COLECO

ADAM TECHNICAL MANUAL

IV - EOS 6 ABSOLUTE LISTINGS
V - OS 7 ABSOLUTE LISTINGS

SECTION IV

EOS 6 ABSOLUTE LISTINGS

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                       1 ^Z80^
                       3  NAME ^Rev 06.1 - TF^
                       4
                       5 ; Project:  EOS 6 Absolute Listing
                       6 ;
                       7 ;     ************************************************************
                       8 ;     ****                                                    ****
                       9 ;     ****              EOS ABSOLUTE LISTING                  ****
                      10 ;     ****                                                    ****
                      11 ;     ****           Electronic Development Group             ****
                      12 ;     ****              Coleco Industries Inc.                ****
                      13 ;     ****              999 Quaker Lane South                 ****
                      14 ;     ****              West Hartford, Connecticut            ****
                      15 ;     ****                                   06110            ****
                      16 ;     ****                                                    ****
                      17 ;     ************************************************************
                      18 ;
                      19 ;     This absolute listing was generated to ease software
                      20 ;     development on ADAM.   This listing provides the location
                      21 ;     of both released and unreleased entry points.  Released
                      22 ;     entry points begin immediately in this file with the jump
                      23 ;     table and end before the first code segment listed.
                      24 ;     Released entry points include the jump table, common data
                      25 ;     areas(EOS_COMN), common data tables, and equates which
                      26 ;     describe the released data structures.  Direct access
                      27 ;     to code segments is STRONGLY DISCOURAGED and may make
                      28 ;     your application incompatable with some ADAMs.  There is
                      29 ;     more than one version of EOS on the market at this time
                      30 ;     and updates are planned.
                      31 ;
                      32 ;     For further information on ADAM or EOS consult the
                      33 ;     ADAM Technical Reference Manual or send your questions
                      34 ;     in writing to:
                      35 ;
                      36 ;                    Coleco Software Support
                      37 ;
                      38 ;     at the above address.
                      39 ;
                      40 ; COLECO MAKES NO REPRESENTATIONS OR WARRANTIES WHATSOEVER, INCLUDING
                      41 ; WITHOUT LIMITATION ANY IMPLIED WARRANTIES OR MERCHANTABILITY AND
                      42 ; FITNESS FOR A PARTICULAR PURPOSE, IN CONNECTION WITH THE MATERIALS
                      43 ; CONTAINED HEREIN, AND SUCH MATERIALS ARE DISCLOSED AS IS.  COLECO
                      44 ; SHALL HAVE NO LIABILITY FOR ANY LOSSES CAUSED TO RECIPIENTS OF THESE
                      45 ; MATERIALS BY REASON OF ANY CHANGES OR MODIFICATIONS MADE BY COLECO IN
                      46 ; THESE MATERIALS AFTER THEIR DISCLOSURE HEREIN.  IN ADDITION, COLECO
                      47 ; SHALL HAVE NO LIABILITY FOR ANY CONSEQUENTIAL, SPECIAL, INDIRECT OR
                      48 ; INCIDENTAL DAMAGES OR LOSSES WHATSOEVER, INCLUDING LOSS OF PROFITS, IN
                      49 ; CONNECTION WITH THE USE OF THE MATERIALS DISCLOSED HEREIN.
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      51 ;*****************************************************************
                      52 ;
                      53 ;       Rev History
                      54 ;       Rev.  Date          Name      Change
                      55 ;        6.1  21jun0844     af        Merge sources for absolute listing.
                      56 ;                                     No changes made to code.
                      57 ;        6.0  08oct1545     jkl       Code released for production.
                      58 ;
                      59 ;[    code commented out by ¢;eee' was removed before EOS  ]
                      60 ;[ was grouped into one file, code commented out by simple ]
                      61 ;[ ¢;' was removed in the process of grouping EOS in order ]
                      62 ;[ to avoid assembler complaints or LOCAL/GLOBAL conflicts ]
                      63 ;[    ¢;***********...' separates code segments which were ]
                      64 ;[ originally in separate files                            ]
                      65 ;*****************************************************************
                      66 ;*****************************************************************
                      67 ;*****************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    69  NAME ^Rev 11 - RPD^
                    70 ;
                    71 ; Rev  Date          Made by    Change
                    72 ; 11   15oct425p     RPD        changed EOS_4 to CV_A
                    73 ; 10   13oct1000p    RPD        made only OS7 jump entries conditionally assembled
                    74 ;  9   12oct2300     vsb        Added __SCAN_FOR_FILE,__POSIT_FILE,__FILE_QUERY plus __EOS1 thru __EOS4
                    75 ;  8   07oct1710     rfj        __EFXOVER changed to __EFFECT_OVER
                    76 ;                               Added __UPDATE_SPINNER
                    77 ;  7   05oct1725     rfj        Renamed WR_SPR_NM_TBL to WR_SPR_ATTRIBUTE
                    78 ;  6   03oct1207     rfj        Increased TEMP_STACK to 12 bytes,
                    79 ;                                Removed SAVED_COUNT
                    80 ;  5   30sept825p    RPD        replaced CONS_OUT data area with new CONS_OUT2
                    81 ;  4   29sept235p    RPD        replaced PLAY_SONGS with SOUNDS
                    82 ;  3   26sept1005a   RPD        added conditional assembly stuff for super games
                    83 ;  2   25sept850a    RPD        merged all of EOS ram together into this file
                    84 ;  1   20sept1005a   RPD        new memory mapping
                    85 ;  0   mid-sept      VB         original map @ E800H
                    86 ;
<FFFF>              87 TRUE            EQU     OFFFFH
<0000>              88 FALSE           EQU     00000H
                    89
<FFFF>              90 IN_EOS          EQU     TRUE
<0000>              91 SUPERGAME       EQU     FALSE
                    92 ;
                    93 ; CURRENT EOS MEMORY MAPPING
                    94 ;
                    95 ;FCB_S          EQU     0D390H                  ;FCB area
                    96 ;THREE1K_BLKS   EQU     0D400H                  ;3 FCB 1K buffers
<E000>              97 EOS_CODE        EQU     0E000H                  ;address of EOS code segment
<FBFF>              98 EOS_GLB_TBL     EQU     0FBFFH                  ;address of EOS global tables
<FC30>              99 EOS_JMP_TBL     EQU     0FC30H                  ;address of EOS jump table
<FD60>             100 EOS_GLB_RAM     EQU     0FD60H                  ;address of EOS global data area
<FEC0>             101 EOS_PCB_DCB     EQU     0FEC0H                  ;address of EOS PCB/DCB areas
                   102
                   103 ;
                   104 ;   These are key record sizes, used in various EOS routines.
                   105 ;   See "INCLUDE FMGR_EQUS" AND "INCLUDE P_DCB_EQU" for details.
                   106 ;
                   107
<001A>             108 DIR_ENT_LENGTH    DEFL    26
<0004>             109 P_SIZE            DEFL    4       ; THE NUMBER OF BYTES IN THE PCB
<0015>             110 D_SIZE            DEFL    21      ; THE NUMBER OF BYTES IN THE DCB
                   111 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     113 ;
                     114 ; jump table area
                     115 ;
                     116 ****************************************************
                     117
                     118           ORG EOS_JMP_TBL
                     119
                     120 ****************************************************
                     121
          <FC30>     122 _EOS_START                      EQU $
FC30 C3F832          123       JP        __EOS_START
          <FC33>     124 _CONS_DISP                      EQU $
FC33 C3F627          125       JP        __CONS_DISP
          <FC36>     126 _CONS_INIT                      EQU $
FC36 C3F5DC          127       JP        __CONS_INIT
          <FC39>     128 _CONS_OUT                       EQU $
FC39 C3F60A          129       JP        __CONS_OUT
          <FC3C>     130 _DLY_AFT_HRD_RES                EQU $
FC3C C3F95F          131       JP        __DLY_AFT_HRD_RES
          <FC3F>     132 _END_PR_BUFF                    EQU $
FC3F C3F588          133       JP        __END_PR_BUFF
          <FC42>     134 _END_PR_CH                      EQU $
FC42 C3F57C          135       JP        __END_PR_CH
          <FC45>     136 _END_RD_1_BLOCK                 EQU $
FC45 C3FAE2          137       JP        __END_RD_1_BLOCK
          <FC48>     138 _END_RD_CH_DEV                  EQU $
FC48 C3FBA5          139       JP        __END_RD_CH_DEV
          <FC4B>     140 _END_RD_KBD                     EQU $
FC4B C3F4E0          141       JP        __END_RD_KBD
          <FC4E>     142 _END_WR_1_BLOCK                 EQU $
FC4E C3FB1B          143       JP        __END_WR_1_BLOCK
          <FC51>     144 _END_WR_CH_DEV                  EQU $
FC51 C3FBE1          145       JP        __END_WR_CH_DEV
          <FC54>     146 _FIND_DCB                       EQU $
FC54 C3F446          147       JP        __FIND_DCB
          <FC57>     148 _GET_DCB_ADDR                   EQU $
FC57 C3F446          149       JP        __GET_DCB_ADDR
          <FC5A>     150 _GET_PCB_ADDR                   EQU $
FC5A C3FA4C          151       JP        __GET_PCB_ADDR
          <FC5D>     152 _HARD_INIT                      EQU $
FC5D C3F8F6          153       JP        __HARD_INIT
          <FC60>     154 _HARD_RESET_NET                 EQU $
FC60 C3F94B          155       JP        __HARD_RESET_NET
          <FC63>     156 _PR_BUFF                        EQU $
FC63 C3F515          157       JP        __PR_BUFF
          <FC66>     158 _PR_CH                          EQU $
FC66 C3F4FC          159       JP        __PR_CH
          <FC69>     160 _RD_1_BLOCK                     EQU $
FC69 C3FA9E          161       JP        __RD_1_BLOCK
          <FC6C>     162 _RD_KBD                         EQU $
FC6C C3F4BA          163       JP        __RD_KBD
          <FC6F>     164 _RD_KBD_RET_CODE                EQU $
FC6F C3FA7B          165       JP        __RD_KBD_RET_CODE
          <FC72>     166 _RD_PR_RET_CODE                 EQU $
FC72 C3FA7F          167       JP        __RD_PR_RET_CODE
          <FC75>     168 _RD_RET_CODE                    EQU $
FC75 C3F             169       JP        __RD_RET_CODE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

          <FC78>    170 _RD_TAPE_RET_CODE                EQU $
    FC78 C3FA83     171       JP      __RD_TAPE_RET_CODE
          <FC7B>    172 _RELOC_PCB                       EQU $
    FC7B C3FA2F     173       JP      __RELOC_PCB
          <FC7E>    174 _REQUEST_STATUS                  EQU $
    FC7E C3F473     175       JP      __REQUEST_STATUS
          <FC81>    176 _REQ_KBD_STAT                    EQU $
    FC81 C3F4CB     177       JP      __REQ_KBD_STAT
          <FC84>    178 _REQ_PR_STAT                     EQU $
    FC84 C3F5D2     179       JP      __REQ_PR_STAT
          <FC87>    180 _REQ_TAPE_STAT                   EQU $
    FC87 C3F5D7     181       JP      __REQ_TAPE_STAT
          <FC8A>    182 _SCAN_ACTIVE                     EQU $
    FC8A C3F9CB     183       JP      __SCAN_ACTIVE
          <FC8D>    184 _SOFT_INIT                       EQU $
    FC8D C3F922     185       JP      __SOFT_INIT
          <FC90>    186 _SOFT_RES_DEV                    EQU $
    FC90 C3FA5D     187       JP      __SOFT_RES_DEV
          <FC93>    188 _SOFT_RES_KBD                    EQU $
    FC93 C3FA51     189       JP      __SOFT_RES_KBD
          <FC96>    190 _SOFT_RES_PR                     EQU $
    FC96 C3FA55     191       JP      __SOFT_RES_PR
          <FC99>    192 _SOFT_RES_TAPE                   EQU $
    FC99 C3FA59     193       JP      __SOFT_RES_TAPE
          <FC9C>    194 _START_PR_BUFF                   EQU $
    FC9C C3F580     195       JP      __START_PR_BUFF
          <FC9F>    196 _START_PR_CH                     EQU $
    FC9F C3F56D     197       JP      __START_PR_CH
          <FCA2>    198 _START_RD_1_BLOCK                EQU $
    FCA2 C3FAC6     199       JP      __START_RD_1_BLOCK
          <FCA5>    200 _START_RD_CH_DEF                 EQU $
    FCA5 C3FB86     201       JP      __START_RD_CH_DEV
          <FCA8>    202 _START_RD_KBD                    EQU $
    FCA8 C3F4D0     203       JP      __START_RD_KBD
          <FCAB>    204 _START_WR_1_BLOCK                EQU $
    FCAB C3FAFF     205       JP      __START_WR_1_BLOCK
          <FCAE>    206 _START_WR_CH_DEV                 EQU $
    FCAE C3FBC2     207       JP      __START_WR_CH_DEV
          <FCB1>    208 _SYNC                            EQU $
    FCB1 C3F970     209       JP      __SYNC
          <FCB4>    210 _WR_1_BLOCK                      EQU $
    FCB4 C3FAB2     211       JP      __WR_1_BLOCK
          <FCB7>    212 _WR_CH_DEV                       EQU $
    FCB7 C3FB75     213       JP      __WR_CH_DEV
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     215 ;
                     216 ; FILE MANAGER ENTRIES
                     217 ;
          <FCBA>     218 _FMGR_INIT                      EQU $
FCBA C3EEEA          219         JP      __FMGR_INIT
          <FCBD>     220 _INIT_TAPE_DIR                  EQU $
FCBD C3F323          221         JP      __INIT_TAPE_DIR
          <FCC0>     222 _OPEN_FILE                      EQU $
FCC0 C3EA00          223         JP      __OPEN_FILE
          <FCC3>     224 _CLOSE_FILE                     EQU $
FCC3 C3EB04          225         JP      __CLOSE_FILE
          <FCC6>     226 _RESET_FILE                     EQU $
FCC6 C3EB6C          227         JP      __RESET_FILE
          <FCC9>     228 _MAKE_FILE                      EQU $
FCC9 C3E690          229         JP      __MAKE_FILE
          <FCCC>     230 _QUERY_FILE                     EQU $
FCCC C3E61B          231         JP      __QUERY_FILE
          <FCCF>     232 _SET_FILE                       EQU $
FCCF C3E651          233         JP      __SET_FILE
          <FCD2>     234 _READ_FILE                      EQU $
FCD2 C3EC17          235         JP      __READ_FILE
          <FCD5>     236 _WRITE_FILE                     EQU $
FCD5 C3ED8F          237         JP      __WRITE_FILE
          <FCD8>     238 _SET_DATE                       EQU $
FCD8 C3EEC5          239         JP      __SET_DATE
          <FCDB>     240 _GET_DATE                       EQU $
FCDB C3EED4          241         JP      __GET_DATE
          <FCDE>     242 _RENAME_FILE                    EQU $
FCDE C3F10F          243         JP      __RENAME_FILE
          <FCE1>     244 _DELETE_FILE                    EQU $
FCE1 C3F14E          245         JP      __DELETE_FILE
          <FCE4>     246 _RD_DEV_DEP_STAT                EQU $
FCE4 C3F488          247         JP      __RD_DEV_DEP_STAT
          <FCE7>     248 _GOTO_WP                        EQU $
FCE7 C3FA94          249         JP      __GOTO_WP
          <FCEA>     250 _READ_EOS                       EQU $
FCEA C3FA9D          251         JP      __READ_EOS
          <FCED>     252 _TRIM_FILE                      EQU $
FCED C3F241          253         JP      __TRIM_FILE
          <FCF0>     254 _CHECK_FCB                      EQU $
FCF0 C3F089          255         JP      __CHECK_FCB
          <FCF3>     256 _READ_BLOCK                     EQU $
FCF3 C3F17B          257         JP      __READ_BLOCK
          <FCF6>     258 _WRITE_BLOCK                    EQU $
FCF6 C3F1E6          259         JP      __WRITE_BLOCK
          <FCF9>     260 _MODE_CHECK                     EQU $
FCF9 C3F0D9          261         JP      __MODE_CHECK
          <FCFC>     262 _SCAN_FOR_FILE                  EQU $
FCFC C3EF0B          263         JP      __SCAN_FOR_FILE
          <FCFF>     264 _FILE_QUERY                     EQU $
FCFF C3E618          265         JP      __FILE_QUERY
          <FD02>     266 _POSIT_FILE                     EQU $
FD02 C3F442          267         JP      __POSIT_FILE
          <FD05>     268 _EOS_1                          EQU $
FD05 C3F442          269         JP      __EOS_1
          <FD08>     270 _EOS_2                          EQU $
FD08 C3              271         JP      __EOS_2
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

            <FD08>    272 _EOS_3                    EQU $
    FD08 C3F442       273       JP      __EOS_3
            <FD0E>    274 _CV_A                     EQU $
    FD0E C3F442       275       JP      __CV_A
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

