARRY
  055F ED52          1792                   SBC HL,DE
  0561 23            1793                   INC HL  ;HL := (F_G_N + NMGN - 1)/8 - F_G_N/8 + 1 = NUMBER COLR GENS
  0562 E5            1794                   PUSH HL
  0563 FDE1          1795                   POP IY
  0565 E1            1796                   POP           HL              ;RESTORE REG
  0566 29            1797                   ADD           HL,HL           ;STEP OVER PTRN_GNRTRS
  0567 29            1798                   ADD           HL,HL
  0568 29            1799                   ADD           HL,HL
  0569 C1            1800                   POP           BC
  056A 09            1801                   ADD           HL,BC           ;HL->COLOR GNRTR SOURCE
  056B 3E04          1802                   LD            A,4             ;SIGNAL PTRN COLOR TBL
  056D CD1C27        1803                   CALL          PUT_VRAM_
  0570 F1            1804                   POP           AF              ;FIX STACK
  0571 C9            1805                   RET
                     1806 ; Internal routine to initialize X_Pat_Pos in Old_Screen
  0572 C5            1807 INIT_XP_OS:       PUSH          BC
  0573 FDE1          1808                   POP           IY              ;IY -> STATUS
  0575 D5            1809                   PUSH          DE              ;SAVE -> GRAPHICS
  0576 5E            1810                   LD            E,[HL]          ;DE := OLD_SCREEN ADDRESS
  0577 23            1811                   INC           HL
  0578 56            1812                   LD            D,[HL]
  0579 CB7A          1813                   BIT           7,D             ;? OLD SCRN IN CROM
  057B 2014          1814                   JR            NZ,SM_BY_OLD
  057D 7A            1815                   LD            A,D             ;OLD_SCREEN IN VRAM?
  057E FE70          1816                   CP            70H
  0580 3806          1817                   JR            C,OS_IN_VRAM
  0582 3E80          1818                   LD            A,80H           ;INIT X_PAT_POS = 80H
  0584 12            1819                   LD            [DE],A
  0585 180A          1820                   JR            SM_BY_OLD
  0587 80            1821 INIT_80:          DEFB          80H
  0588 210587        1822 OS_IN_VRAM:       LD            HL,INIT_80
  058B 010001        1823                   LD            BC,1            ;ONE BYTE TO MOVE TO VRAM
  058E CD1D01        1824                   CALL          VRAM_WRITE
           <0591>    1825 SM_BY_OLD       EQU           $
  0591 D1            1826                   POP           DE              ;DE -> GRAPHICS
  0592 13            1827                   INC           DE              ;DE -> FIRST_GEN_NAME
  0593 C9            1828                   RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1829 ;
                     1830 ;Internal rout to setup Ptrn Gen VRAM & Color Gen VRAM
           <0594>    1831 SUP_GEN_CLR    EQU               $
  0594 F5            1832                     PUSH          AF              ;SAVE FOR RESTORE
  0595 C5            1833                     PUSH          BC
  0596 FDE5          1834                     PUSH          IY
  0598 D5            1835                     PUSH          DE
  0599 E5            1836                     PUSH          HL
  059A 3E03          1837                     LD            A,3             ;SIGNAL PTRN GEN FILL
  059C CD1C27        1838                     CALL          PUT_VRAM_
  059F E1            1839                     POP           HL              ;RESTORE
  05A0 D1            1840                     POP           DE
  05A1 FDE1          1841                     POP           IY
  05A3 C1            1842                     POP           BC
  05A4 F1            1843                     POP           AF
  05A5 F5            1844                     PUSH          AF
  05A6 C5            1845                     PUSH          BC              ;SAVE FOR RESTORE
  05A7 FDE5          1846                     PUSH          IY
  05A9 D5            1847                     PUSH          DE
  05AA E5            1848                     PUSH          HL
  05AB CB67          1849                     BIT           4,A             ;HOW MANY COLOR GEN BYTES?