          <FD11>    277 PORT_COLLECTION                 EQU $
    FD11 C3E191     278         JP      __PORT_COLLECTION
          <FD14>    279 SWITCH_MEM                      EQU $
    FD14 C3E185     280         JP      __SWITCH_MEM
          <FD17>    281 PUT_ASCII                       EQU $
    FD17 C3E153     282         JP      __PUT_ASCII
                    283
                    284 ;
                    285 ; micro OS7 entries
                    286 ;
                    287
          <FD1A>    288 WRITE_VRAM                      EQU $
    FD1A C3E000     289         JP      __WRITE_VRAM
          <FD1D>    290 READ_VRAM                       EQU $
    FD1D C3E01A     291         JP      __READ_VRAM
          <FD20>    292 WRITE_REGISTER                  EQU $
    FD20 C3E034     293         JP      __WRITE_REGISTER
          <FD23>    294 READ_REGISTER                   EQU $
    FD23 C3E04F     295         JP      __READ_REGISTER
          <FD26>    296 FILL_VRAM                       EQU $
    FD26 C3E059     297         JP      __FILL_VRAM
          <FD29>    298 INIT_TABLE                      EQU $
    FD29 C3E066     299         JP      __INIT_TABLE
          <FD2C>    300 PUT_VRAM                        EQU $
    FD2C C3E0C9     301         JP      __PUT_VRAM
          <FD2F>    302 GET_VRAM                        EQU $
    FD2F C3E0CF     303         JP      __GET_VRAM
          <FD32>    304 CALC_OFFSET                     EQU $
    FD32 C3E10A     305         JP      __CALC_OFFSET
          <FD35>    306 PX_TO_PTRN_POS                  EQU $
    FD35 C3E129     307         JP      __PX_TO_PTRN_POS
          <FD38>    308 LOAD_ASCII                      EQU $
    FD38 C3E149     309         JP      __LOAD_ASCII
          <FD3B>    310 WR_SPR_ATTRIBUTE                EQU $
    FD3B C3E1C5     311         JP      __WR_SPR_ATTRIBUTE      ;same as wr_spr_nm_tbl
          <FD3E>    312 POLLER                          EQU $
    FD3E C3E253     313         JP      __POLLER
          <FD41>    314 UPDATE_SPINNER                  EQU $
    FD41 C3E2A4     315         JP      __UPDATE_SPINNER
          <FD44>    316 DECLSN                          EQU $
    FD44 C3E355     317         JP      __DECLSN
          <FD47>    318 DECMSN                          EQU $
    FD47 C3E35F     319         JP      __DECMSN
          <FD4A>    320 MSNTOLSN                        EQU $
    FD4A C3E369     321         JP      __MSNTOLSN
          <FD4D>    322 ADD816                          EQU $
    FD4D C3E374     323         JP      __ADD816
          <FD50>    324 SOUND_INIT                      EQU $
    FD50 C3E3AB     325         JP      __SOUND_INIT
          <FD53>    326 TURN_OFF_SOUND                  EQU $
    FD53 C3E3D1     327         JP      __TURN_OFF_SOUND        ;not globalized in os7?
          <FD56>    328 PLAY_IT                         EQU $
    FD56 C3E3E7     329         JP      __PLAY_IT
          <FD59>    330 SOUNDS                          EQU $
    FD59 C3E406     331         JP      __SOUNDS                ;equals calls to play_songs and sound_man
          <FD5C>    332 EFFECT_OVER                     EQU $
    FD5C C3E[illegible]     333         JP      __EFFECT_OVER
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      335 ;
                      336 ; NOTE:   See INTERRUPT_VECTORS
                      337 ;
                      338 ;                ORG     EOS_GLB_TBL
                      339 ;               ;GLB     VECTOR_08H
                      340 ;VECTOR_08H      EQU     $+0
                      341 ;               ;GLB     INT_VCTR_TBL
                      342 ;INT_VCTR_TBL    EQU     VECTOR_08H
                      343 ;               ;GLB     VECTOR_10H
                      344 ;VECTOR_10H      EQU     $+3
                      345 ;               ;GLB     VECTOR_18H
                      346 ;VECTOR_18H      EQU     $+6
                      347 ;               ;GLB     VECTOR_20H
                      348 ;VECTOR_20H      EQU     $+9
                      349 ;               ;GLB     VECTOR_28H
                      350 ;VECTOR_28H      EQU     $+12
                      351 ;               ;GLB     VECTOR_30H
                      352 ;VECTOR_30H      EQU     $+15
                      353 ;               ;GLB     VECTOR_38H
                      354 ;VECTOR_38H      EQU     $+18
                      355 ;               ;GLB     VECTOR_66H
                      356 ;VECTOR_66H      EQU     $+21
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      358 ;
                      359 ;      NOTE: See SWITCH_TABLE
                      360 ;
                      361 ;;                ;GLB     MEM_CNFGOO
                      362 ;MEM_CNFGOO      EQU     $+24
                      363 ;;                ;GLB     SWITCH_TABLE
                      364 ;SWITCH_TABLE    EQU     MEM_CNFGOO
                      365 ;;                ;GLB     MEM_CNFGO1
                      366 ;MEM_CNFGO1      EQU     $+25
                      367 ;;                ;GLB     MEM_CNFGO2
                      368 ;MEM_CNFGO2      EQU     $+26
                      369 ;;                ;GLB     MEM_CNFGO3
                      370 ;MEM_CNFGO3      EQU     $+27
                      371 ;;                ;GLB     MEM_CNFGO4
                      372 ;MEM_CNFGO4      EQU     $+28
                      373 ;;                ;GLB     MEM_CNFGO5
                      374 ;MEM_CNFGO5      EQU     $+29
                      375 ;;                ;GLB     MEM_CNFGO6
                      376 ;MEM_CNFGO6      EQU     $+30
                      377 ;;                ;GLB     MEM_CNFGO7
                      378 ;MEM_CNFGO7      EQU     $+31
                      379 ;;                ;GLB     MEM_CNFGO8
                      380 ;MEM_CNFGO8      EQU     $+32
                      381 ;;                ;GLB     MEM_CNFGO9
                      382 ;MEM_CNFGO9      EQU     $+33
                      383 ;;                ;GLB     MEM_CNFGOA
                      384 ;MEM_CNFGOA      EQU     $+34
                      385 ;;                ;GLB     MEM_CNFGOB
                      386 ;MEM_CNFGOB      EQU     $+35
                      387 ;;                ;GLB     MEM_CNFGOC
                      388 ;MEM_CNFGOC      EQU     $+36
                      389 ;;                ;GLB     MEM_CNFGOD
                      390 ;MEM_CNFGOD      EQU     $+37
                      391 ;;                ;GLB     MEM_CNFGOE
                      392 ;MEM_CNFGOE      EQU     $+38
                      393 ;;                ;GLB     MEM_CNFGOF
                      394 ;MEM_CNFGOF      EQU     $+39
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     396 ;
                     397 ;     NOTE: See PORT_TABLE
                     398 ;
                     399 ;;                ;GLB     MEM_SWITCH_PORT
                     400 ;MEM_SWITCH_PORT EQU    $+40
                     401 ;;                ;GLB     PORT_TABLE
                     402 ;PORT_TABLE     EQU     MEM_SWITCH_PORT
                     403 ;;                ;GLB     NET_RESET_PORT
                     404 ;NET_RESET_PORT EQU     $+41
                     405 ;;                ;GLB     VDP_CTRL_PORT
                     406 ;VDP_CTRL_PORT  EQU     $+42
                     407 ;;                ;GLB     VDP_DATA_PORT
                     408 ;VDP_DATA_PORT  EQU     $+43
                     409 ;;                ;GLB     CONTROLLER_0_PORT
                     410 ;CONTROLLER_0_PORT EQU  $+44
                     411 ;;                ;GLB     CONTROLLER_1_PORT
                     412 ;CONTROLLER_1_PORT EQU  $+45
                     413 ;;                ;GLB     STROBE_SET_PORT
                     414 ;STROBE_SET_PORT EQU    $+46
                     415 ;;                ;GLB     STROBE_RESET_PORT
                     416 ;STROBE_RESET_PORT EQU  $+47
                     417 ;;                ;GLB     SOUNDPORT
                     418 ;SOUNDPORT       EQU     $+48
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     420
                     421 ;RAM DEFINITIONS FOR EOS
                     422
                     423 ;REV 0          (V/D     8-24-83)
                     424
                     425
                     426 ******************************************
                     427
                     428           ORG   EOS_GLB_RAM
                     429
                     430 ******************************************
                     431
                     432       ;GLB     CLEAR_RAM_START
                     433
FD60                 434 CLEAR_RAM_START:
                     435
                     436 ;
                     437 ; EOS revision number initialized by EOS_START
                     438 ;
                     439       ;GLB     REV_NUM
FD60                 440 REV_NUM
FD60                 441        DEFS    1
                     442
                     443
                     444        IF      .NT.SUPERGAME
                     445 ;
                     446 ; OS7 equivalent global data structures
                     447 ;
                     448                ;GLB     VDP_MODE_WORD
FD61                 449 VDP_MODE_WORD   DEFS    2
                     450                ;GLB     VDP_STATUS_BYTE
FD63                 451 VDP_STATUS_BYTE DEFS    1
                     452                ;GLB     VRAM_ADDR_TABLE
FD64                 453 VRAM_ADDR_TABLE
                     454                ;GLB     SPRITEATTRTBL
FD64                 455 SPRITEATTRTBL   DEFS    2
                     456                ;GLB     SPRITEGENTBL
FD66                 457 SPRITEGENTBL    DEFS    2
                     458                ;GLB     PATTRNNAMETBL
FD68                 459 PATTRNNAMETBL   DEFS    2
                     460                ;GLB     PATTRNGENTBL
FD6A                 461 PATTRNGENTBL    DEFS    2
                     462                ;GLB     COLORTABLE
FD6C                 463 COLORTABLE      DEFS    2
                     464              ELSE
                     465                 DEFS    2+6+1
                     466              ENDIF
                     467                 ;GLB     CUR_BANK
FD6E                 468 CUR_BANK        DEFS    1
                     469
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    471 ;
                    472 ; EOS global data structures
                    473 ;
                    474         ;GLB    DEFAULT_BT_DEV
                    475         ;GLB    CURRENT_DEV
                    476
FD6F                477 DEFAULT_BT_DEV:
FD6F                478 CURRENT_DEV:
FD6F                479         DEFS    1
                    480
                    481         ;GLB    CURRENT_PCB
                    482
FD70                483 CURRENT_PCB:
FD70                484         DEFS    2                       ; HOLDS THE START ADDRESS OF THE PCB
                    485         IF      .NT.SUPERGAME
                    486         ;GLB    DEVICE_ID
                    487         ENDIF
FD72                488 DEVICE_ID:
FD72                489         DEFS    1                       ; SAVE DEVICE ID
                    490
                    491         ;GLB    FILE_NAME_ADDR
FD73                492 FILE_NAME_ADDR
FD73                493         DEFS    2
                    494
                    495         ;GLB    KEYBOARD_BUFFER
                    496
FD75                497 KEYBOARD_BUFFER:
FD75                498         DEFS            1               ; HOLDS THE KEY THAT IS READ FROM THE
                    499                                         ; KEYBOARD
                    500
                    501         ;GLB    PRINT_BUFFER
                    502
FD76                503 PRINT_BUFFER:
FD76                504         DEFS            16              ; HOLDS THE STRING TO BE PRINTED
                    505
                    506         ;GLB    SECTORS_TO_INIT,SECTOR_NO
                    507
FD86                508 SECTORS_TO_INIT:
FD86                509         DEFS    1
                    510
FD87                511 SECTOR_NO:
FD87                512         DEFS    4
                    513
                    514         ;GLB    DCB_IMAGE
                    515
FD8B                516 DCB_IMAGE:
FD8B                517         DEFS    21
                    518
                    519         ;GLB    QUERY_BUFFER
FDA0                520 QUERY_BUFFER
FDA0                521         DEFS    DIR_ENT_LENGTH
                    522
                    523         ;GLB    FCB_BUFFER
FDBA                524 FCB_BUFFER
FDBA                525         DEFS    DIR_ENT_LENGTH
                    526
                    527         ;GLB    FILE_COUNT,MOD_FILE_COUNT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    FDD4              528 FILE_COUNT
    FDD4              529         DEFS    1
                      530
    FDD5              531 MOD_FILE_COUNT
    FDD5              532         DEFS    1
                      533
                      534         ;GLB    RETRY_COUNT,FILE_NUMBR
    FDD6              535 RETRY_COUNT
    FDD6              536         DEFS    1
                      537
    FDD7              538 FILE_NUMBR
    FDD7              539         DEFS     1
                      540
                      541         ;GLB    FILENAME_CMPS
    FDD8              542 FILENAME_CMPS
    FDD8              543         DEFS    1
                      544
                      545         ;GLB    DIR_BLOCK_NO,FOUND_AVAIL_ENT
    FDD9              546 DIR_BLOCK_NO
    FDD9              547         DEFS    2
                      548
    FDD8              549 FOUND_AVAIL_ENT
    FDD8              550         DEFS    1
                      551
                      552         ;GLB    VOL_BLK_SZ,BLK_STRT_PTR
    FDDC              553 BLK_STRT_PTR:
    FDDC              554 VOL_BLK_SZ:
    FDDC              555         DEFS    4
                      556
                      557 ; FILE MANAGER RAM STORAGE
                      558
                      559                 ;GLB    EOS_YEAR,EOS_MONTH,EOS_DAY
    FDE0              560 EOS_YEAR        DEFS    1       ;FMGR'S DATE STORAGE
    FDE1              561 EOS_MONTH       DEFS    1
    FDE2              562 EOS_DAY         DEFS    1
                      563
                      564                 ;GLB    FMGR_DIR_ENT
    FDE3              565 FMGR_DIR_ENT    DEFS    DIR_ENT_LENGTH ;PLACE FOR FMGR TO PUT A DIR. ENTRY
                      566
                      567                 ;GLB    FCB_HEAD_ADDR,FCB_DATA_ADDR
    FDFD              568 FCB_HEAD_ADDR   DEFS    2       ;POINTER TO START OF FCB HEADS
    FDFF              569 FCB_DATA_ADDR   DEFS    2       ;POINTER TO START OF FCB BUFFERS
                      570
                      571                 ;GLB    FNUM,BYTES_REQ,BYTES_TO_GO,USER_BUF
                      572                 ;GLB    BUF_START,BUF_END,BLOCKS_REQ
                      573                 ;GLB    USER_NAME,START_BLOCK
    FE01              574 FNUM            DEFS    1       ;FILE NUMBER GIVEN TO READ/WRITE ROUTINES
    FE02              575 BYTES_REQ       DEFS    2       ;NUMBER OF BYTES REQUESTED BY CALLER
    FE04              576 BYTES_TO_GO     DEFS    2       ;NUMBER OF BYTES STILL TO GIVE TO CALLER
    FE06              577 USER_BUF        DEFS    2       ;ADDRESS OF CALLER'S BUFFER
    FE08              578 BUF_START       DEFS    2       ;ADDRESS OF MY OWN BUFFER
    FE0A              579 BUF_END         DEFS    2       ;ADDRESS OF END + 1 OF MY BUFFER
    FE0C              580 BLOCKS_REQ      DEFS    4       ;NUMBER OF BLOCKS REQUESTED (MAKE_FILE)
    FE10              581 USER_NAME       DEFS    2       ;POINTER TO USER'S NAME STRING
    FE12              582 START_BLOCK     DEFS    4       ;START BLOCK # OF A FILE
                      583
                      584         ;GLB    NEW_HOLE_START,NEW_HOLE_SIZE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     585
    FE16             586 NEW_HOLE_START
    FE16             587         DEFS    4
                     588
    FE1A             589 NEW_HOLE_SIZE:
    FE1A             590         DEFS    2
                     591
    FE1C             592 STACK_START:
    FE1C             593         DEFS    60
                     594
                     595       ;GLB      EOS_STACK
    FE58             596 EOS_STACK:
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      598 ;             ;GLB    SAVED_COUNT
                      599                                       ; controller data area
                      600               ;GLB    SPIN_SWO_CT     ;THESE TWO BYTES MUST BE IN THIS ORDER!!!
                      601               ;GLB    SPIN_SW1_CT
                      602               ;GLB    PERSONAL_DEBOUNCE_TABLE
                      603               ;GLB    TEMP_STACK      ;Used by put_ascii when bank switching
                      604                                       ; sound data areas
                      605               ;GLB    PTR_TO_LST_OF_SND_ADDRS
                      606               ;GLB    PTR_TO_S_ON_0
                      607               ;GLB    PTR_TO_S_ON_1
                      608               ;GLB    PTR_TO_S_ON_2
                      609               ;GLB    PTR_TO_S_ON_3
                      610               ;GLB    SAVE_CTRL
                      611
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     613 ;SAVED_COUNT     DEFS     2
                     614 ;
                     615 ; controller data area
                     616 ;
    FE58             617 SPIN_SW0_CT      DEFS             1              ;THESE TWO BYTES MUST BE IN THIS ORDER!!!
    FE59             618 SPIN_SW1_CT      DEFS             1
    FE5A             619 PERSONAL_DEBOUNCE_TABLE           DEFS           08
    FE62             620                  DEFS             12             ;USED BY TEMP_STACK
            <FE6E>   621 TEMP_STACK       EQU              $              ;Used by put_ascii when bank switching
                     622
                     623 ;
                     624 ; sound data areas
                     625 ;
    FE6E             626 PTR_TO_LST_OF_SND_ADDRS DEFS       2
    FE70             627 PTR_TO_S_ON_0           DEFS       2
    FE72             628 PTR_TO_S_ON_1           DEFS       2
    FE74             629 PTR_TO_S_ON_2           DEFS       2
    FE76             630 PTR_TO_S_ON_3           DEFS       2
    FE78             631 SAVE_CTRL               DEFS       1
                     632
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     634 ;
                     635 ; data area for CONS_OUT
                     636 ;
                     637                ;GLB     OLDCHAR_
                     638                ;GLB     X_MIN
                     639                ;GLB     X_MAX
                     640                ;GLB     Y_MIN
                     641                ;GLB     Y_MAX
                     642                ;GLB     LINEBUFFER_
                     643                ;GLB     NUM_LINES
                     644                ;GLB     NUM_COLUMNS
                     645                ;GLB     UPPER_LEFT
                     646                ;GLB     PTRN_NAME_TBL
                     647                ;GLB     CURSOR
                     648
    FE79             649 OLDCHAR_        DEFS    1               ;Storage of char under cursor
    FE7A             650 X_MIN           DEFS    1               ;Absolute X coord of window
    FE7B             651 X_MAX           DEFS    1
    FE7C             652 Y_MIN           DEFS    1
    FE7D             653 Y_MAX           DEFS    1
    FE7E             654 LINEBUFFER_     DEFS    33
    FE9F             655 NUM_LINES       DEFS    1
    FEA0             656 NUM_COLUMNS     DEFS    1
    FEA1             657 UPPER_LEFT      DEFS    2
    FEA3             658 PTRN_NAME_TBL   DEFS    2
    FEA5             659 CURSOR          DEFS    2
                     660
                     661       ;GLB     CLEAR_RAM_SIZE
                     662
            <0147>   663 CLEAR_RAM_SIZE         EQU     ($-CLEAR_RAM_START)
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     665 ;
                     666 ; this is the PCB/DCB area
                     667 ;
                     668 ******************************************************:
                     669
                     670         ORG     EOS_PCB_DCB
                     671
                     672 ******************************************************
                     673
                     674        ;GLB     PCB
                     675
     FEC0            676 PCB:
     FEC0            677         DEFS    P_SIZE          ; PCB
                     678
     FEC4            679 DCBS:
     FEC4            680         DEFS    15*D_SIZE       ; 1 DCB PER NETWORK ADDRESS
                     681
     FFFF            682 RESERVED_BYTE:
     FFFF            683         DEFS    1               ; SAVED IN CASE WE DO FAST DMA HERE
                     684 ;
                     685 ;*********************************************************************
                     686 ;*********************************************************************
                     687 ;*********************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     689  NAME ^Rev 00 - RPD^
                     690
                     691
                     692 ********************************************
                     693
                     694                 ORG EOS_GLB_TBL
                     695
                     696 ********************************************
                     697 ;
                     698 ; NOTE: these are defined global in EOS_COMN
                     699 ;
                     700 ;                       ;GLB     VECTOR_08H
                     701 ;                       ;GLB     VECTOR_10H
                     702 ;                       ;GLB     VECTOR_18H
                     703 ;                       ;GLB     VECTOR_20H
                     704 ;                       ;GLB     VECTOR_28H
                     705 ;                       ;GLB     VECTOR_30H
                     706 ;                       ;GLB     VECTOR_38H
                     707 ;                       ;GLB     VECTOR_66H
                     708 ;
FBFF C9              709 VECTOR_08H      RET             ;rst 8
FC00 00              710                 NOP
FC01 00              711                 NOP
FC02 C9              712 VECTOR_10H      RET             ;rst 10
FC03 00              713                 NOP
FC04 00              714                 NOP
FC05 C9              715 VECTOR_18H      RET             ;rst 18
FC06 00              716                 NOP
FC07 00              717                 NOP
FC08 C9              718 VECTOR_20H      RET             ;rst 20
FC09 00              719                 NOP
FC0A 00              720                 NOP
FC0B C9              721 VECTOR_28H      RET             ;rst 28
FC0C 00              722                 NOP
FC0D 00              723                 NOP
FC0E C9              724 VECTOR_30H      RET             ;rst 30
FC0F 00              725                 NOP
FC10 00              726                 NOP
FC11 C9              727 VECTOR_38H      RET             ;rst 38
FC12 00              728                 NOP
FC13 00              729                 NOP
FC14 ED45            730 VECTOR_66H      RETN            ;nmi vector
FC16 00              731                 NOP
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     734
                     735 ;
                     736 ; NOTE: these are defined global in EOS_COMN
                     737 ;
                     738 ;              ;GLB     SWITCH_TABLE
                     739 ;              ;GLB     MEM_CNFGO0
                     740 ;              ;GLB     MEM_CNFGO1
                     741 ;              ;GLB     MEM_CNFGO2
                     742 ;              ;GLB     MEM_CNFGO3
                     743 ;              ;GLB     MEM_CNFGO4
                     744 ;              ;GLB     MEM_CNFGO5
                     745 ;              ;GLB     MEM_CNFGO6