  05AD 2000          1850                     JR            NZ,ONE_BYTE
  05AF 09            1851                     ADD           HL,BC           ;HL->COLOR GEN SOURCE
  05B0 3E04          1852                     LD            A,4             ;SIGNAL PTRN COLOR FILL
  05B2 CD1C27        1853                     CALL          PUT_VRAM_
  05B5 E1            1854 O_B_RET:           POP           HL
  05B6 D1            1855                     POP           DE
  05B7 FDE1          1856                     POP           IY
  05B9 C1            1857                     POP           BC
  05BA F1            1858                     POP           AF
  05BB C9            1859                     RET
                     1860 ; For each item to send, duplicate the color byte 8 times [in C_BUFF]
                     1861 ; then send this generator to VRAM color table indexed by DE
  05BC               1862 ONE_BYTE:
  05BC 09            1863           ADD HL,BC               ;HL -> COLOR BYTE
  05BD 4D            1864           LD C,L
  05BE 44            1865           LD B,H                  ;BC -> COLOR BYTE
  05BF FDE5          1866           PUSH IY
  05C1 E1            1867           POP HL                  ;HL = ITEM COUNT
  05C2               1868 NEXT_COLOR:
  05C2 E5            1869           PUSH HL                 ;SAVE COUNTER
  05C3 0A            1870           LD A,[BC]               ;GET COLOR BYTE
  05C4 C5            1871           PUSH BC                 ;SAVE POINTER TO COLOR
  05C5 010008        1872           LD BC,8                 ;CREATE 8 DUPLICATES
  05C8 2A8006        1873           LD HL,[WORK_BUFFER]
  05CB 09            1874           ADD HL,BC               ;PLACE THEM HERE, STARTING AT END OF BUFFER
  05CC 0608          1875           LD B,8
  05CE 2B            1876 DUPLI:   DEC HL
  05CF 77            1877           LD [HL],A
  05D0 10FC          1878           DJNZ DUPLI
  05D2 D5            1879           PUSH DE                 ;SAVE INDEX INTO TABLES
  05D3 FD210001      1880           LD IY,1                 ;1 ITEM TO SEND
  05D7 3E04          1881           LD A,4                  ;COLOR TABLE CODE
  05D9 CD1C27        1882           CALL PUT_VRAM_
  05DC D1            1883           POP DE                  ;GET INDEX BACK
  05DD C1            1884           POP BC                  ;POINTER TO COLOR BYTE
  05DE 13            1885           INC DE                  ;INCREMENT INDEX
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   05DF 03           1886         INC BC                    ;INCREMENT COLOR POINTER
   05E0 E1           1887         POP HL                    ;GET ITEM COUNTER
   05E1 2B           1888         DEC HL
   05E2 7C           1889         LD A,H
   05E3 B5           1890         OR L
   05E4 200C         1891         JR NZ,NEXT_COLOR
   05E6 18CD         1892         JR O_B_RET
                     1893 ;Internal rout to update to next VRAM index screen area
          <05E8>     1894 SUP_UPDATE     EQU           $
   05E8 C5           1895                PUSH          BC
   05E9 010100       1896                LD            BC,100H
   05EC EB           1897                EX            DE,HL
   05ED 09           1898                ADD           HL,BC
   05EE EB           1899                EX            DE,HL
   05EF C1           1900                POP           BC
   05F0 C9           1901                RET
                     1902 ;
          <05F1>     1903 ACT_MOBILE     EQU           $
                     1904 ; SUBCASE Mobile
   05F1 CD0572       1905                CALL          INIT_XP_OS      ;X_PAT_POS := 80H
                     1906 ; INSERT NEW_GENERATOR ADDRESS IN OBJECT CRAM
   05F4 13           1907                INC           DE
   05F5 1A           1908                LD            A,[DE]
   05F6 FD7705       1909                LD            [IY+5],A
   05F9 13           1910                INC           DE
   05FA 1A           1911                LD            A,[DE]
   05FB FD7706       1912                LD            [IY+6],A        ;INIT NEW_GEN IN STATUS
   05FE F1           1913                POP           AF
   05FF C9           1914                RET
          <0600>     1915 ACT_OSPRT  EQU          $
                     1916 ; SUBCASE Sprite size 0
          <0600>     1917 ACT_1SPRT  EQU          $
                     1918 ; SUBCASE Sprite size 1
   0600 03           1919                INC           BC              ;->NEXT_GEN IN CRAM
   0601 03           1920                INC           BC
   0602 03           1921                INC           BC
   0603 03           1922                INC           BC
   0604 03           1923                INC           BC
   0605 EB           1924                EX            DE,HL           ;HL->FIRST_GEN_NAME
   0606 23           1925                INC           HL
   0607 7E           1926                LD            A,[HL]
   0608 5F           1927                LD            E,A             ;SV INDEX TO VRAM
   0609 1600         1928                LD            D,0
   060B D5           1929                PUSH          DE
   060C 23           1930                INC           HL              ;DE=PTRN_PTR
   060D 5E           1931                LD            E,[HL]
   060E 23           1932                INC           HL
   060F 56           1933                LD            D,[HL]
   0610 23           1934                INC           HL              ;CALC & SET NEXT_GEN CRAM
   0611 86           1935                ADD           A,[HL]
   0612 02           1936                LD            [BC],A
   0613 4E           1937                LD            C,[HL]
   0614 0600         1938                LD            B,0
   0616 C5           1939                PUSH          BC
   0617 FDE1         1940                POP           IY
   0619 EB           1941                EX            DE,HL           ;HL->SOURCE PTRN GEN
   061A D1           1942                POP           DE              ;DE=INDEX TO PTRN GEN VRAM
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   061B F1          1943                POP        AF
   061C D0          1944                RET        NC
   061D 3E01        1945                LD         A,1           ;SIGNAL SPRITE PRTN GEN FILL
   061F CD1C27      1946                CALL       PUT_VRAM_
   0622 C9          1947                RET
                    1948
                    1949  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1951
                     1952 ********************  PUTOBJ  *********************************************
                     1953 ;DESCRIPTION:     PUTOBJ VECTORS TO ONE OF 5 SPECIFIC ROUTINES FOR PLACING THE
                     1954 ;                 DIFFERENT OBJECT TYPES ON THE DISPLAY
                     1955 ;INPUT:            IX = ADDRESS OF OBJECT TO BE PROCESSED
                     1956 ;                  B = PARAMETER TO BE PASSED SPECIFIC PUT ROUTINES
                     1957
                     1958 * IN ADDITION, THIS MODULE CONTAINS ROUTINES WHICH ALLOW VRAM OPERATIONS
                     1959 * TO BE DEFERRED, TYPICALLY UNTIL AN INTERRUPT OCCURS, AND PERFORMED
                     1960 * IN A BLOCK BY A CENTRAL WRITER ROUTINE.
                     1961 *******************************************************************************
                     1962
                     1963                  DATA
  73CA               1964 QUEUE_SIZE      DEFS           1
                     1965 * THIS IS THE SIZE OF THE DEFERRED WRITE QUEUE. IT IS SET BY THE
                     1966 * CARTRIDGE PROGRAMMER. IT HAS RANGE 0 - 255.
                     1967
  73CB               1968 QUEUE_HEAD      DEFS           1
  73CC               1969 QUEUE_TAIL      DEFS           1
                     1970 * THESE ARE THE INDICES OF THE HEAD AND TAIL OF THE WRITE QUEUE.
                     1971
  73CD               1972 HEAD_ADDRESS    DEFS           2
  73CF               1973 TAIL_ADDRESS    DEFS           2
                     1974 * THESE ARE THE ADDRESSES OF THE QUEUE HEAD AND TAIL
                     1975
                     1976 ;TRUE            EQU            1
                     1977 ;FALSE           EQU            0
                     1978 * VALUES FOR BOOLEAN DEFERAL_FLAG
                     1979
  73D1               1980 BUFFER          DEFS           2
                     1981 * THIS IS A POINTER TO THE BEGINNING OF THE DEFERRED WRITE QUEUE. THE
                     1982 * CARTRIDGE PROGRAMMER IS RESPONSIBLE FOR PROVIDING A RAM AREA TO HOLD
                     1983 * THE QUEUE, AND PASSING ITS LOCATION AND SIZE TO INIT_QUEUE.