                     746 ;              ;GLB     MEM_CNFGO7
                     747 ;              ;GLB     MEM_CNFGO8
                     748 ;              ;GLB     MEM_CNFGO9
                     749 ;              ;GLB     MEM_CNFGOA
                     750 ;              ;GLB     MEM_CNFGOB
                     751 ;              ;GLB     MEM_CNFGOC
                     752 ;              ;GLB     MEM_CNFGOD
                     753 ;              ;GLB     MEM_CNFGOE
                     754 ;              ;GLB     MEM_CNFGOF
                     755
<0000>               756 LO_BOOT_ROM_N_ALPHA_MASK              EQU     0000B
<0001>               757 LO_INTRINSIC_RAM_0_TO_7FFFH_MASK      EQU     0001B
<0002>               758 LO_EXPANSION_0_TO_7FFFH_MASK          EQU     0010B
<0003>               759 LO_OS7_N_INTRINSIC_2000H_7FFFH_MASK   EQU     0011B
                     760
<0000>               761 HI_INTRINSIC_8000H_TO_0FFFFH_MASK     EQU     0000B
<0004>               762 HI_AUXILLIARY_SLOT_8000H_0FFFFH       EQU     0100B
<0008>               763 HI_EXPANSION_8000H_TO_0FFFFH_MASK     EQU     1000B
<000C>               764 HI_COLECOVISION_CARTRIDGE_SLOT        EQU     1100B
                     765
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      767
    FC17              768 SWITCH_TABLE:
                      769
    FC17              770 MEM_CNFG00
    FC17 00           771                 DEFB    HI_INTRINSIC_8000H_TO_0FFFFH_MASK.OR.LO_BOOT_ROM_N_ALPHA_MASK
                      772
    FC18              773 MEM_CNFG01
    FC18 01           774                 DEFB    HI_INTRINSIC_8000H_TO_0FFFFH_MASK.OR.LO_INTRINSIC_RAM_0_TO_7FFFH_MASK
                      775
    FC19              776 MEM_CNFG02
    FC19 02           777                 DEFB    HI_INTRINSIC_8000H_TO_0FFFFH_MASK.OR.LO_EXPANSION_0_TO_7FFFH_MASK
                      778
    FC1A              779 MEM_CNFG03
    FC1A 03           780                 DEFB    HI_INTRINSIC_8000H_TO_0FFFFH_MASK.OR.LO_OS7_N_INTRINSIC_2000H_7FFFH_MASK
                      781
    FC1B              782 MEM_CNFG04
    FC1B 04           783                 DEFB    HI_AUXILLIARY_SLOT_8000H_0FFFFH.OR.LO_BOOT_ROM_N_ALPHA_MASK
                      784
    FC1C              785 MEM_CNFG05
    FC1C 05           786                 DEFB    HI_AUXILLIARY_SLOT_8000H_0FFFFH.OR.LO_INTRINSIC_RAM_0_TO_7FFFH_MASK
                      787
    FC1D              788 MEM_CNFG06
    FC1D 06           789                 DEFB    HI_AUXILLIARY_SLOT_8000H_0FFFFH.OR.LO_EXPANSION_0_TO_7FFFH_MASK
                      790
    FC1E              791 MEM_CNFG07
    FC1E 07           792                 DEFB    HI_AUXILLIARY_SLOT_8000H_0FFFFH.OR.LO_OS7_N_INTRINSIC_2000H_7FFFH_MASK
                      793
    FC1F              794 MEM_CNFG08
    FC1F 08           795                 DEFB    HI_EXPANSION_8000H_TO_0FFFFH_MASK.OR.LO_BOOT_ROM_N_ALPHA_MASK
                      796
    FC20              797 MEM_CNFG09
    FC20 09           798                 DEFB    HI_EXPANSION_8000H_TO_0FFFFH_MASK.OR.LO_INTRINSIC_RAM_0_TO_7FFFH_MASK
                      799
    FC21              800 MEM_CNFG0A
    FC21 0A           801                 DEFB    HI_EXPANSION_8000H_TO_0FFFFH_MASK.OR.LO_EXPANSION_0_TO_7FFFH_MASK
                      802
    FC22              803 MEM_CNFG0B
    FC22 0B           804                 DEFB    HI_EXPANSION_8000H_TO_0FFFFH_MASK.OR.LO_OS7_N_INTRINSIC_2000H_7FFFH_MASK
                      805
    FC23              806 MEM_CNFG0C
    FC23 0C           807                 DEFB    HI_COLECOVISION_CARTRIDGE_SLOT.OR.LO_BOOT_ROM_N_ALPHA_MASK
                      808
    FC24              809 MEM_CNFG0D
    FC24 0D           810                 DEFB    HI_COLECOVISION_CARTRIDGE_SLOT.OR.LO_INTRINSIC_RAM_0_TO_7FFFH_MASK
                      811
    FC25              812 MEM_CNFG0E
    FC25 0E           813                 DEFB    HI_COLECOVISION_CARTRIDGE_SLOT.OR.LO_EXPANSION_0_TO_7FFFH_MASK
                      814
    FC26              815 MEM_CNFG0F
    FC26 0F           816                 DEFB    HI_COLECOVISION_CARTRIDGE_SLOT.OR.LO_OS7_N_INTRINSIC_2000H_7FFFH_MASK
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     819
                     820 ;
                     821 ; NOTE: these are defined global in EOS_COMN
                     822 ;
                     823 ;                    ;GLB     PORT_TABLE              ;port value memory locations
                     824 ;                    ;GLB     MEM_SWITCH_PORT
                     825 ;                    ;GLB     NET_RESET_PORT
                     826 ;                    ;GLB     VDP_CTRL_PORT
                     827 ;                    ;GLB     VDP_DATA_PORT
                     828 ;                    ;GLB     CONTROLLER_0_PORT
                     829 ;                    ;GLB     CONTROLLER_1_PORT
                     830 ;                    ;GLB     STROBE_SET_PORT
                     831 ;                    ;GLB     STROBE_RESET_PORT
                     832 ;                    ;GLB     SOUNDPORT
                     833 ;
                     834         ************************************************
                     835         *** VDP ports must remain next to each other  ***
                     836         ***    and in CTRL -> DATA order              ***
                     837         ************************************************
                     838
FC27                 839 PORT_TABLE:
                     840
FC27                 841 MEM_SWITCH_PORT
FC27 7F              842                      DEFB    07FH            ;MEM_SWITCH_PORT        defined here
                     843
FC28                 844 NET_RESET_PORT
FC28 3F              845                      DEFB    03FH            ;Adam_net reset port    defined here
                     846
FC29                 847 VDP_CTRL_PORT
FC29 BF              848                      DEFB    0BFH            ;VDP ctrl port          01D43H
                     849
FC2A                 850 VDP_DATA_PORT
FC2A BE              851                      DEFB    0BEH            ;VDP data port          01D47H
                     852
FC2B                 853 CONTROLLER_0_PO
FC2B FC              854                      DEFB    0FCH            ;Controller 0           0114BH
                     855
FC2C                 856 CONTROLLER_1_PO
FC2C FF              857                      DEFB    0FFH            ;Controller 1           01151H
                     858
FC2D                 859 STROBE_SET_PORT
FC2D 80              860                      DEFB    080H            ;Controller strobe set  01157H
                     861
FC2E                 862 STROBE_RESET_PO
FC2E C0              863                      DEFB    0C0H            ;Controller strobe reset 01168H
                     864
FC2F                 865 SOUNDPORT
FC2F FF              866                      DEFB    0FFH            ;Sound port             0018EH
                     867 ;
                     868 ;*****************************************************************
                     869 ;*****************************************************************
                     870 ;*****************************************************************
                     871 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     874 ;
                     875 ; globals routines defined
                     876 ;
                     877                ;GLB     __WRITE_VRAM           ;VDP support routines
                     878                ;GLB     __READ_VRAM
                     879                ;GLB     __WRITE_REGISTER
                     880                ;GLB     __READ_REGISTER
                     881                ;GLB     __FILL_VRAM
                     882                ;GLB     __INIT_TABLE
                     883                ;GLB     __PUT_VRAM
                     884                ;GLB     __GET_VRAM
                     885                ;GLB     __CALC_OFFSET
                     886                ;GLB     __PX_TO_PTRN_POS
                     887                ;GLB     __LOAD_ASCII
                     888                ;GLB     __PUT_ASCII
                     889                ;GLB     __WR_SPR_ATTRIBUTE
                     890
                     891                ;GLB     __DECODER
                     892                ;GLB     __POLLER               ;controller support routines
                     893                ;GLB     __UPDATE_SPINNER
                     894
                     895                ;GLB     __DECLSN               ;sound chip support routines
                     896                ;GLB     __DECMSN
                     897                ;GLB     __MSNTOLSN
                     898                ;GLB     __ADD816
                     899                ;GLB     __SOUND_INIT
                     900                ;GLB     __TURN_OFF_SOUND
                     901                ;GLB     __PLAY_IT
                     902                ;GLB     __SOUNDS
                     903                ;GLB     __EFFECT_OVER
                     904
                     905                ;GLB     __SWITCH_MEM           ;memory bank switch support routine
                     906                ;GLB     __PORT_COLLECTION      ;STORE THE PORT ADDRESSES FROM OS_7
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      908 ;
                      909 ; external data areas used
                      910 ;
                      911                ;EXT     VDP_MODE_WORD
                      912                ;EXT     VDP_STATUS_BYTE
                      913                ;EXT     VRAM_ADDR_TABLE
                      914                ;EXT     SPRITEATTRTBL
                      915 ;***           ;EXT     SPRITEGENTBL
                      916 ;***           ;EXT     PATTRNNAMETBL
                      917                ;EXT     PATTRNGENTBL
                      918 ;***           ;EXT     COLORTABLE
                      919                ;EXT     CUR_BANK
                      920
                      921                                          ; controller data area
                      922                ;EXT     SPIN_SW0_CT      ;THESE TWO BYTES MUST BE IN THIS ORDER!!!
                      923 ;***           ;EXT     SPIN_SW1_CT
                      924                ;EXT     PERSONAL_DEBOUNCE_TABLE
                      925                ;EXT     TEMP_STACK        ;Used by put_ascii when bank switching
                      926                                          ; sound data areas
                      927                ;EXT     PTR_TO_LST_OF_SND_ADDRS
                      928                ;EXT     PTR_TO_S_ON_0
                      929                ;EXT     PTR_TO_S_ON_1
                      930                ;EXT     PTR_TO_S_ON_2
                      931                ;EXT     PTR_TO_S_ON_3
                      932                ;EXT     SAVE_CTRL
                      933 ;
                      934 ;    Port address and bank select externals follow
                      935 ;
                      936                ;EXT     PORT_TABLE
                      937                ;EXT     MEM_SWITCH_PORT
                      938 ;***           ;EXT     NET_RESET_PORT
                      939                ;EXT     VDP_CTRL_PORT
                      940 ;***           ;EXT     VDP_DATA_PORT
                      941                ;EXT     CONTROLLER_0_PORT
                      942 ;***           ;EXT     CONTROLLER_1_PORT
                      943                ;EXT     STROBE_SET_PORT
                      944 ;***           ;EXT     STROBE_RESET_PORT
                      945                ;EXT     SOUNDPORT
                      946
                      947 ;***           ;EXT     SWITCH_TABLE
                      948                ;EXT     MEM_CNFGO0
                      949 ;***           ;EXT     MEM_CNFGO1
                      950 ;***           ;EXT     MEM_CNFGO2
                      951                ;EXT     MEM_CNFGO3
                      952 ;***           ;EXT     MEM_CNFGO4
                      953 ;***           ;EXT     MEM_CNFGO5
                      954 ;***           ;EXT     MEM_CNFGO6
                      955 ;***           ;EXT     MEM_CNFGO7
                      956 ;***           ;EXT     MEM_CNFGO8
                      957 ;***           ;EXT     MEM_CNFGO9
                      958 ;***           ;EXT     MEM_CNFGOA
                      959 ;***           ;EXT     MEM_CNFGOB
                      960 ;***           ;EXT     MEM_CNFGOC
                      961 ;***           ;EXT     MEM_CNFGOD
                      962 ;***           ;EXT     MEM_CNFGOE
                      963 ;***           ;EXT     MEM_CNFGOF
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     965 ;
                     966 ; externals used
                     967 ;
                     968                ;EXT     WRITE_REGISTER
                     969 ;***           ;EXT     INIT_TABLE
                     970                ;EXT     SWITCH_MEM
                     971 ;***           ;EXT     PUT_VRAM
                     972                ;EXT     WRITE_VRAM
                     973                ;EXT     READ_VRAM
                     974 ;
                     975 ; local equates
                     976 ;
          <0102>     977 LOC_IN_ALPHA   EQU     0102H                ;pointer to ascii gens in rev 28+ alpha
                     978
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      981 ; Offsets into a volume descriptor (the first entry in the directory).
                <0000>  982 VOL_NAME         EQU     0       ;LOGICAL VOLUME NAME
                <000C>  983 VOL_DIRSIZE      EQU     12      ;*** 7 BITS ONLY *** # BLOCKS IN DIRECTORY
                <000C>  984 VOL_ATTR         EQU     12      ;*** TOP BIT ONLY *** SET FOR DELETE PROTECTION
                <000D>  985 VOL_DIR_CHECK    EQU     13      ; CONTAINS 4 UNIQUE BYTES FOR DIRECTORY EXISTENCE VERIFICATION
                <0011>  986 VOL_SIZE         EQU     17      ;VOLUME SIZE (IN BLOCKS) (4 BYTES)
                <0017>  987 VOL_YEAR         EQU     23      ;CREATION DATE -- YEAR
                <0018>  988 VOL_MONTH        EQU     24      ;                 MONTH
                <0019>  989 VOL_DAY          EQU     25      ;                 & DAY
                <001A>  990 VOL_DES_LENGTH   EQU     26
                        991
                        992 ; Offsets into a directory entry.
                <0000>  993 DIR_NAME         EQU     0       ;FILE NAME
                <000C>  994 DIR_ATTR         EQU     12      ;FILE ATTRIBUTE BYTE
                <000D>  995 DIR_START_BLOCK  EQU     13      ;STARTING BLOCK #
                <0011>  996 DIR_MAX_LENGTH   EQU     17      ;TOTAL # BLOCKS ALLOCATED
                <0013>  997 DIR_USED_LENGTH  EQU     19      ;# OF BLOCKS USED (FULL + 1 PARTIAL)
                <0015>  998 DIR_LAST_COUNT   EQU     21      ;NUMBER OF BYTES IN LAST PARTIAL BLOCK
                <0017>  999 DIR_YEAR         EQU     23      ;CREATION DATE -- YEAR
                <0018> 1000 DIR_MONTH        EQU     24      ;                 MONTH
                <0019> 1001 DIR_DAY          EQU     25      ;                 & DAY
                <001A> 1002 DIR_ENT_LENGTH   DEFL    26
                       1003
                <0027> 1004 ENT_PER_BLOCK    EQU     1024/26 ;NUMBER OF ENTRIES PER DIR BLOCK
                       1005
                       1006 ; Offsets into an FCB header.
                       1007
                       1008 ; Copy of DIR entry
                       1009
                <0000> 1010 FCB_NAME         EQU     0       ;FILE NAME
                <000C> 1011 FCB_ATTR         EQU     12      ;FILE ATTRIBUTE BYTE
                <000D> 1012 FCB_START_BLOCK  EQU     13      ;STARTING BLOCK #
                <000D> 1013 FCB_FIRST_BLOCK  EQU     FCB_START_BLOCK
                <0011> 1014 FCB_MAX_LENGTH   EQU     17      ;TOTAL # BLOCKS ALLOCATED
                <0013> 1015 FCB_USED_LENGTH  EQU     19      ;# OF BLOCKS USED (FULL + 1 PARTIAL)
                <0015> 1016 FCB_LAST_COUNT   EQU     21      ;NUMBER OF BYTES IN LAST PARTIAL BLOCK
                       1017
                <0016> 1018 FCB_STORED_BYTES EQU     FCB_LAST_COUNT+1 ; NUMBER OF BYTES STORED ON DEVICE
                       1019                                   ; ... THE UPPER HALF OF THIS FCB EQU LIST
                       1020
                <0017> 1021 FCB_DEVICE       EQU     26-3            ;NUMBER OF DEVICE CONTAINING FILE
                <0018> 1022 FCB_MODE         EQU     27-3            ;FILE MODE (UNUSED, READ, WRITE,APPEND)
                <0019> 1023 FCB_BLOCK        EQU     28-3            ;BLOCK NUMBER CURRENTLY IN BUFFER
                <001D> 1024 FCB_LAST_BLOCK   EQU     32-3            ;LAST BLOCK NUMBER IN FILE
                <0021> 1025 FCB_POINTER      EQU     36-3            ;POINTER INTO BLOCK BUFFER
                <0023> 1026 FCB_LENGTH       EQU     38-3
                       1027
                       1028 ; Possible FCB modes.
                       1029 *
                       1030 * PROTECTED MODE EQUATES
                       1031 *
                <0000> 1032 MODE_UNUSED      EQU     0       ;MANY PEOPLE ASSUME THIS IS ZERO!
                <0001> 1033 MODE_READ        EQU     MODE_UNUSED+1
                <0002> 1034 MODE_WRITE       EQU     MODE_READ+1
                <0003> 1035 MODE_UPDATE      EQU     MODE_WRITE+1
                <0004> 1036 MODE_EXEC        EQU     MODE_UPDATE+1
                <0004> 1037 MODE_MAX         EQU     MODE_EXEC+0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1038 *
<0005>               1039 MODE_REMAINDER_BIT EQU  5          ; INDICATES TO ALLOCATE REST OF TAPE
<0006>               1040 MODE_DIRTY_BIT  EQU     6          ; TEST BIT 6
<0040>               1041 MODE_DIRTY      EQU     010000000B ;INDICATES MODIFIED BUFFER
<0007>               1042 MODE_LAST_BLOCK_BIT  EQU  7        ; TEST BIT 7
<0080>               1043 MODE_LAST       EQU     100000000B ;INDICATES LAST BLOCK OF FILE
<0007>               1044 MODE_MODE       EQU     00000111B ;BITS TO STORE MODES
                     1045
                     1046 ; File attribute bits.
<0080>               1047 ATTR_PERMANENT  EQU     100000000B
<0040>               1048 ATTR_WRITE_PROT EQU     010000000B
<0020>               1049 ATTR_READ_PROT  EQU     001000000B
<0010>               1050 ATTR_USER       EQU     000100000B
<0008>               1051 ATTR_SYSTEM     EQU     000010000B
<0004>               1052 ATTR_DELETED    EQU     000001000B
<0002>               1053 ATTR_DEL_BIT    EQU     2
<0002>               1054 ATTR_EXECUTE    EQU     000000010B
<0000>               1055 ATTR_HOLE_BIT   EQU     0
<0001>               1056 ATTR_HOLE       EQU     00000001B
                     1057
                     1058 ; System-wide file name length.
<000C>               1059 NAME_LENGTH     EQU     12
                     1060
                     1061 ; Numbers of things we have.
<0003>               1062 NUM_FCBS        EQU     3       ;1 FOR THE SYSTEM, 2 FOR THE USER
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1065 DCB_NOT_FOUND          EQU     1       ; THERE WAS NO DCB FOR THE DEVICE REQUESTED.
                     1066 DCB_BUSY               EQU     2       ; DCB IS BUSY
                     1067 DCB_IDLE_ERR           EQU     3       ; DCB IS IDLE
                     1068
                     1069 NO_DATE_ERR            EQU     4
                     1070 NO_FILE_ERR            EQU     5
                     1071 FILE_EXISTS_ERR        EQU     6
                     1072 NO_FCB_ERR             EQU     7
                     1073 MATCH_ERR              EQU     8
                     1074 BAD_FNUM_ERR           EQU     9
                     1075 EOF_ERR                EQU     10
                     1076 TOO_BIG_ERR            EQU     11
                     1077 FULL_DIR_ERR           EQU     12
                     1078 FULL_TAPE_ERR          EQU     13
                     1079 FILE_NM_ERR            EQU     14              ;DLS(8/28/83)
                     1080 RENAME_ERR             EQU     15              ;DLS(8/30/83)
                     1081 DELETE_ERR             EQU     16              ;DLS(8/30/83)
                     1082 RANGE_ERR              EQU     17              ;DLS(8/31/83)
                     1083
                     1084 CANT_SYNC1             EQU     18
                     1085 CANT_SYNC2             EQU     19
                     1086 PRT_ERR                EQU     20
                     1087
                     1088 RQ_TP_STAT_ERR         EQU     21
                     1089 DEVICE_DEPD_ERR        EQU     22
                     1090 PROG_NON_EXIST         EQU     23          ; PROGRAM DOES NOT CURRENTLY EXIST
                     1091 NO_DIR_ERR             EQU     24          ; NO DIRECTORY ON TAPE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1094 ;THESE OUR EQUATES THAT ARE USED BY THE EOS PROGRAMS TO REFEFERNCE
                     1095 ;PCB AND DCB INFORMATION
                     1096
                     1097
                     1098 ;PCB EQUATES
                     1099
<0000>               1100 P_COM_STAT           EQU    0        ; THIS IS THE COMMAND/STATUS BYTE
                     1101
<0001>               1102 P_REL_ADDR           EQU    1        ; THIS IS THE RELOCATION ADDRESS
<0001>               1103 P_REL_ADDR_LO        EQU    P_REL_ADDR+0
<0002>               1104 P_REL_ADDR_HI        EQU    P_REL_ADDR+1
                     1105
<0003>               1106 P_NUM_DCBS           EQU    3        ; THIS IS THE NUMBER OF DCBS DEFINED
                     1107
                     1108
<0004>               1109 P_SIZE               DEFL   4        ; THE NUMBER OF BYTES IN THE PCB
                     1110
                     1111
                     1112
                     1113 ;DCB EQUATES
                     1114
<0000>               1115 D_COM_STAT           EQU    0        ; THE COMMAND STATUS BYTE
                     1116
<0001>               1117 D_BUF_ADR            EQU    1        ; ADDRESS OF THE DATA BUFFER
<0001>               1118 D_BUF_ADR_LO         EQU    D_BUF_ADR+0
<0002>               1119 D_BUF_ADR_HI         EQU    D_BUF_ADR+1
                     1120
<0003>               1121 D_BUF_LEN            EQU    3        ; THE LENGTH OF THE DATA BUFFER
<0003>               1122 D_BUF_LEN_LO         EQU    D_BUF_LEN+0
<0004>               1123 D_BUF_LEN_HI         EQU    D_BUF_LEN+1
                     1124
<0005>               1125 D_SECT_NUM           EQU    5        ; THE BLOCK DEVICE SECTOR NUMBER
                     1126
<0009>               1127 D_SEC_DEV_ID         EQU    9        ; SECONDARY DEVICE ID
                     1128
<000E>               1129 D_RET_COUNT          EQU    14       ; THE NUMBER OF TIMES A COMMAND WILL
                     1130                                      ; BE RETRIED.
<000E>               1131 D_RET_COUNT_LO       EQU    D_RET_COUNT+0
<000F>               1132 D_RET_COUNT_HI       EQU    D_RET_COUNT+1
                     1133
<0010>               1134 D_DEV_ADDR           EQU    16       ; THE DEVICE ADDRESS (ID)
                     1135
<0011>               1136 D_MAX_MSG_LEN        EQU    17       ; THE MAX LENGTH OF A DATA STRING
                     1137                                      ; FOR THE DEVICE
<0011>               1138 D_MAX_MSG_LN_LO      EQU    D_MAX_MSG_LEN+0
<0012>               1139 D_MAX_MSG_LN_HI      EQU    D_MAX_MSG_LEN+1
                     1140
<0013>               1141 D_DEV_TYPE           EQU    19       ; THE DEVICE TYPE, BLOCKED OR CHARACTER
                     1142
<0014>               1143 D_STATUS_FLAGS       EQU    20       ; DEVICE DEPENDENT STATUS FLAGS
                     1144
<0015>               1145 D_SIZE               DEFL   21       ; THE NUMBER OF BYTES IN THE DCB
                     1146
                     1147
                     1148
                     1149 ;DEVICE ID'S FOR THE KEYBOARD, PRINTER, AND TAPE DRIVE
                     1150
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