                     1984
                     1985 ;                COMN
                     1986 ;PARAM_AREA      DEFS           3
                     1987 * PARAM_AREA IS THE COMMON PARAMETER PASSING AREA FOR PASCAL ENTRY PTS
                     1988
                     1989
                     1990                 PROG
          <0623>     1991 SET_UP_WRITE    EQU            $
                     1992
                     1993 * SET_UP_WRITE SETS UP A DEFERRED VRAM OPERATION.
                     1994
                     1995 *   PUT DATA AT QUEUE_HEAD
  0623 DDE5          1996                 PUSH           IX
  0625 2A73CD        1997                 LD             HL,[HEAD_ADDRESS]
  0628 D1            1998                 POP            DE
  0629 73            1999                 LD             [HL],E          ; PUT DATA POINTER
  062A 23            2000                 INC            HL
  062B 72            2001                 LD             [HL],D
  062C 23            2002                 INC            HL
  062D 70            2003                 LD             [HL],B          ;STORE PUTOBJ PARAMETER
  062E 23            2004                 INC            HL
  062F EB            2005                 EX             DE,HL           ; HEAD ADDRESS IN DE
                     2006
                     2007 *   INCREMENT QUEUE_HEAD
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  0630 3A73CB       2008                 LD          A,[QUEUE_HEAD]
  0633 3C           2009                 INC         A                  ; NEW HEAD IN A
                    2010
                    2011 *   IF QUEUE_HEAD = QUEUE_SIZE THEN
  0634 2173CA       2012                 LD          HL,QUEUE_SIZE
  0637 BE           2013                 CP          [HL]
  0638 2000         2014                 JR          NZ,NOT_TOO_BIG
                    2015
                    2016 *      QUEUE_HEAD := 0
  063A 3E00         2017                 LD          A,0
  063C 3273CB       2018                 LD          [QUEUE_HEAD],A
                    2019
                    2020 *      HEAD_ADDRESS := BUFFER
  063F 2A7301       2021                 LD          HL,[BUFFER]
  0642 2273CD       2022                 LD          [HEAD_ADDRESS],HL
                    2023
  0645 1807         2024                 JR          SET_UP_ENDIF
                    2025 *   ELSE
        <0647>      2026 NOT_TOO_BIG     EQU         $
                    2027
                    2028 *      STORE NEW QUEUE_HEAD
  0647 3273CB       2029                 LD          [QUEUE_HEAD],A
                    2030
                    2031 *      STORE HEAD_ADDRESS
  064A ED5373CD     2032                 LD          [HEAD_ADDRESS],DE
                    2033
                    2034 *   END IF
  064E              2035 SET_UP_ENDIF
                    2036
                    2037 * END SET_UP_WRITE
  064E C9           2038                 RET
                    2039
                    2040 * PROCEDURE  INIT_QUEUE (SIZE:BYTE;VAR A_QUEUE:QUEUE)
                    2041
                    2042 * SIZE PASSED IN A, LOCATION PASSED IN HL
                    2043 * DESTROYS: A
                    2044
  064F 00020001     2045 INIT_QUEUE_P    DEFW        2,1,-2
  0653 FFFE
                    2046 * THIS IS THE PARAMETER DESCRIPTOR FOR INIT_QUEUEQ
                    2047
                    2048 * BEGIN INIT_QUEUE
                    2049                 GLB         INIT_QUEUEQ
        <0655>      2050 INIT_QUEUEQ     EQU         $
  0655 01064F       2051                 LD          BC,INIT_QUEUE_P
  0658 1173BA       2052                 LD          DE,PARAM_AREA
  065B CD0098       2053                 CALL        PARAM_
  065E 3A73BA       2054                 LD          A,[PARAM_AREA]