              <0001>  1151 KEYBOARD_ID            EQU     1       ; KYBD  ID
              <0002>  1152 PRINTER_ID             EQU     2       ; PRINTER ID
              <0008>  1153 TAPE_ID                EQU     8       ; TAPE DRIVE ID
                      1154
              <0002>  1155 ERROR_RETRY            EQU     2       ; MAX RETRYS ON ERRORS, READ_BLOCK AND WRITE_BLOCK
                      1156
              <000F>  1157 MAX_DEV_ADDR           EQU     15      ; HIGEST POSSIBLE DEVICE ADDRESS
                      1158                                        ;  ON NETWORK
                      1159
                      1160
                      1161 ;PCB COMMAND EQUATES
                      1162
                      1163
              <0000>  1164 PCB_IDLE               EQU     0       ; THIS IS AN IDLE STATE
                      1165
              <0001>  1166 PCB_SYNC1              EQU     1       ; SYNC BYTE 1
              <0081>  1167 PCB_SYNC1_ACK          EQU     PCB_SYNC1+80H
                      1168
              <0002>  1169 PCB_SYNC2              EQU     2       ; SYNC BYTE 2
              <0082>  1170 PCB_SYNC2_ACK          EQU     PCB_SYNC2+80H
                      1171
              <0003>  1172 PCB_SNA                EQU     3       ; SET NEW PCB ADDRESS
              <0083>  1173 PCB_SNA_ACK            EQU     PCB_SNA+80H
                      1174
              <0004>  1175 PCB_RESET              EQU     4       ; RESET ALL NODES
              <0084>  1176 PCB_RESET_ACK          EQU     PCB_RESET+80H
                      1177
              <0005>  1178 PCB_WAIT               EQU     5       ;
              <0085>  1179 PCB_WAIT_ACK           EQU     PCB_WAIT+80H
                      1180
                      1181
                      1182
                      1183 ;DCB COMMAND EQUATES
                      1184
              <0000>  1185 DCB_IDLE               EQU     00      ;
              <0001>  1186 DCB_STATUS             EQU     01      ; REQUEST STATUS
              <0002>  1187 DCB_RESET              EQU     02      ; RESET NODE
              <0003>  1188 DCB_WR                 EQU     03      ; WRITE DATA TO DEVICE
              <0004>  1189 DCB_RD                 EQU     04      ; READ DATA FROM DEVICE
                      1190
                      1191
                      1192
              <FEC0>  1193 INIT_PCB_ADDR          EQU     0FEC0H  ; INITIAL ADDRESS OF THE PCB
                      1194
              <D390>  1195 FCB_S                  EQU     0D390H  ; FCB HEADER AREA
              <D400>  1196 THREE1K_BLKS           EQU     0D400H  ; 3K FCB DATA AREA
                      1197
                      1198
                      1199 ;GENERAL USAGE EQUATES FOR USE WITH DCB INFO
                      1200
              <0007>  1201 CMND_COMPLETE_BIT      EQU     7       ; THIS IS THE BIT THAT INDICATES THE
                      1202                                        ; COMMAND HAS BEEN PROCESSED.
              <0080>  1203 CMND_FIN_STATUS        EQU     80H     ; THIS IS THE STATUS OF A COMMAND
                      1204                                        ; THAT COMPLETED WITH NO ERRORS
              <008C>  1205 KBD_NAK                EQU     8CH     ; INDICATES NO KEY READY
                      1206
              <0086>  1207 PR_NAK                 EQU     86H     ; INDICATES THE PRINTER IS BUSY
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1208
             <0003>  1209 ETX                    EQU     03H     ; END OF DATA STRING INDICATOR
                     1210
             <009B>  1211 TIMEOUT                EQU     9BH     ; DEVICE TIMED OUT
                     1212 ;
                     1213 ;*************************************************************************
                     1214 ;*************************************************************************
                     1215 ;*************************************************************************
                     1216 ;
    FC30             1217 SKIP
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    1219  NAME ^Rev 07 - jkl^
                    1220
                    1221 De_A_uOS_00 MACRO              ;Header Rev. 5
                    1222                .GOTO Ede_A_uOS_00
                    1223
                    1224  Project:      ADAM, 83-101
                    1225
                    1226      ************************************************************
                    1227      ****                                                    ****
                    1228      ****              A_uOS_00              RPD             ****
                    1229      ****                                                    ****
                    1230      ************************************************************
                    1231
                    1232
                    1233        Rev History
                    1234        Rev.   Date          Name      Change
                    1235         7     08oct1545     jkl       make __LOAD_ASCII load chars from 0 to 7F
                    1236         6     08oct0003     rfj       Fixed 100H bug with Bob Greenberg's
                    1237                                       algorithm
                    1238         5     07oct1708     rfj       __EFXOVER changed to __EFFECT_OVER
                    1239         4     05oct10:40    rfj       Decoder now saves Acc before getting
                    1240                                       interrupt data
                    1241                                       Removed excess documentation on Put/Load Ascii
                    1242                                       Commented out all unreferenced symbols with ;eee
                    1243                                       Renamed WR_SPR_NM_TBL to WR_SPR_ATTRIBUTE
                    1244                                       globalized __UPDATE_SPINNER
                    1245                                       new address for the ASCII table referenced
                    1246         3     28sept1317    rfj       Multy changes after personal review
                    1247                                       but not approved
                    1248         2     25sept900a    RPD       made ram external
                    1249         1     24sept1250p   RPD       addition of more routines & new port accessing
                    1250         0     24aug430p     RPD       Initial creation date (READ & WRITE VRAM)
                    1251
                    1252  A_uOS_00 stands for ADAM micro Operating System version 0. This file
                    1253  contains a scaled down model of COLECOVISION OS 7. Primary routines from
                    1254  OS 7 are duplicated for ADAM application programs use. The functional areas
                    1255  that are supported by these routines are the VDP, the controllers, the sound
                    1256  chip, memory bank switch, port collection.
                    1257
                    1258 Ede_A_uOS_00  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1260
                     1261 ************************************************************
                     1262
                     1263                 ORG  EOS_CODE
                     1264
                     1265 ************************************************************
                     1266
                     1267 ;  Name:            WRITE_VRAM
                     1268 ;
                     1269 ;  Function:        Writes to VRAM the contents of the data in a buffer area.
                     1270 ;
                     1271 ;  Entry:           BC - number of bytes to be written
                     1272 ;                   DE - starting VRAM address to be written to
                     1273 ;                   HL - address of buffer containing the data
                     1274 ;
                     1275 ;  Exit:            None.
                     1276 ;
                     1277 ;  Registers used: AF, BC, DE, HL
                     1278 ;
                     1279 ;  Size:            ROM - 28 bytes
                     1280 ;                   RAM - 0 bytes
                     1281 ;
                     1282 ;  Comments:        This version of WRITE_VRAM is provided for EOS operation.
                     1283 ;                   NOTE: The 100H bug found in OS 7 has been corrected.
                     1284 ;
                     1285 ;  Comparision:     left out pascal entry point
                     1286 ;                   code compacted
                     1287 ;                   100H bug fixed
                     1288 ;                   edited for documentation
                     1289 ;                   reference port table
                     1290
E000                 1291 __WRITE_VRAM:
E000 C5              1292                 PUSH    BC                      ;BC has the number of bytes to xfer
E001 EB              1293                 EX      DE,HL                   ;HL<--vram addr,DE<--source address
E002 CDE1E9          1294                 CALL    SET_WRITE               ;pump the vram addr to the vdp
E005 69              1295                 LD      L,C                     ;save C (the data_port_addr)
E006 C1              1296                 POP     BC                      ;restore the xfer count
E007 EB              1297                 EX      DE,HL                   ;HL<--source address,port in E
E008 79              1298                 LD      A,C                     ;Save low order of count in A
E009 4B              1299                 LD      C,E                     ; And get the port into C
E00A 50              1300                 LD      D,B                     ;Get HI order count in D
                     1301                                                 ; to free up B for low order count
E00B 14              1302                 INC     D                       ;Increment HI count to cover
                     1303                                                 ; for 0 in DEC D after low loop
E00C 47              1304                 LD      B,A                     ;Put low order in B for OUTI
E00D B7              1305                 OR      A                       ;Check if low order is zero the first time
E00E 2806            1306                 JR      Z,OUT_DEC_HI_BYTE       ;If so then decrement HI count before low loop
E010                 1307 OUTPUT_LOOP:
E010 EDA3            1308                 OUTI                            ;DATA_PORT = buffer_data  write the data to the VDP
                     1309                                                 ;for OUTI: [C] <-- [HL], B = B - 1 and HL = HL + 1
E012 00              1310                 NOP                             ;delay for slow VDP
E013 00              1311                 NOP
E014 20FA            1312                 JR      NZ,OUTPUT_LOOP          ;until byte count low (B reg) = 0
E016                 1313 OUT_DEC_HI_BYTE:
E016 15              1314                 DEC     D                       ;byte count high = byte count high - 1
E017 20F7            1315                 JR      NZ,OUTPUT_LOOP          ;until byte count high (A reg) = 0
E019 C9              1316                 RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1318 ;
                     1319 ;  Name:            READ_VRAM
                     1320 ;
                     1321 ;  Function:        Reads from VRAM and puts the read data into a buffer area.
                     1322 ;
                     1323 ;  Entry:           BC - number of bytes to be read
                     1324 ;                   DE - starting VRAM address to be read from
                     1325 ;                   HL - address of buffer to receive the read data
                     1326 ;
                     1327 ;  Exit:            None.
                     1328 ;
                     1329 ;  Registers used:  AF, BC, DE, HL
                     1330 ;
                     1331 ;  Size:            ROM - 22 bytes
                     1332 ;                   RAM - 0 bytes
                     1333 ;
                     1334 ;  Comments:        This version of READ_VRAM is provided for stand alone operation.
                     1335 ;                   NOTE: The 100H bug found in OS 7 has been corrected.
                     1336 ;
                     1337 ;  Comparision:     left out pascal entry point
                     1338 ;                   code compacted
                     1339 ;                   100H bug fixed
                     1340 ;                   edited for documentation
                     1341 ;
E01A                 1342 __READ_VRAM:
E01A C5              1343                 PUSH    BC                 ;BC has the number of bytes to xfer
E01B EB              1344                 EX      DE,HL              ;HL<--vram addr,DE<--source address.
E01C CDE1E7          1345                 CALL    SET_READ           ;pump the vram addr to the vdp
E01F 69              1346                 LD      L,C                ;save C (the data_port_addr)
E020 C1              1347                 POP     BC                 ;restore the xfer count
E021 EB              1348                 EX      DE,HL              ;HL<--source address,port in E
E022 79              1349                 LD      A,C                ;Save low order of count in A
E023 48              1350                 LD      C,E                ; And get the port into C
E024 50              1351                 LD      D,B                ;Get HI order count in D
                     1352                                            ; to free up B for low order count
E025 14              1353                 INC     D                  ;Increment Hi count to cover
                     1354                                            ; for 0 in DEC D after low loop
E026 47              1355                 LD      B,A                ;Put low order in B for OUTI
E027 B7              1356                 OR      A                  ;Check if low order is zero the first time
E028 2806            1357                 JR      Z,IN_DEC_HI_BYTE   ;If so then decrement Hi count before low loop
E02A                 1358 INPUT_LOOP:
E02A EDA2            1359                 INI                        ;DATA_PORT = buffer_data  read the data from to the VDP
                     1360                                            ;for INI: [C] <-- [HL], B = B - 1 and HL = HL + 1
E02C 00              1361                 NOP                        ;delay for slow VDP
E02D 00              1362                 NOP
E02E 20FA            1363                 JR      NZ,INPUT_LOOP      ;until byte count low (B reg) = 0
E030                 1364 IN_DEC_HI_BYTE:
E030 15              1365                 DEC     D                  ;byte count high = byte count high - 1
E031 20F7            1366                 JR      NZ,INPUT_LOOP      ;until byte count high (A reg) = 0
E033 C9              1367                 RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1369 ;
                     1370 ;  Name:            WRITE_REGISTER
                     1371 ;
                     1372 ;  Function:        Writes a data byte value to a desired VDP register
                     1373 ;
                     1374 ;  Entry:           B - register number to write to
                     1375 ;                   C - data byte value to be written
                     1376 ;
                     1377 ;  Exit:            if register number = 0 or 1, the respective byte
                     1378 ;                   of the VDP_MODE_WORD is updated.
                     1379 ;
                     1380 ;  Registers used:  A,BC,E
                     1381 ;
                     1382 ;  Size:            ROM - 27 bytes
                     1383 ;                   RAM - 0 bytes
                     1384 ;
                     1385 ;  Comments:        This version of WRITE_REGITER is provided for stand alone operation.
                     1386 ;
                     1387 ;  Comparision:     left out pascal entry point
                     1388 ;                   code compacted
                     1389 ;                   edited for documentation
                     1390 ;
E034                 1391 __WRITE_REGISTER:
E034 59              1392                 LD      E,C                     ;Move data to E register
E035 3AFC29          1393                 LD      A,[VDP_CTRL_PORT]       ;Get the control port value
E038 4F              1394                 LD      C,A                     ;Put in C
E039 ED59            1395                 OUT     [C],E                   ;Output data value
E03B 78              1396                 LD      A,B                     ;Get port # in A
E03C F680            1397                 OR      80H                     ;Set high bit for ???????
E03E ED79            1398                 OUT     [C],A                   ;Out to VDP
E040 78              1399                 LD      A,B                     ;Get back port value
E041 B7              1400                 OR      A                       ;Check for zero
E042 7B              1401                 LD      A,E                     ;Put data in accumulator
                     1402
E043 2004            1403                 JR      NZ,CHK_REG_1            ;If not zero then see if register 1
E045 32FD61          1404                 LD      [VDP_MODE_WORD],A       ;If register zero was written to then
                     1405                                                 ; save data in VDP_MODE_WORD
E048 C9              1406                 RET                             ;Exit
                     1407
E049                 1408 CHK_REG_1:
E049 05              1409                 DEC     B                       ;Check for register 1 in B
E04A C0              1410                 RET     NZ                      ;If not then return
E04B 32FD62          1411                 LD      [VDP_MODE_WORD+1],A     ;If so then store data in VDP_MODE_WORD+1
E04E C9              1412                 RET                             ;Exit
                     1413
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1415 ;
                     1416 ;  Name:            READ_REGISTER
                     1417 ;
                     1418 ;  Function:        Reads a data byte value from the Colecovision CTRL_PORT.
                     1419 ;
                     1420 ;  Entry:           None.
                     1421 ;
                     1422 ;  Exit:            A - data byte value read in
                     1423 ;
                     1424 ;  Registers used:  A,C
                     1425 ;
                     1426 ;  Size:            ROM - 3 bytes
                     1427 ;                   RAM - 0 bytes
                     1428 ;
                     1429 ;  Comments:        This version of READ_REGISTER is provided for stand alone operation.
                     1430 ;                   This routine does a direct access to the COLECOVISION I/O ports.
                     1431 ;
                     1432 ;  Comparision:     edited for documentation
                     1433 ;
    E04F             1434 __READ_REGISTER:
    E04F 3AFC29      1435                 LD      A,[VDP_CTRL_PORT]
    E052 4F          1436                 LD      C,A
    E053 ED78        1437                 IN      A,[C]                   ;get the data from CTRL_PORT
    E055 32FD63      1438                 LD      [VDP_STATUS_BYTE],A     ;save a copy of VDP status data
    E058 C9          1439                 RET
                     1440
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1442 ;
                     1443 ;  Name:           FILL_VRAM
                     1444 ;
                     1445 ;  Function:       Fill a VRAM memory buffer with a constant byte value.
                     1446 ;
                     1447 ;  Entry:          A - constant byte value to be written
                     1448 ;                  DE - number of bytes in the VRAM memory buffer
                     1449 ;                  HL - starting VRAM address to be written to
                     1450 ;
                     1451 ;  Exit:           None.
                     1452 ;
                     1453 ;  Registers used: AF, C, DE
                     1454 ;
                     1455 ;  Size:           ROM - 18 bytes
                     1456 ;                  RAM - 0 bytes
                     1457 ;
                     1458 ;  Comments:       This version of FILL_VRAM is provided for stand alone operation.
                     1459 ;                  This routine does a direct access to the COLECOVISION I/O ports.
                     1460 ;
                     1461 ;  Comparision:    edited for documentation
                     1462 ;                  uneeded call to READ_REGISTER taken out
                     1463 ;
                     1464 ;
    E059             1465 __FILL_VRAM:
    E059 F5          1466                 PUSH    AF                      ;Save the data to fill with
    E05A CDE1E9      1467                 CALL    SET_WRITE               ;Sets the ctrl port data and
    E05D E1          1468                 POP     HL                      ;popping AF into HL to restore the fill data
    E05E             1469 FILL:           -                               ;repeat
                     1470                                                 ;  get back a copy of the saved constant
    E05E ED61        1471                 OUT     [C],H                   ;  write the constant out to the VRAM buffer
    E060 1B          1472                 DEC     DE                      ;  byte count = byte count - 1
    E061 7A          1473                 LD      A,D                     ;until byte count = 0
    E062 B3          1474                 OR      E
    E063 20F9        1475                 JR      NZ,FILL
    E065 C9          1476                 RET                             ;all done
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1478 ;
                     1479 ;
                     1480 ;  Name:            INIT_TABLE
                     1481 ;
                     1482 ;  Function:        INIT_TABLE initializes the addresses of the VRAM tables.
                     1483 ;                   The passed address is converted to the correct format needed
                     1484 ;                   to setup the VDP address registers. The following table codes
                     1485 ;                   are use to indentify which table address is being setup:
                     1486 ;
                     1487 ;                     0 - SPRITE ATTRIBUTE TABLE  (SAT)
                     1488 ;                     1 - SPRITE GENERATOR TABLE  (SGT)
                     1489 ;                     2 - PATTERN NAME TABLE      (PNT)
                     1490 ;                     3 - PATTERN GENERATOR TABLE (PGT)
                     1491 ;                     4 - PATTERN COLOR TABEL     (PCT)
                     1492 ;
                     1493 ;  Entry:           A - table code (see above)
                     1494 ;                   HL - table address
                     1495 ;
                     1496 ;  Exit:            None.
                     1497 ;
                     1498 ;  Registers used:  AF, BC, HL, IX, IY
                     1499 ;
                     1500 ;  Size:            ROM - 99 bytes
                     1501 ;                   RAM - 0 bytes
                     1502 ;
                     1503 ;  Comments:        This version of INIT_TABLE is provided for stand alone operation.
                     1504 ;
                     1505 ;  Comparision:     left out pascal entry point
                     1506 ;                   code compacted
                     1507 ;                   edited for documentation
                     1508 ;
E066                 1509 __INIT_TABLE:
E066 4F              1511                 LD      C,A                     ;BC = index into VRAM_ADDR_TABLE  table code
E067 0600            1512                 LD      B,0
E069 DD21FD64        1513                 LD      IX,VRAM_ADDR_TABLE      ;point to start of VRAM_ADDR_TABLE
E06D DD09            1514                 ADD     IX,BC                   ;add in the index
E06F DD09            1515                 ADD     IX,BC
E071 DD7500          1516                 LD      [IX+0],L                ;save the address
E074 DD7401          1517                 LD      [IX+1],H
                     1518                                                 ;CHECK VDP GRAPHICS MODE
E077 3AFD61          1519                 LD      A,[VDP_MODE_WORD]       ;if VDP_MODE_WORD bit 1  mode bit 3  = 1
E07A CB4F            1520                 BIT     1,A
E07C 2827            1521                 JR      Z,INIT_TABLE80
                     1522                                                 ;  its graphics mode 2, look for PGT or PCT first
E07E 79              1523                 LD      A,C                     ;  get the table code
E07F FE03            1524                 CP      3                       ;  if table code = PGT
E081 2806            1525                 JR      Z,CASE_OF_GEN           ;    then setup PGT address
E083 FE04            1526                 CP      4                       ;    if table code = PCT
E085 2810            1527                 JR      Z,CASE_OF_COLOR         ;    then setup PCT address
E087 181C            1528                 JR      INIT_TABLE80            ;    else not a special case, use normal setup
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1530 ;
                     1531 ; special case: graphics mode 2, address of PGT
                     1532 ;
E089                 1533 CASE_OF_GEN:
E089 0604            1534                LD      B,4                 ;register number = PGT base address register
E08B 7D              1535                LD      A,L                 ;if PGT address = 0000H
E08C B4              1536                OR      H
E08D 2004            1537                JR      NZ,CASE_OF_GEN10
E08F 0E03            1538                LD      C,3                 ;  data byte value for address 0000H
E091 1828            1539                JR      INIT_TABLE90
E093                 1540 CASE_OF_GEN10:                             ;else  PGT address <> 0000H
E093 0E07            1541                LD      C,7                 ;  data byte value for address 2000H
E095 1824            1542                JR      INIT_TABLE90        ;endif
                     1543 ;
                     1544 ; special case: graphics mode 2, address of PCT
                     1545 ;
E097                 1546 CASE_OF_COLOR:
E097 0603            1547                LD      B,3                 ;register number = PCT base address register
E099 7D              1548                LD      A,L                 ;if PCT address = 0000H
E09A B4              1549                OR      H
E09B 2004            1550                JR      NZ,CASE_OF_CLR10
E09D 0E7F            1551                LD      C,7FH               ;  data byte value for address 0000H
E09F 181A            1552                JR      INIT_TABLE90
E0A1                 1553 CASE_OF_CLR10:                             ;else  PCT address <> 0000H
E0A1 0EFF            1554                LD      C,0FFH              ;  data byte value for address 2000H
E0A3 1816            1555                JR      INIT_TABLE90        ;endif
                     1556 ;
                     1557 ; not a special case (i.e. not mode 2, PGT or PCT)
                     1558 ;
E0A5                 1559 INIT_TABLE80:                              ;base address = table address / factor
E0A5 FD21E0BF        1560                LD      IY,BASE_FACTORS     ;point to start of the the factor table
E0A9 FD09            1561                ADD     IY,BC               ;index into the table by the table code
E0AB FD09            1562                ADD     IY,BC
E0AD FD7E00          1563                LD      A,[IY+0]            ;get the shift count
E0B0 FD4601          1564                LD      B,[IY+1]            ;get the register number
E0B3                 1565 DIVIDE:
E0B3 CB3C            1566                SRL     H                   ;shift hi-byte
E0B5 CB1D            1567                RR      L                   ;shift lo-byte
E0B7 3D              1568                DEC     A                   ;shift count = shift count - 1
E0B8 20F9            1569                JR      NZ,DIVIDE
E0BA 4D              1570                LD      C,L                 ;get the data byte value
E0BB                 1571 INIT_TABLE90:
E0BB CDFD20          1572                CALL    WRITE_REGISTER      ;setup the base register
E0BE C9              1573                RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1575 ;
                     1576 ; base factor table, used to calculate base addresses
                     1577 ;
    E0BF             1578 BASE_FACTORS:
    E0BF 07          1579                DEFB    7                   ;table code 0 base factor
    E0C0 05          1580                DEFB    5                   ;register 5, SAT base address
    E0C1 0B          1581                DEFB    11                  ;table code 1 base factor
    E0C2 06          1582                DEFB    6                   ;register 6, SGT base address
    E0C3 0A          1583                DEFB    10                  ;table code 2 base factor
    E0C4 02          1584                DEFB    2                   ;register 2, PNT base address
    E0C5 0B          1585                DEFB    11                  ;table code 3 base factor
    E0C6 04          1586                DEFB    4                   ;register 4, PGT base address
    E0C7 06          1587                DEFB    6                   ;table code 4 base factor
    E0C8 03          1588                DEFB    3                   ;register 3, PCT base address
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1590 ;
                     1591 ;  Name:           PUT_VRAM
                     1592 ;
                     1593 ;  Function:       Gets a block of data from a user buffer and puts it into VRAM.
                     1594 ;                  The following table codes are used to identify which VRAM table
                     1595 ;                  is being referenced:
                     1596 ;
                     1597 ;                    0 - SPRITE ATTRIBUTE TABLE  (SAT)
                     1598 ;                    1 - SPRITE GENERATOR TABLE  (SGT)
                     1599 ;                    2 - PATTERN NAME TABLE      (PNT)
                     1600 ;                    3 - PATTERN GENERATOR TABLE (PGT)
                     1601 ;                    4 - PATTERN COLOR TABLE     (PCT)
                     1602 ;
                     1603 ;  Entry:          A - table code (see above)
                     1604 ;                  DE - starting index into the table
                     1605 ;                  HL - address of user buffer
                     1606 ;                  IY - block size (or byte count)
                     1607 ;
                     1608 ;  Exit:           None.
                     1609 ;
                     1610 ;  Registers used: AF, DE, HL, IY
                     1611 ;
                     1612 ;  Routines used:  SET_COUNT
                     1613 ;
                     1614 ;  Size:           ROM - 0 bytes
                     1615 ;                  RAM - 0 bytes
                     1616 ;
                     1617 ;  Comments:       This version of PUT_VRAM is provided for stand alone operation.
                     1618 ;
                     1619 ;  Comparision:    left out pascal entry point
                     1620 ;                  took out the mux sprites capablity
                     1621 ;                  code compacted
                     1622 ;                  edited for documentation
                     1623 ;
EOC9                 1624 __PUT_VRAM:
EOC9 CDE0D5          1625                 CALL    SET_COUNT              ;setup the actual byte count and the absolute VRAM address
                     1626 *               CALL    WRITE_VRAM             ;VRAM data = user buffer
                     1627 *               RET
EOCC C3FD1A          1628                 JP      WRITE_VRAM
                     1629
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1631 ;
                     1632 ;  Name:            GET_VRAM
                     1633 ;
                     1634 ;  Function:        Gets a block of data from VRAM and stores it into a user buffer.
                     1635 ;                   The following table codes are used to identify which VRAM table
                     1636 ;                   is being referenced:
                     1637 ;
                     1638 ;                     0 - SPRITE ATTRIBUTE TABLE  (SAT)
                     1639 ;                     1 - SPRITE GENERATOR TABLE  (SGT)
                     1640 ;                     2 - PATTERN NAME TABLE      (PNT)
                     1641 ;                     3 - PATTERN GENERATOR TABLE (PGT)
                     1642 ;                     4 - PATTERN COLOR TABEL     (PCT)
                     1643 ;
                     1644 ;  Entry:           A - table code (see above)
                     1645 ;                   DE - starting index into the table
                     1646 ;                   HL - address of user buffer
                     1647 ;                   IY - block size (or byte count)
                     1648 ;
                     1649 ;  Exit:            None.
                     1650 ;
                     1651 ;  Registers used:  AF, DE, HL, IY
                     1652 ;
                     1653 ;  Routines used:   SET_COUNT
                     1654 ;
                     1655 ;  Size:            ROM - 0 bytes
                     1656 ;                   RAM - 0 bytes
                     1657 ;
                     1658 ;  Comments:        This version of GET_VRAM is provided for stand alone operation.
                     1659 ;
                     1660 ;  Comparision:     left out pascal entry point
                     1661 ;                   code compacted
                     1662 ;                   edited for documentation
                     1663 ;
EOCF                 1664 __GET_VRAM:
EOCF CDE0D5          1665                CALL    SET_COUNT               ;setup the actual byte count and the absolute VRAM address
                     1666 *              CALL    READ_VRAM               ;user buffer = VRAM data
                     1667 *              RET
EOD2 C3FD1D          1668                JP      READ_VRAM
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1670 ;
                     1671 ; CALLED BY PUT_VRAM_ AND GET_VRAM_
                     1672 ;
                     1673 ; SETS BYTE COUNT AND INDEX FOR WRITES TO AND READS FROM VRAM.
                     1674 ;
                     1675 ;  TABLE          BYTES/ITEM
                     1676 ;  SPRITE_NAME         4
                     1677 ;  SPRITE_GEN          8
                     1678 ;  PATTERN_NAME        1
                     1679 ;  PATTERN_GEN         8
                     1680 ;  COLOR (MODE 1)      1
                     1681 ;  COLOR (MODE 2)      8
                     1682 ;
   EOD5              1683 SET_COUNT
   EOD5 FDE5         1684                PUSH    IY                      ;Save count on stack
   EOD7 4F           1685                LD      C,A                     ;Get table code #
   EOD8 FE04         1686                CP      4                       ;See if color table
   EODA 2008         1687                JR      NZ,NOT_COLOR_TBL_ACCESS ;If not then check for name table
                     1688                                                ;Color Table
   EODC 3AFD61       1689                LD      A,[VDP_MODE_WORD]       ;Get mode word to check graphics type
   EODF E602         1690                AND     02H                     ;Bit 1 indicates mode 1 or 2
   EOE1 2814         1691                JR      Z,ADD_TO_BASE           ;If mode 1 then start index is not modified
   EOE3 79           1692                LD      A,C                     ;Get table code back
                     1693                                                ;Fall through to multiply by 8 for mode 2.
   EOE4              1694 NOT_COLOR_TBL_ACCESS
   EOE4 FE02         1695                CP      2                       ;Check for Name table
   EOE6 280F         1696                JR      Z,ADD_TO_BASE           ;If so then don't modify start index
                     1697                                                ;At this point the table code is:
                     1698                                                ; Sprite attribute table, Sprite gen table,
                     1699                                                ; Pattern gen table, or Color table in Mode 2
                     1700
   EOE8 EB           1701                EX      DE,HL                   ;Get count in HL
   EOE9 29           1702                ADD     HL,HL                   ;*2
                     1703                                                ;
   EOEA 29           1704                ADD     HL,HL                   ;*4
   EOEB B7           1705                OR      A                       ;Check for Sprite attr table
   EOEC 2801         1706                JR      Z,HAVE_CNT              ;If so then Index*4 is complete
   EOEE 29           1707                ADD     HL,HL                   ;If not then Index*8 is required
                     1708                                                ; this is for the pattern gen table,
                     1709                                                ; sprite gen table, or color gen table
                     1710                                                ; in mode 2.
   EOEF              1711 HAVE_CNT:
   EOEF EB           1712                EX      DE,HL                   ;Put updated index in DE
   EOF0 E3           1713                EX      [SP],HL                 ;Get count off stack
                     1714                                                ;Put address of RAM buffer on stack
                     1715
                     1716                                                ;Now modify the count for:
                     1717                                                ; Sprite attribute table, Sprite gen table,
                     1718                                                ; Pattern gen table, or Color table in Mode 2
   EOF1 29           1719                ADD     HL,HL
   EOF2 29           1720                ADD     HL,HL
   EOF3 2801         1721                JR      Z,HAVE_CNT2             ;If Sprite attr table then *4 is enoungh
                     1722                                                ; (the zero flag is still set from earlier)
   EOF5 29           1723                ADD     HL,HL                   ;Count *8 for remaining tables
   EOF6              1724 HAVE_CNT2:
   EOF6 E3           1725                EX      [SP],HL                 ;Save updated count on stack
                     1726                                                ;Restore RAM buffer off stack
```

```
LOCATION OBJECT CODE LINE      SOURCE LINE

    EOF7             1727 ADD_TO_BASE:
    EOF7 79          1728                LD      A,C                     ;Get table code back
    EOF8 01FD64      1729                LD      BC,VRAM_ADDR_TABLE      ;Get top of address table
    EOFB E5          1730                PUSH    HL                      ;Save RAM buffer on stack
    EOFC 2600        1731                LD      H,00H                   ;Put table code into HL
    EOFE 6F          1732                LD      L,A
    EOFF 29          1733                ADD     HL,HL                   ;*2 to offset for addresses in table
    E100 09          1734                ADD     HL,BC                   ;Add to top of table
    E101 7E          1735                LD      A,[HL]                  ;Low byte of VRAM table address
    E102 23          1736                INC     HL
    E103 66          1737                LD      H,[HL]                  ;High byte of VRAM table address
    E104 6F          1738                LD      L,A
    E105 19          1739                ADD     HL,DE                   ;Add offset from start of table
    E106 EB          1740                EX      DE,HL                   ;Absolute VRAM address in DE
    E107 E1          1741                POP     HL                      ;Restore RAM buffer address
    E108 C1          1742                POP     BC                      ;Restore Count into BC
                     1743                                                ;Registers are now ready for
                     1744                                                ; a call to the basic READ/WRITE VRAM routines
    E109 C9          1745                RET                             ;Exit
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1747 ************** CALC.OFFSET ************************************************
                     1748 ;
                     1749 ;DESCRIPTION:    THIS ROUTINE CALCULATES THE PROPER OFFSET INTO THE NAME TABLE
                     1750 ;                 FOR THE PATTERN POSITION GIVEN BY X_PAT_POS, Y_PAT_POS.  THE
                     1751 ;                 FORMULA USED IS: OFFSET = 32*Y_PAT_POS + X_PAT_POS
                     1752 ;
                     1753 ;INPUT:          D = Y_PAT_POS
                     1754 ;                 E = X_PAT_POS
                     1755 ;
                     1756 ;OUTPUT:         DE = OFFSET
                     1757 ;
                     1758 ********************************************************************************
                     1759 ;
E10A                 1760 __CALC_OFFSET:
E10A E5              1761                 PUSH    HL                      ;save HL
E10B CB7A            1762                 BIT     7,D                     ;if Y_PAT_POS is minus
E10D 2804            1763                 JR      Z,ELSE_11
E10F 26FF            1764                 LD      H,OFFH                  ;  extend negative sign
E111 1802            1765                 JR      END_IF_11
E113                 1766 ELSE_11:                                        ;else
E113 2600            1767                 LD      H,0                     ;  extend positive sign
E115                 1768 END_IF_11:                                      ;endif
                     1769
E115 6A              1770                 LD      L,D                     ;OFFSET = 32*Y_PAT_POS + X_PAT_POS
E116 29              1771                 ADD     HL,HL                   ;HL=2*Y_PAT_POS
E117 29              1772                 ADD     HL,HL                   ;   4*    "
E118 29              1773                 ADD     HL,HL                   ;   8*    "
E119 29              1774                 ADD     HL,HL                   ;  16*    "
E11A 29              1775                 ADD     HL,HL                   ;  32*    "
                     1776
E11B CB7B            1777                 BIT     7,E                     ;if X_PAT_POS is minus
E11D 2804            1778                 JR      Z,ELSE_12
E11F 16FF            1779                 LD      D,OFFH                  ;  extend negative sign
E121 1802            1780                 JR      END_IF_12
E123                 1781 ELSE_12:                                        ;else
E123 1600            1782                 LD      D,0                     ;  extend positive sign
E125                 1783 END_IF_12:                                      ;endif
                     1784
E125 19              1785                 ADD     HL,DE                   ;HL=32*Y_PAT_POS + X_PAT_POS
E126 EB              1786                 EX      DE,HL
E127 E1              1787                 POP     HL                      ;restore HL intact
E128 C9              1788                 RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1790 ******************** PX.TO.PTRN.POS ************************************
                     1791 ;DESCRIPTION:    DIVIDES REG DE BY 8, IF SIGNED RESULT > 127 THEN E := MAX SIGNED
                     1792 ;                POSITIVE NUMBER. IF RESULT < -128, THEN E := MIN NEGATIVE NUM
                     1793 ;INPUT:           DE = 16 BIT SIGNED NUMBER
                     1794 ;OUTPUT:          DE/8 < -128      E = -128
                     1795 ;         -128 <= DE/8 <=+127      E = DE/8
                     1796 ;         +127 <  DE/8             E = +127
                     1797 ******************************************************************************
                     1798 ;
    E129             1799 __PX_TO_PTRN_POS:
                     1800
    E129 E5          1801          PUSH  HL                  ;HL USED TO TEST MAGNITUDE
                     1802 ;
    E12A C5          1803          PUSH  BC                  ;B WILL BE USED AS A COUNTER IN THE DIVIDE
    E12B 0603        1804          LD    B,3
    E12D             1805 PX_2_P_P_1:                        ;(THIS IS THE START OF THE DIVIDE LOOP)
    E12D CB2A        1806          SRA   D                   ;16 BIT SHIFT LEFT
    E12F CB1B        1807          RR    E
    E131 10FA        1808          DJNZ  PX_2_P_P_1          ;DIVIDE BY 8 LOOP
    E133 C1          1809          POP   BC                  ;RESTORE BC TO OLD VALUE
                     1810 ;
    E134 21FF80      1811          LD    HL,OFF80H           ;SET HL TO -128
    E137 CB7A        1812          BIT   7,D                 ;IS RESULT OF DIVIDE NEGATIVE?
    E139 2006        1813          JR    NZ,NEGTV
                     1814 ;
    E13B 19          1815          ADD   HL,DE               ;IF DE-128 < 0 (Fnc) RETURN
    E13C E1          1816          POP   HL                  ;same as (IF DE<=127 RETURN)
    E13D D0          1817          RET   NC
    E13E 1E7F        1818          LD    E,7FH               ;IF > 128, THEN E := MAX SIGNED + NUM.
    E140 C9          1819          RET
                     1820 ;
    E141 2600        1821 NEGTV:   LD    H,00H               ;SET HL TO 128
                     1822 ;
    E143 19          1823          ADD   HL,DE               ;IF DE+128 >= 0 RETURN
    E144 E1          1824          POP   HL                  ;same as (IF DE>=-128 RETURN)
    E145 D8          1825          RET   C
    E146 1E80        1826          LD    E,80H               ;IF < -128, THE E := MIN SIGNED - NUM.
    E148 C9          1827          RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

            <007F>  1829 DEL             EQU     7FH             ;ASCII code for it
                    1830 ;ASCII_GEN       EQU     LOC_IN_ALPHA    ;1st byte of generator for NUL
                    1831 ;
                    1832 ; __LOAD_ASCII
                    1833 ;
                    1834 ;    Loads the ASCII character generators into VRAM at the
                    1835 ;    current pattern generator table (NOTE - INIT_TABLE must
                    1836 ;    have been called prior to this).  Only the characters
                    1837 ;    from the space to the DEL (20h to 7Fh) are loaded.
                    1838 ;
                    1839 ;Input Parameters:
                    1840 ;   NONE
                    1841 ;
                    1842 ;Returns:
                    1843 ;   A, BC, HL, DE are destroyed
                    1844 ;
                    1845 ;  Falls into __PUT_ASCII
                    1846 ;
    E149            1847 __LOAD_ASCII:
                    1848                 ;LD      HL,[PATTRNGENTBL] ;get base of current generator table
    E149 ED5BFD6A   1849                 LD      DE,[PATTRNGENTBL] ;get base of current generator table
                    1850                 ;LD      DE,00H+8         ;offset into it where SPACE will be
                    1851                 ;ADD     HL,DE
                    1852                 ;EX      DE,HL            ;leave VRAM address in DE
    E14D 210000     1853                 LD      HL,00H           ;want to start with the null
    E150 010080     1854                 LD      BC,(DEL-00H+1)   ;load the entire ASCII set (0..7F)
                    1855 ;
                    1856 ; fall thru to PUT_ASCII
                    1857 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1859 ;
                     1860 ;  __PUT_ASCII
                     1861 ;
                     1862 ;      Copys a specified number of ASCII character generators into VRAM.
                     1863 ;      Swaps to ROM that contains the generators then swaps back, keeps
                     1864 ;      track of callers stack and uses a local stack that is known to
                     1865 ;      be in RAM when the ROM is swapped in.
                     1866 ;
                     1867 ;Input Parameters:
                     1868 ;  HL - Character to 1st load (generally in range 0..FFH)
                     1869 ;  BC - Number of characters to load (Not the number of bytes)
                     1870 ;  DE - Address in VRAM to load the 1st generator
                     1871 ;
                     1872 ;  A, BC, HL, DE, IX  are destroyed
                     1873 ;
                     1874 ;  Calls WRITE_VRAM
                     1875 ;        SWITCH_MEM
                     1876 ;
E153                 1877 __PUT_ASCII:
E153 29              1878                 ADD     HL,HL           ;multiply char to start loading by 8
E154 29              1879                 ADD     HL,HL           ;so can use it as an offset into the
E155 29              1880                 ADD     HL,HL           ;ASCII generator
E156 C5              1881                 PUSH    BC              ;save number of chars desired
                     1882                 ;LD      BC,ASCII_GEN
                     1883                 ;LD      BC,[LOC_IN_ALPHA]
                     1884                 ;ADD     HL,BC           ;now HL points to the 1st generator desired
E157 E3              1885                 EX      [SP],HL         ;get number of chars desired
E158 29              1886                 ADD     HL,HL           ;and multiply by 8
E159 29              1887                 ADD     HL,HL           ;  since each generator is 8 bytes
E15A 29              1888                 ADD     HL,HL           ;     long
E15B E3              1889                 EX      [SP],HL         ;get back pointer into ASCII generators
E15C C1              1890                 POP     BC              ;get number of chars times 8
                     1891
E15D DD210000        1892                 LD      IX,0000H        ;  to use to save callers SP
E161 DD39            1893                 ADD     IX,SP
E163 31FE6E          1894                 LD      SP,TEMP_STACK   ;set stack into known RAM
E166 DDE5            1895                 PUSH    IX              ;save callers SP
                     1896
E168 3AFD6E          1897                 LD      A,[CUR_BANK]    ;remember which bank setting we
E16B F5              1898                 PUSH    AF              ;   are in
E16C 3AFC17          1899                 LD      A,[MEM_CNFGOO]  ;get value to let us get at the ALPHA roms
                     1900
E16F C5              1901                 PUSH    BC              ;count would get destroyed by SWITCH_MEM
E170 CDFD14          1902                 CALL    SWITCH_MEM
E173 C1              1903                 POP     BC
                     1904
E174 D5              1905                 PUSH    DE
E175 ED5B0102        1906                 LD      DE,[LOC_IN_ALPHA]
E179 19              1907                 ADD     HL,DE
E17A D1              1908                 POP     DE
                     1909
E17B CDFD1A          1910                 CALL    WRITE_VRAM
                     1911
E17E F1              1912                 POP     AF              ;get back callers mem configuration
E17F CDFD14          1913                 CALL    SWITCH_MEM
                     1914
E182 E1              1915                 POP     HL              ;get callers SP
```