  0661 2A73BB       2055                 LD          HL,[PARAM_AREA+1]
                    2056
                    2057                 GLB         INIT_QUEUE
        <0664>      2058 INIT_QUEUE      EQU         $
                    2059
                    2060 *   QUEUE_SIZE := SIZE
  0664 3273CA       2061                 LD          [QUEUE_SIZE],A
                    2062
                    2063 *   QUEUE_HEAD := QUEUE_TAIL := 0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  0667 3E00          2064                   LD          A,0
  0669 3273CB        2065                   LD          [QUEUE_HEAD],A
  066C 3273CC        2066                   LD          [QUEUE_TAIL],A
                     2067
                     2068 *   BUFFER := TAIL_ADDRESS := HEAD_ADDRESS := LOCATION
  066F 227301        2069                   LD          [BUFFER],HL
  0672 2273CD        2070                   LD          [HEAD_ADDRESS],HL
  0675 2273CF        2071                   LD          [TAIL_ADDRESS],HL
                     2072
                     2073 * END INIT_QUEUE
  0678 C9            2074                   RET
                     2075
                     2076 * PROCEDURE  WRITER_
                     2077
                     2078 * TAKES NO PARAMETERS
                     2079 * DESTROYS: ALL
                     2080
                     2081 * BEGIN WRITER_
                     2082                   GLB         WRITER_
          <0679>     2083 WRITER_            EQU         $
                     2084
                     2085 *   SAVE DEFERAL FLAG
  0679 3A73C6        2086                   LD          A,[DEFER_WRITES]
  067C F5            2087                   PUSH        AF
                     2088
                     2089 *   DEFER_WRITES := FALSE
  067D 3E00          2090                   LD          A,FALSE
  067F 3273C6        2091                   LD          [DEFER_WRITES],A
                     2092
                     2093 *   WHILE QUEUE_TAIL <> QUEUE_HEAD DO
          <0682>     2094 WRTR_WHILE        EQU         $
  0682 3A73CC        2095                   LD          A,[QUEUE_TAIL]
  0685 2173CB        2096                   LD          HL,QUEUE_HEAD
  0688 BE            2097                   CP          [HL]
  0689 2831          2098                   JR          Z,WRTR_END_WHILE
                     2099
                     2100 *     WRITE DATA AT QUEUE_TAIL TO VRAM
  068B 2A73CF        2101                   LD          HL,[TAIL_ADDRESS]
  068E 5E            2102                   LD          E,[HL]          ; GET OBJECT POINTER
  068F 23            2103                   INC         HL
  0690 56            2104                   LD          D,[HL]
  0691 23            2105                   INC         HL
  0692 46            2106                   LD          B,[HL]          ; GET PARAMETER
  0693 23            2107                   INC         HL
                     2108
                     2109 *     PROCESS OBJECT IN QUEUE
  0694 D5            2110                   PUSH        DE
  0695 DDE1          2111                   POP         IX
  0697 E5            2112                   PUSH        HL              ;SAVE QUEUE TAIL ADDRESS
  0698 CD06E3        2113                   CALL        DO_PUTOBJ
                     2114
                     2115 *     INCREMENT QUEUE_TAIL
  069B 3A73CC        2116                   LD          A,[QUEUE_TAIL]
  069E 3C            2117                   INC         A
                     2118
                     2119 *     IF QUEUE_TAIL = QUEUE_SIZE THEN
  069F 2173CA        2120                   LD          HL,QUEUE_SIZE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  06A2 BE           2121                CP          [HL]
  06A3 200E         2122                JR          NZ,WRTR_ELSE
                    2123
                    2124 *        QUEUE_TAIL := 0
  06A5 3E00         2125                LD          A,0
  06A7 3273CC       2126                LD          [QUEUE_TAIL],A
                    2127
                    2128 *        TAIL_ADDRESS := BUFFER
  06AA 2A7301       2129                LD          HL,[BUFFER]
  06AD 2273CF       2130                LD          [TAIL_ADDRESS],HL
  06B0 E1           2131                POP         HL              ;RESTORE STACK POINTER
                    2132
  06B1 1807         2133                JR          WRTR_END_IF
                    2134 *      ELSE
         <06B3>     2135 WRTR_ELSE      EQU         $
                    2136
                    2137 *        STORE NEW QUEUE_TAIL
  06B3 3273CC       2138                LD          [QUEUE_TAIL],A
                    2139
                    2140 *        TAIL_ADDRESS := TAIL_ADDRESS + 3
  06B6 E1           2141                POP         HL
  06B7 2273CF       2142                LD          [TAIL_ADDRESS],HL
                    2143
                    2144 *      END_IF
         <06BA>     2145 WRTR_END_IF    EQU         $
                    2146
  06BA 18C6         2147                JR          WRTR_WHILE
                    2148 *    END_WHILE
         <06BC>     2149 WRTR_END_WHILE EQU         $
                    2150
                    2151 * RESTORE DEFERAL FLAG
  06BC F1           2152                POP         AF
  06BD 3273C6       2153                LD          [DEFER_WRITES],A
                    2154
                    2155 * END WRITER_
  06C0 C9           2156                RET
                    2157
                    2158                GLB         PUTOBJ_
                    2159
                    2160
                    2161         ;EXT PUTSEMI,PUT_MOBILE,PUTOSPRITE,PUT1SPRITE,PUTCOMPLEX
                    2162
                    2163         ;EXT  DEFER_WRITES
                    2164         ;EXT  PARAM_
                    2165          GLB  PUTOBJQ
  06C1 00020002     2166 PUTOBJ_PAR:    DEFW  2,2,1
  06C5 0001
                    2167
                    2168 * PROCEDURE PUT_OBJP (VAR DATA:BUFFER;PARAM:BYTE);
                    2169
                    2170 * THIS IS THE PASCAL ENTRY POINT TO THE PUTOBJ ROUTINE
                    2171
                    2172          PROG
  06C7              2173 PUTOBJQ:
  06C7 0106C1       2174          LD   BC,PUTOBJ PAR
  06CA 11738A       2175          LD   DE,PARAM AREA
  06Cl       9B      176          CAl      AM
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   06D0 DD2A73BA    2177           LD    IX,[PARAM_AREA]
   06D4 3A73BC      2178           LD    A,[PARAM_AREA+2]
   06D7 47          2179           LD    B,A
                    2180
          <0001>    2181 DEFER                EQU              1
   06D8             2182 PUTOBJ_
   06D8 3A73C6      2183           LD       A,[DEFER_WRITES]  ;CHECK IF DEFERRED WRITE IS DESIRED
   06DB FE01        2184           CP       DEFER
   06DD 2004        2185           JR       NZ,DO_PUTOBJ     ;IF NOT, PROCESS OBJECT
   06DF CD0623      2186           CALL     SET_UP_WRITE     ;IF SO, SET UP FOR DEFERRED WRITE
   06E2 C9          2187           RET
   06E3 DD6601      2188 DO_PUTOBJ LD H,[IX+1]               ;GET ADDRESS OF GRAPHICS FOR OBJ_n
   06E6 DD6E00      2189           LD L,[IX+0]
   06E9 7E          2190           LD A,[HL]                ;A := OBJ_TYPE
   06EA 4F          2191           LD C,A                   ;SAVE COPY
   06EB E60F        2192           AND 0FH                  ;MASK FOR OBJ_TYPE NUMBER
   06ED CA06FF      2193           JP Z,PUTSEMI             ;0 = SEMI_MOBILE
   06F0 3D          2194           DEC A
   06F1 CA0A87      2195           JP Z,PUT_MOBILE           ;1 = MOBILE
   06F4 3D          2196           DEC A