```
LOCATION OBJECT CODE LINE      SOURCE LINE

    E183 F9          1916                  LD      SP,HL           ;and restore it
    E184 C9          1917                  RET                     ;  and back to caller
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1919 ;
                     1920 ;   Switch_mem: memory bank switching routine
                     1921 ;
                     1922 ;   Inputs:
                     1923 ;                A = Appropriate input from SWITCH_TABLE
                     1924 ;
                     1925 ;                Example:
                     1926 ;   Switch Table Offset   Memory configuration
                     1927 ;   ------------------------------------------------------------
                     1928 ;
                     1929 ;         0 -          Boot ROM and Alpha Roms
                     1930 ;         1 -          Intrinsic RAM (0000H thru 7FFFH)
                     1931 ;         2 -          Expansion memory (0000H thru 7FFFH)
                     1932 ;         3 -          OS_7 and Intrinsic RAM (2000H thru 7FFFH)
                     1933 ;         4
                     1934 ;         .
                     1935 ;         D
                     1936 ;         E
                     1937 ;         F
                     1938 ;
                     1939 ;   ***Note: The values in the table are subject to change in future
                     1940 ;            releases of ADAM hardware.
                     1941 ;
                     1942 ;   For an example of the use of SWITCH_MEM see PORT_COLLECTION below.
                     1943 ;
                     1944 ;
                     1945 ;    Outputs:
                     1946 ;                CUR_BANK, a defined memory location will contain
                     1947 ;                the updated input parameter for applications and
                     1948 ;                EOS routines which need to determine which
                     1949 ;                memory space is currently active.
                     1950 ;
E185                 1951 __SWITCH_MEM
E185 47              1952          LD      B,A                     ;SAVE THE PORT DATA TO WRITE
E186 3AFC27          1953          LD      A,[MEM_SWITCH_PORT]     ;GET THE PORT NUMBER
E189 4F              1954          LD      C,A
E18A ED41            1955          OUT     [C],B                   ;SWITCH MEMORY BANKS
E18C 78              1956          LD      A,B
E18D 32FD6E          1957          LD      [CUR_BANK],A            ;REMEMBER WHAT WAS SWITCHED TO.
E190 C9              1958          RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1960 ;
                     1961 ;    Port collection must be called during initialization
                     1962 ;    It switches to OS7 ROM grabs the ports and switches back
                     1963 ;
                     1964
    E191             1965 __PORT_COLLECTION:
    E191 3AFD6E      1966          LD      A,[CUR_BANK]            ;GET THE CURRENT BANK
    E194 F5          1967          PUSH    AF
                     1968 ;
    E195 3AFC1A      1969          LD      A,[MEM_CNFGO3]          ;SWAP IN THE OS
    E198 CDFD14      1970          CALL    SWITCH_MEM
    E19B 21FC29      1971          LD      HL,PORT_TABLE+2         ;HL POINTS TO THE PORT TABLE
                     1972 ;
    E19E 3A1D43      1973          LD      A,[O1D43H]              ;VDP CONTROL PORT
    E1A1 77          1974          LD      [HL],A
    E1A2 23          1975          INC     HL
                     1976 ;
    E1A3 3A1D47      1977          LD      A,[O1D47H]              ;VDP DATA PORT
    E1A6 77          1978          LD      [HL],A
    E1A7 23          1979          INC     HL
                     1980 ;
    E1A8 3A114B      1981          LD      A,[O114BH]              ;CONTROLLER O
    E1AB 77          1982          LD      [HL],A
    E1AC 23          1983          INC     HL
                     1984 ;
    E1AD 3A1151      1985          LD      A,[O1151H]              ;CONTROLLER 1
    E1BO 77          1986          LD      [HL],A
    E1B1 23          1987          INC     HL
                     1988 ;
    E1B2 3A1157      1989          LD      A,[O1157H]              ;CONTROLLER STROBE SET
    E1B5 77          1990          LD      [HL],A
    E1B6 23          1991          INC     HL
                     1992 ;
    E1B7 3A1168      1993          LD      A,[O1168H]              ;CONTROLLER STROBE RESET
    E1BA 77          1994          LD      [HL],A
    E1BB 23          1995          INC     HL
                     1996 ;
    E1BC 3AO18E      1997          LD      A,[O018EH]              ;SOUND PORT
    E1BF 77          1998          LD      [HL],A
    E1CO F1          1999          POP     AF
    E1C1 CDFD14      2000          CALL    SWITCH_MEM
    E1C4 C9          2001          RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2003 ;
                     2004 ;  Name:            WR_SPR_ATTRIBUTE
                     2005 ;
                     2006 ;  Function:        Transfers the local copy of the sprite attribute table
                     2007 ;                    to VRAM.  WR_SPR_ATTRIBUTE will also reorder the
                     2008 ;                    sprite table to reduce 5th sprite priority problems.
                     2009 ;
                     2010 ;  Entry:           DE - address of Local Sprite Table
                     2011 ;                   HL - address of Priority table
                     2012 ;                    A - number of sprites to be transfered (length of Local Sprite
                     2013 ;                        table divided by 4
                     2014 ;  Exit:            None.
                     2015 ;
                     2016 ;  Registers used: AF, BC, DE, HL,
                     2017 ;
                     2018 ;  Data structures accessed:
                     2019 ;                   VRAM_ADDR_TABLE (SPRITEATTRTBL) for the starting address
                     2020 ;                   of the sprite attribute table in VRAM.
                     2021 ;
                     2022 ;                   PORT_PTR_TABLE (CTRL_PORT_PTR, DATA_PORT_PTR) for
                     2023 ;                   VDP port access.
                     2024 ;
                     2025 ;                   Local Sprite Table - User defined memory image of
                     2026 ;                   the sprite attribute table
                     2027 ;
                     2028 ;                   Priority Table - User defined sprite priority list
                     2029 ;                   for the transfer of the sprite attributes
                     2030 ;
                     2031 ;  Size:            ROM - xx bytes
                     2032 ;                   RAM - xx bytes
                     2033 ;
                     2034 ;  Comments:
                     2035 ;
                     2036 ;  Comparision:     left out pascal entry point
                     2037 ;                   added register parameters in place of defined memory pointers
                     2038 ;                   code compacted
                     2039 ;                   edited for documentation
                     2040 ;
E1C5                 2041 __WR_SPR_ATTRIBUTE
                     2042                                              ;Initialize VDP address pointer
E1C5 F5              2043                PUSH    AF
E1C6 E5              2044                PUSH    HL
E1C7 2AFD64          2045                LD      HL,[SPRITEATTRTBL]
E1CA CDE1E9          2046                CALL    SET_WRITE
E1CD E1              2047                POP  HL
E1CE F1              2048                POP  AF
E1CF 47              2049                LD   B,A
                     2050                                              ;B = Sprite Count
                     2051                                              ;C = port address
                     2052                                              ;DE = address of Local Sprite Attribute Table
                     2053                                              ;HL = address of Priority table
                     2054                                              ;VDP is initialized to appropriate address
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E1D0             2056 LOOP_EVERY_SPRITE:
                     2057
                     2058 *              Index to sprite with the next highest priority
                     2059
    E1D0 7E          2060            LD   A,[HL]          ;Get current sprite as defined by the priority table
    E1D1 87          2061            ADD  A,A             ;*2 Multiply by an attribute entry (4)
    E1D2 87          2062            ADD  A,A             ;*4
                     2063
    E1D3 23          2064            INC  HL              ;Index to next location in priority table
    E1D4 E5          2065            PUSH HL              ;Save for next sprite
                     2066
    E1D5 6F          2067            LD   L,A             ;Offset into the Local Sprite Table by the current priority
    E1D6 2600        2068            LD   H,0
    E1D8 19          2069            ADD  HL,DE
                     2070
                     2071
    E1D9 78          2072            LD   A,B             ;Save sprite count in Acc.
    E1DA 0604        2073            LD   B,4             ;Set count for write to VRAM
                     2074                                 ;C has port address
                     2075                                 ;HL has the pointer to a sprite's attributes
    E1DC             2076 LOOP_EVERY_BYTE
    E1DC EDA3        2077            OUTI                 ;Output a byte
    E1DE 00          2078            NOP                  ;Wait for VDP to catch up (worst case)
    E1DF 00          2079            NOP
    E1E0 20FA        2080            JR   NZ,LOOP_EVERY_BYTE
                     2081
    E1E2 47          2082            LD   B,A             ;Restore sprite count to B register
    E1E3 E1          2083            POP  HL              ;Restore index into the priority table
    E1E4 10EA        2084            DJNZ LOOP_EVERY_SPRITE ; If more sprites left then loop back
                     2085
    E1E6 C9          2086            RET                  ;Else exit
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2088 ;
                     2089 ;  Name:            SET_READ / SET_WRITE
                     2090 ;
                     2091 ;  Function:        Initiates the VDP for reads or writes.
                     2092 ;
                     2093 ;  Entry:           HL - VRAM address to start read / writes
                     2094 ;
                     2095 ;  Exit:            C register contains the VDP_DATA_PORT value
                     2096 ;                   HL remains unchanged
                     2097 ;
                     2098 ;  Registers used: AF, BC, HL,
                     2099 ;
                     2100 ;  Data structures accessed:
                     2101 ;
                     2102 ;                   PORT_TABLE (CTRL_PORT_PTR, DATA_PORT_PTR) for
                     2103 ;                   VDP port access.
                     2104 ;
                     2105 ;  Size:            ROM - xx bytes
                     2106 ;                   RAM - xx bytes
                     2107 ;
                     2108 ;  Comments:
                     2109 ;
                     2110 ;  Comparision:
                     2111 ;
E1E7                 2112 SET_READ:
E1E7 AF              2113            XOR  A              ;Clear accumulator for later OR which
                     2114                                ; determines a read or write
                     2115
E1E8 C2              2116            DEFB 0C2H           ;This is the beginning of a JP NZ which
                     2117                                ; will never be true but leaves the
                     2118                                ; accumulator intact over the next
                     2119                                ; two bytes ( the LD A,40H is decoded
                     2120                                ; as the last two bytes of the jump
E1E9                 2121 SET_WRITE:
E1E9 3E40            2122            LD   A,40H          ;Set bit to tell VDP that write is requested
                     2123
E1EB ED4BFC29        2124            LD   BC,[VDP_CTRL_PORT] ;Get BOTH port addresses
E1EF ED69            2125            OUT  [C],L          ;Output low 8 bits of address
E1F1 B4              2126            OR   H              ;OR read/write bit (Bit 6) into the high 8 bytes
E1F2 ED79            2127            OUT  [C],A          ;Output result
E1F4 48              2128            LD   C,B            ;Put DATA_PORT value into the C register
                     2129
E1F5 C9              2130            RET                 ;Exit to caller
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2132 ;CONTROLLER SOFTWARE
                     2133 ;
                     2134 ******************************DATA*****************************
                     2135 *
                     2136 *  DECODER TABLE FOR THE KEYBOARD
                     2137
                     2138 *
    E1F6             2139 DEC_KBD_TBL
    E1F6 OF          2140            DEFB KBD_NULL        ; NULL ENTRY
    E1F7 06          2141            DEFB 6               ; '6'
    E1F8 01          2142            DEFB 1               ; '1'
    E1F9 03          2143            DEFB 3               ; '3'
    E1FA 09          2144            DEFB 9               ; '9'
    E1FB 00          2145            DEFB 0               ; '0'
    E1FC 0A          2146            DEFB 10              ; '*'
    E1FD 0C          2147            DEFB 12              ; Purple Action Controller key (third one down)
    E1FE 02          2148            DEFB 2               ; '2'
    E1FF 0B          2149            DEFB 11              ; '#'
    E200 07          2150            DEFB 7               ; '7'
    E201 0D          2151            DEFB 13              ; Blue Action Controller key (fourth (or last) one down
    E202 05          2152            DEFB 5               ; '5'
    E203 04          2153            DEFB 4               ; '4'
    E204 08          2154            DEFB 8               ; '8'
    E205 OF          2155            DEFB KBD_NULL        ; NULL ENTRY
                     2156
          <000F>     2157 KBD_NULL       EQU          OFH
          <000F>     2158 KBD_MASK       EQU          OFH      ;MASK FOR INPUT DATA BYTE
          <0040>     2159 FIRE_MASK      EQU          40H
          <0040>     2160 ARM_MASK       EQU          40H
          <000F>     2161 JOY_MASK       EQU          OFH
                     2162
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2164 ;
                     2165 ;__DECODER
                     2166 ;          Will read all the data, from both segments, for a given controller
                     2167 ;          All data is returned in registers
                     2168 ;          Segment 0 data is returned the same as with OS-7
                     2169 ;
                     2170 ;    NOTE - some operations are done in an odd order so that a certain
                     2171 ;           number of cycles (T states) pass between port addressing.
                     2172 ;           Turns off interupts if they were on at entry time for
                     2173 ;           195 T states (approx. 54.5 micro secs), the iterupts
                     2174 ;           are reenabled after this time if they were on before.
                     2175 ;
                     2176 ;    Callers  -  The input parameters are different than OS-7 and
                     2177 ;                both segments are returned so only one call is required
                     2178 ;                for all the data from a given controller.
                     2179 ;                The values returned are the same for segment 0 (H, L, E)
                     2180 ;                but segment 1 values are returned in B and D.
                     2181 ;
                     2182 ;Stack Usage:
                     2183 ;  Uses 2 words (pushes only)
                     2184 ;
                     2185 ;Input Parameters:
                     2186 ;    A  - controller number, 0 or 1
                     2187 ;
                     2188 ;Returns:
                     2189 ;    H  - fire button data
                     2190 ;    L  - joystick data
                     2191 ;    B  - arm button data
                     2192 ;    C  - raw data for segment 0, (joystick, fire)
                     2193 ;    D  - keyboard data
                     2194 ;    E  - spinner count data
                     2195 ;    A  - raw data for segment 1, (keyboard, arm, super controller buttons)
                     2196 ;
                     2197 ;    SPIN_SWx_CT  - reset to 0 after read
                     2198 ;
                     2199 ;    strobe port left in the reset condition
                     2200 ;
                     2201 ;Date:
                     2202 ; 9/24/83    11:14     JKL
                     2203 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E206             2205 __DECODER:
    E206 4F          2206         LD       C,A                    ;Save controller #
    E207 ED57        2207         LD       A,I                    ;get current state of interupts, on/off, into P/V flag
    E209 F5          2208         PUSH     AF                     ;and save for later so can turn them back on if needed
    E20A 79          2210         LD       A,C                    ;Restore controller #
    E20C ED4BFC2D    2211         LD       BC,[STROBE_SET_PORT]   ;C<--SET PORT ,B<--RST PORT
    E210 ED79        2212         OUT      [C],A                  ;STROBE TO SEGMENT 1 (SET)
    E212 50          2213         LD       D,B                    ;SAVE RESET PORT
    E213 21FE58      2214         LD       HL,SPIN_SWO_CT         ;point at spinner count for controller 0
    E216 ED4BFC2B    2215         LD       BC,[CONTROLLER_O_PORT] ;C<--PORT_0, B<--PORT_1
    E21A B7          2216         OR       A                      ;A contains controller number
    E21B 2002        2217         JR       NZ,GOT_CORRECT_CONTROLLER_IN_C ;
    E21D 23          2218         INC      HL                     ;Point to spinner for ctrlr 1
    E21E 48          2219         LD       C,B                    ;C<--PORT_1
    E21F             2220 GOT_CORRECT_CONTROLLER_IN_C:
    E21F AF          2221         XOR      A
    E220 5E          2222         LD       E,[HL]                 ;HAVE SPIN SWITCH COUNT
    E221 77          2223         LD       [HL],A                 ;RESET COUNT
    E222 42          2224         LD       B,D                    ;B<--RST PORT
    E223 ED78        2225         IN       A,[C]                  ;READ DATA FROM SEG 1
    E225 2F          2226         CPL
                     2227 ;
    E226 2600        2228         LD       H,0
    E228 6F          2229         LD       L,A                    ;HL<--offset into DEC_KBD_TBL
    E229 79          2230         LD       A,C                    ;swap RST_PORT/DATA_PORT
    E22A 48          2231         LD       C,B
    E22B ED79        2232         OUT      [C],A                  ;STOBE TO SEG_0 (reset)
    E22D 4F          2233         LD       C,A                    ;C<--DATA PORT
    E22E F1          2234         POP      AF                     ;get back entry state of interupt reg
    E22F E2E233      2235         JP       PO,LEAVE_THEM_OFF
    E232 FB          2236         EI                              ;turn interupts back on only if they were on entry
    E233             2237 LEAVE_THEM_OFF:
    E233 7D          2238         LD       A,L
    E234 F5          2239         PUSH     AF                     ;save raw data  seg_1
    E235 E640        2240         AND      ARM_MASK               ;Test sleeve
    E237 47          2241         LD       B,A                    ;HAVE ARM DATA   (in B)
    E238 7D          2242         LD       A,L
    E239 E60F        2243         AND      KBD_MASK               ;isolate KBD data
    E23B 6F          2244         LD       L,A
    E23C D5          2245         PUSH     DE                     ;Save spin switch count on stack (reg E)
    E23D 11E1F6      2246         LD       DE,DEC_KBD_TBL         ;Get top of keypad decoder table
    E240 19          2247         ADD      HL,DE                  ;point to DEC_KBD_TBL entry
    E241 D1          2248         POP      DE                     ;Restore Spinner count (reg E)