   06F5 CA080F      2197           JP Z,PUT0SPRITE          ;2 = SPRITE0
   06F8 3D          2198           DEC A
   06F9 CA0955      2199           JP Z,PUT1SPRITE          ;3 = SPRITE1
   06FC C30EA2      2200           JP PUTCOMPLEX            ;>3 = COMPLEX
                    2201 ;
                    2202 ;          END      ;pname
                    2203  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2205
                     2206 ********************* PUT_SEMI ***********************************************
                     2207 ;DESCRIPTION:     PUTS SEMI_MOBILE OBJECTS ON SCREEN
                     2208 ;
                     2209 ;INPUT:             IX = ADDRESS OF OBJECT TO BE PROCESSED
                     2210 ;                   HL = ADDRESS OF OBJECT'S GRAPHICS TABLES IN ROM
                     2211 ****************************************************************************
                     2212
                     2213                GLB              PUTSEMI
                     2214
                     2215
  06FF DD5603        2216 PUTSEMI:  LD D,[IX+3]             ;GET ADDRESS OF STATUS
  0702 DD5E02        2217           LD E,[IX+2]
  0705 D5            2218           PUSH DE                 ;                     AND PUT INTO IY
  0706 FDE1          2219           POP IY
  0708 FD5602        2220           LD D,[IY+2]             ;GET X_LOCATION
  070B FD5E01        2221           LD E,[IY+1]
  070E CD07E8        2222           CALL     PX_TO_PTRN_POS
                     2223
  0711 4B            2224           LD C,E                  ;C := PATTERN PLANE COL.
  0712 FD5604        2225           LD D,[IY+4]             ;GET Y_LOCATION
  0715 FD5E03        2226           LD E,[IY+3]
  0718 CD07E8        2227           CALL     PX_TO_PTRN_POS
                     2228
  071B 43            2229           LD B,E                  ;B := PATTERN PLANE ROW
  071C FD5E00        2230           LD E,[IY+0]             ;GET FRAME NUMBER
                     2231 ;
                     2232 ; HL = GRAPHICS_n, IX = OBJ_n, IY = STATUS_n, C = COL., B = ROW, E = FRAME
                     2233 ;
  071F 1600          2234           LD D,0                  ;DE HAS FRAME NUMBER
  0721 19            2235           ADD HL,DE
  0722 19            2236           ADD HL,DE               ;2*FRAME NUMBER + ADDR OF GRAPHICS_n
  0723 1E05          2237           LD E,5                  ;FRAME POINTER OFFSET
  0725 19            2238           ADD HL,DE               ;HL NOW POINTS TO LOCATION HOLDING ADDRESS
                     2239                                   ;OF FRAME
  0726 5E            2240           LD E,[HL]               ;GET ADDRESS INTO DE
  0727 23            2241           INC HL
  0728 56            2242           LD D,[HL]
  0729 EB            2243           EX DE,HL                ;HL := ADDRESS OF FRAME
  072A C5            2244           PUSH BC
  072B D1            2245           POP DE                  ;DE := Y_PAT_POS & X_PAT_POS
  072C 4E            2246           LD C,[HL]               ;C := X_EXTENT
  072D 23            2247           INC HL
  072E 46            2248           LD B,[HL]               ;B := Y_EXTENT
  072F 23            2249           INC HL                  ;HL POINTS TO FIRST NAME IN LIST
                     2250 ;
                     2251 ; TEST TO SEE IF OLD_SCREEN IS TO BE SAVED
                     2252 ;
  0730 DD7E05        2253           LD A,[IX+5]             ;GET HIGH BYTE OF OLD_SCREEN ADDRESS
  0733 CB7F          2254           BIT 7,A                 ;TEST BIT 15 OF OLD_SCREEN ADDRESS
  0735 2804          2255           JR Z,S_OLD_SCRN
                     2256 ;
  0737 CD080B        2257           CALL PUTFRAME
  073A C9            2258           RET
                     2259
  073B               2260 S_OLD_SCRN
```