    E242 56          2249         LD       D,[HL]                 ;HAVE KEYBOARD DATA
    E243 ED78        2250         IN       A,[C]                  ;segment 0 data
    E245 2F          2251         CPL
                     2252 ;
    E246 F5          2253         PUSH     AF                     ;save raw data
    E247 6F          2254         LD       L,A
    E248 E640        2255         AND      FIRE_MASK              ;isolate fire data
    E24A 67          2256         LD       H,A                    ;HAVE FIRE_DATA
    E24B 7D          2257         LD       A,L
    E24C E60F        2258         AND      JOY_MASK               ;isolate joystick data
    E24E 6F          2259         LD       L,A                    ;HAVE JOY DATA
    E24F F1          2260         POP      AF                     ;get back raw data for seg 0
    E250 4F          2261         LD       C,A                    ;C = raw data segment 0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E251 F1          2262       POP     AF                    ;A = raw data segment 1
    E252 C9          2263       RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    2265 ;
                    2266 ;__POLLER
                    2267 ;
                    2268 ;   Used to fill a table of values for the 2 controllers
                    2269 ;   Does some debouncing if called twice in succession
                    2270 ;
                    2271 ;   NOTE - Does not return at bottom of code, return is in middle
                    2272 ;          Interupts will be off for a while when call to DECODE is performed.
                    2273 ;
                    2274 ;   Callers - This has different input parameters than OS-7 and
                    2275 ;             the controller map is ordered differently.
                    2276 ;
                    2277 ;
                    2278 ;Stack Usage:
                    2279 ;  will use 4 words (2 pushes, 2 call)
                    2280 ;
                    2281 ;Calls:
                    2282 ;  __DECODER
                    2283 ;  DEBOUNCE
                    2284 ;  READ_N_DEBOUNCE  (a routine that is nested in __POLLER)
                    2285 ;
                    2286 ;Input Parameters:
                    2287 ;  IX - pointer to 1st byte of users controller map (10 bytes)
                    2288 ;   A - controller enabled and spinner enable
                    2289 ;         bit 0 - set if want controller 0 enabled
                    2290 ;         bit 1 - set if want controller 1 enabled
                    2291 ;         bit 7 - set if want spinner enabled for controllers which are enabled
                    2292 ;
                    2293 ;Returns:
                    2294 ;
                    2295 ;  Data in users controller map is like this off of entry IX value
                    2296 ;           IX+0  - joystick 0
                    2297 ;           IX+1  - fire 0
                    2298 ;           IX+2  - arm 0
                    2299 ;           IX+3  - keyboard 0
                    2300 ;           IX+4  - spinner count 0
                    2301 ;           IX+5  - joystick 1
                    2302 ;           IX+6  - fire 1
                    2303 ;           IX+7  - arm 1
                    2304 ;           IX+8  - keyboard 1
                    2305 ;           IX+9  - spinner count 1
                    2306 ;
                    2307 ;  IX - points to 1 byte past users data table of highest enabled controller
                    2308 ;
                    2309 ;
                    2310 ;  all but IY - destroyed
                    2311 ;
                    2312 ;Date:
                    2313 ; 9/24/83   11:14    JKL
                    2314 ;
                    2315
<0001>              2316 CONTROLLER_0_MASK       EQU   01H
<0001>              2317 CONTROLLER_1_BIT        EQU   1
<0007>              2318 SPINNER_ENABLE_BIT      EQU   7
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E253             2320 __POLLER:
    E253 21FE5A      2321                LD      HL,PERSONAL_DEBOUNCE_TABLE      ;point to controller 0 debounce buffer
    E256 4F          2322                LD      C,A                             ;save enabled controller input
    E257 E601        2323                AND     CONTROLLER_0_MASK               ;see if controller 0 is enabled
    E259 2823        2324                JR      Z,NOT_CONTROLLER_0              ;if not then check to see if controller 1 is
    E25B 3D          2325                DEC     A                               ;to 0 for parameter to __DECODER
    E25C CDE264      2326                CALL    READ_N_DEBOUNCE                 ;get the data and debounce it
    E25F CB49        2327                BIT     CONTROLLER_1_BIT,C              ;see if controller 1 is active
    E261 C8          2328                RET     Z                               ;return if not
    E262             2329 POLL_CONTROLLER_1:
    E262 3E01        2330                LD      A,01H                           ;get data for controller 1
    E264             2331 READ_N_DEBOUNCE:
    E264 C5          2332                PUSH    BC                              ;save input parameter
    E265 E5          2333                PUSH    HL                              ;save debounce buffer ptr
    E266 CDE206      2334                CALL    __DECODER
    E269 4C          2335                LD      C,H                             ;copy over raw data
    E26A 7D          2336                LD      A,L                             ;  with fire and joystick data
    E26B E1          2337                POP     HL                              ;get pointer to debounce buffer
    E26C CDE28B      2338                CALL    DEBOUNCE                        ;debounce the data (on second call)
    E26F C1          2339                POP     BC                              ;get back input parameter
    E270 CB79        2340                BIT     SPINNER_ENABLE_BIT,C            ;see if spinners are enabled
    E272 2807        2341                JR      Z,NEXT_IX
    E274 7B          2342                LD      A,E                             ;if so add
    E275 DD8600      2343                ADD     A,[IX+0]                        ;     new data to that in
    E278 DD7700      2344                LD      [IX+0],A                        ;                   users buffer
    E27B DD23        2345 NEXT_IX:       INC     IX                              ;point to next users buffer
    E27D C9          2346                RET
                     2347
    E27E CB49        2349                BIT     CONTROLLER_1_BIT,C              ;see if controller 1 is enabled
    E280 C8          2350                RET     Z                               ;return if not
    E281 110004      2351                LD      DE,0004H
    E284 19          2352                ADD     HL,DE                           ;point to next debounce buffer
    E285 13          2353                INC     DE
    E286 DD19        2354                ADD     IX,DE                           ;point to next user buffer
    E288 C3E262      2355                JP      POLL_CONTROLLER_1
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    2357 ;
                    2358 ;DEBOUNCE
                    2359 ;        This will check for equal data on two passes, and if
                    2360 ;        the data is the same then the users buffer is updated.
                    2361 ;
                    2362 ;        This is done by check for equal data, if the data is
                    2363 ;        different on each call then the data on this call is
                    2364 ;        stored so that the next call can use this new data.
                    2365 ;        If the data is equal then the data is stored with the
                    2366 ;        hi bit (bit 7) set so that the next call will yield
                    2367 ;        different data (IE there could never be a match on the
                    2368 ;        next call).
                    2369 ;
                    2370 ;Stack Usage:
                    2371 ;  1 word (subroutine call)
                    2372 ;
                    2373 ;Input Parameters:
                    2374 ;  HL - Pointer to debounce buffer
                    2375 ;  IX - Pointer to users controller map
                    2376 ;  A  - joystick data
                    2377 ;  B  - arm data
                    2378 ;  C  - fire data
                    2379 ;  D  - keyboard data
                    2380 ;
                    2381 ;Returns:
                    2382 ;  HL  - Pointing to next buffer (entry value + 4)
                    2383 ;  IX  - Pointing to next data area (entry value + 4)
                    2384 ;  A is changed
                    2385 ;  No others are modified
                    2386 ;
                    2387 ;  Calls self as a nested subroutine CHECK
                    2388 ;
                    2389 ;Date:
                    2390 ; 9/24/83   11:14    JKL
                    2391 ;
                    2392 ;
E28B                2393 DEBOUNCE:
E28B CDE297         2394        CALL    CHECK           ;debounce joystick data
E28E 79             2395        LD      A,C             ;debounce fire data
E28F CDE297         2396        CALL    CHECK
E292 78             2397        LD      A,B             ;debounce arm data
E293 CDE297         2398        CALL    CHECK
E296 7A             2399        LD      A,D             ;debounce keyboard data
                    2400                                ;fall thru to check (return implied)
E297                2401 CHECK
E297 BE             2402        CP      [HL]            ;see if data was the same as last time
E298 2005           2403        JR      NZ,NOT_SAME
E29D F680           2405        OR      80H             ;set hi_order bit if same, so not equal on next call
E29F                2406 NOT_SAME
E29F 77             2407        LD      [HL],A          ;update local debounce buffer
E2A0 DD23           2408        INC     IX              ;point to next buffer location
E2A2 23             2409        INC     HL              ;       in debounce buffer as well
E2A3 C9             2410        RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2412 ;
                     2413 ;Update_Spinner - Controller spin switch interrupt service routine
                     2414 ;
                     2415 ;  This routine processes the spinner switch interrupt and updates
                     2416 ;    the data needed by both DECODER and POLLER
                     2417 ;
                     2418 ; Ram area used: Updates SPIN_SW0_CNT and SPIN_SW1_CNT
                     2419 ;
                     2420 ; The spinner switch maskable interrupt is RST 38H
                     2421 ;
                     2422 ; Destroys AF, BC, HL
                     2423 ;
           <0020>    2424 DIR_MASK       EQU     00100000B           ;bit 5 tells us the direction
           <0004>    2425 INT_BIT        EQU     4                   ;bit that tells which spinner interrupted
                     2426
    E2A4             2427 __UPDATE_SPINNER
    E2A4 ED4BFC2B    2428         LD      BC,[CONTROLLER_0_PORT]      ;Get port values for both controllers
    E2A8 ED78        2429         IN      A,[C]                       ;Input data from controller 0
    E2AA 21FE58      2430         LD      HL,SPIN_SW0_CT              ;Point to spinner update data
    E2AD CB67        2431         BIT     INT_BIT,A                   ;Was this the controller that interrupted ?
    E2AF 2007        2432         JR      NZ,GET_CONTROLLER_1         ;If not then check controller 1
    E2B1 E620        2433         AND     DIR_MASK                    ;Check direction bit
    E2B3 2002        2434         JR      NZ,INCREMENT_0_DIRECTION    ;If set then increment direction
    E2B5 35          2435         DEC     [HL]                        ;Else decrement direction
    E2B6 35          2436         DEC     [HL]                        ;Decrement once more to fall through increment
                     2437
    E2B7             2438 INCREMENT_0_DIRECTION:
    E2B7 34          2439         INC     [HL]                        ;Increment direction
                     2440
    E2B8             2441 GET_CONTROLLER_1
    E2B8 48          2442         LD      C,B                         ;Get controller 1 port into C
    E2B9 ED78        2443         IN      A,[C]                       ;Get controller 1 data
    E2BB 23          2444         INC     HL                          ;Point HL to controller 1 update area
    E2BC CB67        2445         BIT     INT_BIT,A                   ;Is this the controller that interrupted me?
    E2BE C0          2446         RET     NZ                          ;Return if not, Also don't assume
                     2447                                             ; either controller if no bit is set
    E2BF E620        2448         AND     DIR_MASK                    ;Check direction bit
    E2C1 2002        2449         JR      NZ,INCREMENT_1_DIRECTION    ;If set then go increment direction
    E2C3 35          2450         DEC     [HL]                        ;Else decrement direction
    E2C4 C9          2451         RET                                 ;Then return
                     2452
    E2C5             2453 INCREMENT_1_DIRECTION
    E2C5 34          2454         INC     [HL]                        ;Increment direction
    E2C6 C9          2455         RET                                 ;Exit
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2458
                     2459 ;*****************************************************************************
                     2460 ;
                     2461 ;                        NOTES TO SOUND PROGRAMMERS
                     2462 ;
                     2463 ; The only differences between the OS7 version and this ADAM version
                     2464 ;   are transparent to the game programmer and important to the
                     2465 ;   sound programmer. Formerly CALL's to PLAY_SONGS and SND_MANAGER
                     2466 ;   were necessary to update the sounds. Now a CALL to SOUNDS runs
                     2467 ;   the same code. Formerly CALL's to LEAVE_EFFECT and EFXOVER were
                     2468 ;   needed to end a SPECIAL_EFFECT. Now EFFECT_OVER runs the same code.
                     2469 ;   Independent CALL's to PLAY_SONGS, SND_MANAGER, LEAVE_EFFECT and
                     2470 ;   EFXOVER do not run in this module as before!!!!!
                     2471 ;
                     2472 ;*****************************************************************************
                     2473 ;
                     2474 ;       Operating system sound routine EQUATES
                     2475
<000F>               2476 OFF              EQU         0FH ;Off, no sound
                     2477
                     2478 ;       Special byte 0 codes
                     2479
<00FF>               2480 INACTIVE         EQU         0FFH
                     2481 ;*** SEFFECT          EQU         62
<0000>               2482 ENDSDATA         EQU         0
                     2483
                     2484 ;       Offsets within an SxDATA song data area
                     2485
                     2486 ;*** CH               EQU         0
                     2487 ;*** SONGNO           EQU         0
                     2488 ;*** NEXTNOTEPTR      EQU         1
<0003>               2489 FREQ             EQU         3
<0004>               2490 ATN              EQU         4
<0004>               2491 CTRL             EQU         4
<0005>               2492 NLEN             EQU         5
                     2493 ;*** FPS              EQU         6
<0006>               2494 FPSV             EQU         6
<0007>               2495 FSTEP            EQU         7
                     2496 ;*** ALEN             EQU         8
<0008>               2497 ASTEP            EQU         8
                     2498 ;*** APS              EQU         9
<0009>               2499 APSV             EQU         9
                     2500
                     2501 ;       song end codes
                     2502
                     2503 ;*** CH0END           EQU         00010000B
                     2504 ;*** CH1END           EQU         01010000B
                     2505 ;*** CH2END           EQU         10010000B
                     2506 ;*** CH3END           EQU         11010000B
                     2507 ;*** CH0REP           EQU         00011000B
                     2508 ;*** CH1REP           EQU         01011000B
                     2509 ;*** CH2REP           EQU         10011000B
                     2510 ;*** CH3REP           EQU         11011000B
                     2511
                     2512 ;       channel numbers, B7 -B6
                     2513
                     2514 ;*** CH0              EQU         0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2515 ;### CH1          EQU     01000000B
                     2516 ;### CH2          EQU     10000000B
                     2517 ;### CH3          EQU     11000000B
                     2518
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    2520
                    2521 ;****************************
                    2522 ;*        FREQ_SWEEP        *
                    2523 ;****************************
                    2524
                    2525 ;See Users' Manual for description
                    2526 ;RETs Z SET: if note over
                    2527 ;RETs Z RESET: if sweep in progress or note not over
                    2528
    E2C7            2529 FREQ_SWEEP
                    2530
                    2531 ;       * if freq not swept, dec NLEN and RET [setting Z flag]
                    2532
    E2C7 DD7E07     2533         LD      A,[IX+FSTEP]    ;check  for no sweep code
    E2CA B7         2534         OR      A               ;SET Z flag if FSTEP = 0
    E2CB 2009       2535         JR      NZ,L20          ;if PSW is zero note not to be swept
    E2CD DD7E05     2536         LD      A,[IX+NLEN]     ;dec NLEN and
    E2D0 3D         2537         DEC     A               ;SET Z flag if NLEN = 0
    E2D1 C8         2538         RET     Z               ;leave if note over with Z flag SET
    E2D2 DD7705     2539         LD      [IX+NLEN],A     ;store decremented NLEN
    E2D5 C9         2540         RET                     ;RET with Z flag RESET [note not over]
                    2541
                    2542 ;       ENDIF
                    2543
                    2544 ;       * sweep going, so dec FPSV
                    2545
    E2D6            2546 L20
    E2D6 DDE5       2547         PUSH    IX              ;point HL to FPSV
    E2D8 E1         2548         POP     HL
    E2D9 110006     2549         LD      DE,FPSV
    E2DC 19         2550         ADD     HL,DE
    E2DD CDE355     2551         CALL    LOCL_DECLSN         ;dec FPSV
                    2552 *****   JR      NZ,L21          ;if PSW is zero FPSV has timed out
    E2E0 C0         2553         RET     NZ              ;Added uOS
                    2554
                    2555 ;         * dec NLEN and leave if sweep is over
                    2556
    E2E1 CDE369     2557         CALL    LOCL_MSNTOLSN       ;reload FPSV from FPS
    E2E4 2B         2558         DEC     HL              ;point to NLEN [# steps in the sweep]
    E2E5 7E         2559         LD      A,[HL]          ;dec NLEN and
    E2E6 3D         2560         DEC     A               ;SET Z flag if NLEN = 0
    E2E7 C8         2561         RET     Z               ;leave if sweep over with Z flag SET
                    2562
                    2563 ;           * sweep not over, so add FSTEP to FREQ
                    2564
    E2E8 77         2565         LD      [HL],A          ;store decremented NLEN
    E2E9 2B         2566         DEC     HL              ;point HL
    E2EA 2B         2567         DEC     HL              ;to FREQ
    E2EB DD7E07     2568         LD      A,[IX+FSTEP]    ;A = FSTEP [two's complement step size]
    E2EE CDE374     2569         CALL    LOC_ADD816          ;FREQ = FREQ + FSTEP
    E2F1 23         2570         INC     HL              ;point HL to hi FREQ
    E2F2 CB96       2571         RES     2,[HL]          ;RESET B2 in hi FREQ in case add caused > 10 bit FREQ
    E2F4 F6FF       2572         OR      0FFH            ;RESET Z flag, sweep not over yet
                    2573
                    2574 ;       ENDIF
                    2575 ;*** L21
    E2F6 C9         2576         RET
```

```
LOCATION OBJECT CODE LINE      SOURCE LINE
                     2577 ;        END FREQSWE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2579
                     2580 ;**************************
                     2581 ;*       ATN_SWEEP        *
                     2582 ;**************************
                     2583
                     2584 ;See User's Manual for description
                     2585 ;RETs Z SET: if byte 8 is 0 [means sweep is over, or note was never swept]
                     2586 ;RETs Z RESET: if sweep in progress
                     2587
    E2F7             2588 ATN_SWEEP
                     2589
                     2590 ;       * RET with Z SET if byte 8 = 00
                     2591
    E2F7 DD7E08      2592         LD      A,[IX+8]        ;check byte 8 for no sweep code
    E2FA B7          2593         OR      A               ;Z is set if byte 8 = 0
    E2FB C8          2594         RET     Z               ;leave if Z set, sweep not going
                     2595
                     2596 ;       * sweep going, so dec APSV
                     2597
    E2FC DDE5        2598         PUSH    IX              ;point HL to APSV
    E2FE E1          2599         POP     HL
    E2FF 110009      2600         LD      DE,APSV
    E302 19          2601         ADD     HL,DE
    E303 CDE355      2602         CALL    LOCL_DECLSN          ;dec APSV [LSN of byte 9]
                     2603 *****   JR      NZ,L22          ;if PSW is zero APSV has timed out
    E306 C0          2604         RET     NZ              ;Added uOS
                     2605
                     2606 ;       * dec ALEN to see if sweep over
                     2607
    E307 CDE369      2608         CALL    LOCL_MSNTOLSN        ;reload APSV from APS
    E30A 2B          2609         DEC     HL              ;point to ALEN [# of steps in the sweep]
    E30B CDE355      2610         CALL    LOCL_DECLSN          ;dec ALEN [LSN byte 8]
    E30E 2815        2611         JR      Z,L23           ;if PSW is non-zero sweep not over yet
                     2612
                     2613 ;       * add ASTEP to ATN
                     2614
    E310 7E          2615         LD      A,[HL]          ;MSN A = ASTEP
    E311 E6F0        2616         AND     0F0H            ;mask LSN
    E313 5F          2617         LD      E,A             ;E = ASTEP   0
    E314 2B          2618         DEC     HL              ;point HL to ATN
    E315 2B          2619         DEC     HL
    E316 2B          2620         DEC     HL
    E317 2B          2621         DEC     HL
    E318 7E          2622         LD      A,[HL]          ;MSN A = ATN
    E319 E6F0        2623         AND     0F0H            ;A = ATN   0
    E31B 83          2624         ADD     A,E             ;MSN A = [ASTEP + ATN]   0
    E31C 5F          2625         LD      E,A             ;E = [ASTEP + ATN]   0
    E31D 7E          2626         LD      A,[HL]          ;A = ATN   freq or CTRL
    E31E E60F        2627         AND     0FH             ;mask old ATN  A = 0   freq or CTRL
    E320 B3          2628         OR      E               ;OR in new ATN
    E321 77          2629         LD      [HL],A          ;store updated value back into song data area
    E322 F6FF        2630         OR      0FFH            ;RESET Z flag, sweep not over yet
                     2631 *****   JR      L22
    E324 C9          2632         RET                     ;Added uOS
                     2633
                     2634 ;       ELSE    Z flag is SET: sweep over
                     2635
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E325             2636 L23
    E325 3600        2637         LD      [HL],0          ;set byte 8 to 0 to indicate end sweep
                     2638
                     2639 ;       ENDIF
                     2640 ;       ENDIF
                     2641
                     2642 ;### L22
    E327 C9          2643         RET
                     2644
                     2645 ;       END ATNSWEE
                     2646
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2648
                     2649 ;*****************************
                     2650 ;*       UPATNCTRL           *
                     2651 ;*****************************
                     2652
                     2653 ;Perform single byte update of the snd chip noise control register or any
                     2654 ;attenuation register. IX is passed pointing to byte 0 of a song data area, MSN
                     2655 ;register C = formatted channel attenuation code.
                     2656
E328                 2657 UPATNCTRL
E328 DD7E04          2658          LD      A,[IX+4]          ;MSN A = ATN, LSN may be CTRL data
E32B CB61            2659          BIT     4,C               ;test for ATN
                     2660                                    ;if PSW is non-zero ATN is to be sent,
                     2661                                    ; move it to the LSN
E32D 2804            2662          JR      Z,L24
E32F OF              2663          RRCA                      ;swap nibbles
E330 OF              2664          RRCA
E331 OF              2665          RRCA
E332 OF              2666          RRCA                      ;LSN A = ATN
                     2667 .
                     2668 ;         ENDIF
                     2669
E333                 2670 L24
E333 E60F            2671          AND     OFH               ;mask MSN
E335 B1              2672          OR      C                 ;A = formatted register#   ATN or CTRL
                     2673 *****    CALL    OUT_TO_SOUND_PORT  ;output ATN or CTRL data
                     2674 *****    RET
E336 C3E600          2675          JP      OUT_TO_SOUND_PORT  ;output ATN or CTRL data
                     2676
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2678
                     2679 ;**************************
                     2680 ;*       UPFREQ           *
                     2681 ;**************************
                     2682
                     2683 ;Perform double byte update of a sound chip frequency register.  IX is passed
                     2684 ;pointing to byte0 of a song data area, MSN register D = formatted channel
                     2685 ;frequency code.
                     2686
   E339              2687 UPFREQ
   E339 DD7E03       2688         LD      A,[IX+FREQ]      ;A = F2 F3 F4 F5 F6 F7 F8 F9
   E33C E60F         2689         AND     OFH              ;A =  0  0  0  0 F6 F7 F8 F9
   E33E B2           2690         OR      D                ;A =  formatted reg#   F6 F7 F8 F9
   E33F CDE60D       2691         CALL    OUT_TO_SOUND_PORT  ;output first freq byte
   E342 DD7E03       2692         LD      A,[IX+FREQ]      ;A = F2 F3 F4 F5 F6 F7 F8 F9 again
   E345 E6F0         2693         AND     0F0H             ;A = F2 F3 F4 F5  0  0  0  0
   E347 57           2694         LD      D,A              ;save in D
   E348 DD7E04       2695         LD      A,[IX+FREQ+1]    ;LSN A = 0  0 F0 F1
   E34B E60F         2696         AND     OFH              ;A =  0  0  0  0  0  0 F0 F1
   E34D B2           2697         OR      D                ;A = F2 F3 F4 F5  0  0 F0 F1
   E34E OF           2698         RRCA                     ;swap nibbles
   E34F OF           2699         RRCA
   E350 OF           2700         RRCA
   E351 OF           2701         RRCA                     ;A =  0  0 F0 F1 F2 F3 F4 F5
                     2702 ****    CALL    OUT_TO_SOUND_PORT  ;output 2nd [most significant] freq byte
                     2703 ****    RET
   E352 C3E60D       2704         JP      OUT_TO_SOUND_PORT  ;output 2nd [most significant] freq byte
                     2705
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2707
                     2708 ;*****************************
                     2709 ;*       LOCL_DECLSN         *
                     2710 ;*****************************
                     2711
                     2712 ;Without affecting the MSN, decrement the LSN of the byte pointed to by HL.
                     2713 ;HL remains the same.
                     2714 ;RET with Z flag set if dec LSN results in 0, reset otherwise.
                     2715 ;RET with C flag set if dec LSN results in -1, reset otherwise.
                     2716
    E355             2717 __DECLSN:
    E355             2718 LOCL_DECLSN
    E355 AF          2719         XOR     A
    E356 ED67        2720         RRD                     ;A = 0   LSN [HL]
    E358 D601        2721         SUB     1               ;Z flag set if dec to 0, C flag if dec to -1
    E35A F5          2722         PUSH    AF              ;save Z and C flags
    E35B ED6F        2723         RLD                     ;[HL] = old MSN   new LSN
    E35D F1          2724         POP     AF              ;restore Z and C flags, A = 0   new LSN
    E35E C9          2725         RET
                     2726
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2728
                     2729 ;**************************
                     2730 ;*       LOCL_DECMSN      *
                     2731 ;**************************
                     2732
                     2733 ;Without affecting the LSN, decrement the MSN of the byte pointed to by HL.
                     2734 ;HL remains the same.
                     2735 ;RET with Z flag set if dec MSN results in 0, reset otherwise.
                     2736 ;RET with C flag set if dec MSN results in -1, reset otherwise.
                     2737
   E35F              2738 __DECMSN
   E35F              2739 LOCL_DECMSN
   E35F AF           2740         XOR     A
   E360 ED6F         2741         RLD                     ;A = 0   MSN [HL]
   E362 D601         2742         SUB     1               ;Z flag set if dec to 0, C flag set if dec -1
   E364 F5           2743         PUSH    AF              ;save Z and C flags
   E365 ED67         2744         RRD                     ;[HL] = new MSN   old LSN
   E367 F1           2745         POP     AF              ;restore Z and C flags, A = 0   new MSN
   E368 C9           2746         RET
                     2747
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2749
                     2750 ;*************************
                     2751 ;*       LOCL_MSNTOLSN     *
                     2752 ;*************************
                     2753
                     2754 ;Copy MSN of the byte pointed to by HL to the LSN of that byte.  HL remains
                     2755 ;the same.
                     2756
    E369             2757 _MSNTOLSN
    E369             2758 LOCL_MSNTOLSN
    E369 7E          2759         LD      A,[HL]          ;A = MSN   LSN to be changed
    E36A E6F0        2760         AND     0F0H            ;A = MSN   0
    E36C 47          2761         LD      B,A             ;save in B
    E36D 0F          2762         RRCA                    ;swap nibbles
    E36E 0F          2763         RRCA
    E36F 0F          2764         RRCA
    E370 0F          2765         RRCA                    ;A = 0   MSN
    E371 B0          2766         OR      B               ;A = MSN   MSN
    E372 77          2767         LD      [HL],A          ;[HL] = MSN   MSN
    E373 C9          2768         RET
                     2769
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2771
                     2772 ;***************************
                     2773 ;*       LOC_ADD816        *
                     2774 ;***************************
                     2775
                     2776 ;Adds 8 bit two's complement signed value passed in A to the 16 bit location
                     2777 ;pointed to by HL.
                     2778
    E374             2779 __ADD816
    E374             2780 LOC_ADD816
    E374 0600        2781         LD      B,0             ;set B for positive value in A
    E376 CB7F        2782         BIT     7,A             ;if A is positive
    E378 2801        2783         JR      Z,POS           ;skip
                     2784 ****    LD      B,0FFH          ;A is neg: extend sign bit thru B
    E37A 05          2785         DEC     B               ;Added uOS
    E37B             2786 POS
    E37B 86          2787         ADD     A,[HL]          ;do 8 bit add [and set Carry]
    E37C 77          2788         LD      [HL],A          ;store result into LSB 16 bit number
    E37D 23          2789         INC     HL              ;put MSB
    E37E 7E          2790         LD      A,[HL]          ;into A
    E37F 88          2791         ADC     A,B             ;A = MSB + Carry + B [B is 0 or FF]
    E380 77          2792         LD      [HL],A          ;store result into MSB
    E381 2B          2793         DEC     HL              ;re-point HL to LSB 16 bit number
    E382 C9          2794         RET
                     2795
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2797
                     2798 ;****************************
                     2799 ;*     PT_IX_TO_SxDATA      *
                     2800 ;****************************
                     2801
                     2802 ;SONGNO passed in B.
                     2803 ;Point IX to byte 0 in SONGNO's song data area.
                     2804 ;RET with both DE and IX pointing to SxDATA,
                     2805 ;HL pointing to MSB SxDATA entry in LST_OF_SND_ADDRS.
                     2806
    E383             2807 PT_IX_TO_SxDATA
                     2808
                     2809 ;       * IX & DE := addr of byte 0 in SONGNO's song  data area,
                     2810 ;       HL pointing to MSB SxDATA entry in LST_OF_SND_ADDRS.
                     2811
    E383 2AFE6E      2812         LD      HL,[PTR_TO_LST_OF_SND_ADDRS] ;point HL to start LST_OF_SND_ADDRS
    E386 2B          2813         DEC     HL                ;init HL for addition
    E387 2B          2814         DEC     HL
    E388 48          2815         LD      C,B               ;form 4 * SONGNO in C
    E389 0600        2816         LD      B,0
    E38B CB01        2817         RLC     C
    E38D CB01        2818         RLC     C
    E38F 09          2819         ADD     HL,BC             ;HL pts to SxDATA's entry in LST_OF_SND_ADDRS
    E390 5E          2820         LD      E,[HL]            ;move addr SxDATA to IX thru DE
    E391 23          2821         INC     HL
    E392 56          2822         LD      D,[HL]
    E393 D5          2823         PUSH    DE
    E394 DDE1        2824         POP     IX
    E396 C9          2825         RET
                     2826
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2828
                     2829 ;****************************
                     2830 ;*       AREA_SONG_IS       *
                     2831 ;****************************
                     2832
                     2833 ;The address of byte 0 of a song data area is passed in IX.  The song number of
                     2834 ;the song using that area is returned in A [OFFH if inactive].  If a special
                     2835 ;effect was using that area, 62 is returned in A and HL is returned with the
                     2836 ;address of the special sound effect routine.
                     2837
    E397             2838 AREA_SONG_IS
    E397 DD7E00      2839         LD      A,[IX+0]        ;A := CH#   SONGNO or 62, or A := FF
    E39A FEFF        2840         CP      OFFH
    E39C C8          2841         RET     Z               ;leave with A = FF if area inactive
    E39D E63F        2842         AND     00111111B       ;mask CH#
    E39F FE3E        2843         CP      62
    E3A1 C0          2844         RET     NZ              ;leave with A = SONGNO if not a special effect [62]
                     2845
                     2846 ;       special effect, so set HL to addr effect, stored in bytes 1&2
                     2847
    E3A2 DDE5        2848         PUSH    IX              ;point HL to byte 1
    E3A4 E1          2849         POP     HL
    E3A5 23          2850         INC     HL
    E3A6 5E          2851         LD      E,[HL]          ;save LSB effect addr in E
    E3A7 23          2852         INC     HL              ;HL to byte 2
    E3A8 56          2853         LD      D,[HL]          ;save MSB effect addr in D
    E3A9 EB          2854         EX      DE,HL           ;HL := addr special effect
    E3AA C9          2855         RET
                     2856
                     2857 ;       END UTIL
                     2858
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2860
                     2861 ;**************************
                     2862 ;*       INIT_SOUND       *
                     2863 ;**************************
                     2864
                     2865 ;See Users' Manual for description; includes ENTRY POINT ALL_OFF
                     2866 ;addr LST_OF_SND_ADDRS passed in HL
                     2867 ;n = # of song data areas to init, passed in B
                     2868
                     2869 ;       *** Sound chip register code EQUATES ***
                     2870
                     2871 ;       Tone generator frequency and attenuation formatted register codes
                     2872
          <0080>     2873 SR1FRQ      EQU  10000000B      ;BIT7 = 1, BIT6-4 = TONE GEN 1 FREQ CODE
          <0090>     2874 SR1ATN      EQU  10010000B      ;BIT7 = 1, BIT6-4 = TONE GEN 1 ATTN CODE
          <00A0>     2875 SR2FRQ      EQU  10100000B      ;BIT7 = 1, BIT6-4 = TONE GEN 2 FREQ CODE
          <00B0>     2876 SR2ATN      EQU  10110000B      ;BIT7 = 1, BIT6-4 = TONE GEN 2 ATTN CODE
          <00C0>     2877 SR3FRQ      EQU  11000000B      ;BIT7 = 1, BIT6-4 = TONE GEN 3 FREQ CODE
          <00D0>     2878 SR3ATN      EQU  11010000B      ;BIT7 = 1, BIT6-4 = TONE GEN 1 ATTN CODE
                     2879
                     2880 ;       Noise generator control and attenuation formatted register codes
                     2881
          <00E0>     2882 SRNCTL      EQU  11100000B      ;BIT7 = 1, BIT6-4 = NOISE GEN CONTROL CODE
          <00F0>     2883 SRNATN      EQU  11110000B      ;BIT7 = 1, BIT6-4 = NOISE GEN ATTN CODE
                     2884
E3AB                 2885 __SOUND_INIT:
                     2886 ;*** INIT_SOUND
                     2887
                     2888 ;       * initialize PTR_TO_LST_OF_SND_ADDRS with value passed in HL
                     2889
E3AB 22FE6E          2890         LD [PTR_TO_LST_OF_SND_ADDRS],HL
                     2891
                     2892 ;       * store inactive code at byte 0 of each of the n data areas [B = n]
                     2893
E3AE 23              2894         INC     HL              ;pt HL to song 1 data area entry in LST_OF_SND_ADDRS
E3AF 23              2895         INC     HL
E3B0 5E              2896         LD      E,[HL]          ;pt DE to byte 0 in first song data area
E3B1 23              2897         INC     HL
E3B2 56              2898         LD      D,[HL]
E3B3 EB              2899         EX      DE,HL           ;pt HL to byte 0 in first song data area
E3B4 11000A          2900         LD      DE,10           ;set DE for 10 byte increment
E3B7 3EFF            2901         LD      A,0FFH          ;Added uOS
E3B9                 2902 B1:
E3B9 77              2903         LD      [HL],A          ;deactivate area  *** Added uOS
E3BA 19              2904         ADD     HL,DE           ;pt HL to byte 0 next area (10 bytes away)
E3BB 10FC            2905         DJNZ    B1              ;do this for the n (passed in B) data areas
                     2906
                     2907 ;       * store end of data area code (0) at first byte after last song data area
                     2908
E3BD 3600            2909         LD      [HL],0          ;store end of data area code in byte 0 data area n + 1
                     2910
                     2911 ;       * set the 4 channel data area pointers to a dummy, inactive data area
                     2912
E3BF 21E3E6          2913         LD   HL,DUMAREA         ;point HL to inactive byte below [after the RET]
E3C2 22FE70          2914         LD   [PTR_TO_S_ON_0],HL ;store addr DUMAREA at PTR_TO_S_ON_0
E3C5 22FE72          2915         LD   [PTR_TO_S_ON_1],HL ;store addr DUMAREA at PTR_TO_S_ON_1
E3C8 22FF            2916         LD   [PTR_TO_S_ON_2],HL ;store addr D  AREA at PTR_TO_S_ON_2
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E3CB 22FE76      2917         LD   [PTR_TO_S_ON_3],HL ;store addr DUMAREA at PTR_TO_S_ON_3
                     2918
                     2919 ;       * initialize SAVE_CTRL
                     2920
                     2921 *****   LD      A,0FFH          ;note: this is only time MSN SAVE_CTRL will be non zero,
                     2922 *****                           ;Out u0S (loaded earlier)
                     2923
    E3CE 32FE78      2924         LD      [SAVE_CTRL],A   ;thus ensuring PLAY_SONGS will output 1st real CTRL data
    E3D1             2925 __TURN_OFF_SOUND:
                     2926 ;eee ALL_OFF
                     2927
                     2928 ;       * turn off all 4 sound generators
                     2929
    E3D1 3AFC2F      2930         LD      A,[SOUNDPORT]   ;Added u0S
    E3D4 4F          2931         LD      C,A             ;Added u0S
    E3D5 3E9F        2932         LD      A,0FF+SR1ATN    ;form off code for tone generator 1
    E3D7 ED79        2933         OUT     [C],A           ;send it out
    E3D9 3EBF        2934         LD      A,0FF+SR2ATN    ;form off code for tone generator 2
    E3DB ED79        2935         OUT     [C],A           ;send it out
    E3DD 3EDF        2936         LD      A,0FF+SR3ATN    ;form off code for tone generator 3
    E3DF ED79        2937         OUT     [C],A           ;send it out
    E3E1 3EFF        2938         LD      A,0FF+SRNATN    ;form off code for noise generator, N
    E3E3 ED79        2939         OUT     [C],A           ;send it out
    E3E5 C9          2940         RET
                     2941
    E3E6 FF          2942 DUMAREA DEFB    INACTIVE
                     2943
                     2944 ;       END INITSOU
                     2945
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2947
                     2948 ;**************************
                     2949 ;*        JUKE_BOX        *
                     2950 ;**************************
                     2951
                     2952 ;see Users' Manual for description
                     2953 ;SONGNO passed in B
                     2954
    E3E7             2955 __PLAY_IT
    E3E7             2956 JUKE_BOX
                     2957
                     2958 ;       * RET if song already in progress
                     2959
    E3E7 C5          2960         PUSH    BC                ;save SONGNO on stack
    E3E8 CDE383      2961         CALL    PT_IX_TO_SxDATA ;point IX to SONGNO's song data area
    E3EB DD7E00      2962         LD      A,[IX+0]          ;A := CH# [if any]   SONGNO [if any]
    E3EE E63F        2963         AND     3FH               ;A := 0 0 SONGNO
    E3F0 C1          2964         POP     BC                ;B := SONGNO
    E3F1 B8          2965         CP      B                 ;test if already in progress
    E3F2 C8          2966         RET     Z                 ;if so, leave
                     2967
                     2968 ;       * load first note and set NEXT_NOTE_PTR [thru LOAD_NEXT_NOTE]
                     2969
    E3F3 DD7000      2970         LD      [IX+0],B          ;store SONGNO in byte 0
    E3F6 2B          2971         DEC     HL                ;-HL left by PT_IX_TO_SxDATA pointing to MSB SxDATA
    E3F7 2B          2972         DEC     HL                ;-entry in LST_OF_SND_ADDRS point HL to note list
    E3F8 56          2973         LD      D,[HL]            ;-starting addr entry in LST_OF_SND_ADDRS and save this
    E3F9 2B          2974         DEC     HL                ;-addr in DE
    E3FA 5E          2975         LD      E,[HL]            ;DE now has the initial value for NEXT_NOTE_PTR
    E3FB DD7301      2976         LD      [IX+1],E          ;set NEXT_NOTE_PTR for first note in song
    E3FE DD7202      2977         LD      [IX+2],D
    E401 CDE4F2      2978         CALL    LOAD_NEXT_NOTE  ;load note, byte 0 := CH# SONGNO, set new NEXT_NOTE_PTR
    E404 1874        2979         JR      UP_CH_DATA_PTRS ;new song, so update channel data ptrs
                     2980
                     2981 ;       END JUKEBOX
                     2982
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2984
                     2985 ;***************************
                     2986 ;*       PLAY_SONGS_       *
                     2987 ;***************************
                     2988
                     2989 ;        *** Sound chip register code EQUATES
                     2990
                     2991 ;*** WHITE      EQU      000001008   ;BIT2 = 1, white noise code
                     2992 ;*** PERIOD     EQU      000000008   ;BIT2 = 0, periodic noise code
                     2993 ;*** NSRHI      EQU      000000008   ;BITO-1 Set for highest noise shift rate [N/512]
                     2994 ;*** NSRMED     EQU      00000001B   ;BITO-1 Set for medium noise shift rate [N/1024]
                     2995 ;*** NSRLOW     EQU      000000108   ;BITO-1 Set for lowest noise shift rate [N/2048]
                     2996 ;*** NSRTG3     EQU      00000011B   ;BITO-1 Set for shift from tone gen 3 output
                     2997
E406                 2998 __SOUNDS
                     2999 ;*** PLAY_SONGS_
                     3000
                     3001 ;        * output CH1 attenuation and frequency
                     3002
E406 3E9F            3003          LD       A,OFF+SR1ATN     ;format CH1 OFF byte into A
E408 0E90            3004          LD       C,SR1ATN         ;format MSN C for CH1 attenuation
E40A 1680            3005          LD       D,SR1FRQ         ;format MSN D for CH1 frequency
E40C DD2AFE72        3006          LD       IX,[PTR_TO_S_ON_1] ;point IX to byte 0 data area of song for CH1
E410 CDE456          3007          CALL     TONE_OUT
                     3008
                     3009 ;        * output CH2 attenuation and frequency
                     3010
E413 3EBF            3011          LD       A,OFF+SR2ATN     ;format CH2 OFF byte into A
E415 0EB0            3012          LD       C,SR2ATN         ;format MSN C for CH2 attenuation
E417 16A0            3013          LD       D,SR2FRQ         ;format MSN D for CH2 frequency
E419 DD2AFE74        3014          LD       IX,[PTR_TO_S_ON_2] ;point IX to byte 0 data area of song for CH2
E41D CDE456          3015          CALL     TONE_OUT
                     3016
                     3017 ;        * output CH3 attenuation and frequency
                     3018
E420 3EDF            3019          LD       A,OFF+SR3ATN     ;format CH3 OFF byte into A
E422 0ED0            3020          LD       C,SR3ATN         ;format MSN C for CH3 attenuation
E424 16C0            3021          LD       D,SR3FRQ         ;format MSN D for CH3 frequency
E426 DD2AFE76        3022          LD       IX,[PTR_TO_S_ON_3] ;point IX to byte 0 data area of song for CH3
E42A CDE456          3023          CALL     TONE_OUT
                     3024
                     3025 ;        * output CH0 [noise] ATN [and CTRL, if different from last time]
                     3026
E42D 3EFF            3027          LD       A,OFF+SRNATN     ;format CH0 OFF byte into A
E42F 0EF0            3028          LD       C,SRNATN         ;format MSN C for CH0 attenuation
E431 DD2AFE70        3029          LD       IX,[PTR_TO_S_ON_0] ;point IX to byte 0 data area of song for CH0
E435 DD5E00          3030          LD       E,[IX+0]         ;look for inactive code, OFFH
E438 1C              3031          INC      E                ;this sets Z flag if E = OFFH
E439 2005            3032          JR       NZ,L5            ;if PSW is zero song data area is inactive
E43B CDE60D          3033          CALL     OUT_TO_SOUND_PORT  ;turn off CH0
E43E 1825            3034          JR       L6               ;SND_MANAGER
                     3035
                     3036 ;        ELSE
                     3037
E440                 3038 L5
E440 CDE328          3039          CALL     UPATNCTRL        ;send out current ATN
E443 DD7E04          3040          LD       A,[IX+CTRL]      ;LSN A = current CTRL data
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E446 E60F       3041          AND     OFH             ;mask MSN
    E448 21FE78     3042          LD      HL,SAVE_CTRL    ;point to last CTRL data sent
    E44B BE         3043          CP      [HL]            ;compare
    E44C 2817       3044          JR      Z,L6          ;if PSW is zero CTRL has not changed (done)
    E44E 77         3045          LD      [HL],A          ;SAVE_CTRL = new CTRL data
    E44F 0EE0       3046          LD      C,SRNCTL        ;send new CTRL data
    E451 CDE328     3047          CALL    UPATNCTRL
    E454 180F       3048          JR      L6              ;SND_MANAGER
                    3049
                    3050 ;        ENDIF
                    3051 ;        ENDIF
                    3052
    E456            3053 TONE_OUT
    E456 DD5E00     3054          LD      E,[IX+0]        ;look for inactive code, OFFH
    E459 1C         3055          INC     E               ;this sets Z flag if E = OFFH
    E45A 2003       3056          JR      NZ,L7           ;if PSW is zero song data area is inactive
                    3057 ****     CALL    OUT_TO_SOUND_PORT  ;turn off CHx
                    3058 ****     JR      L8
    E45C C3E60D     3059          JP      OUT_TO_SOUND_PORT  ;turn off CHx
                    3060
                    3061 ;        ELSE send out current ATN and FREQ
                    3062
    E45F            3063 L7
    E45F CDE328     3064          CALL    UPATNCTRL       ;send out attenuation
    E462 C3E339     3065          JP      UPFREQ          ;send out frequency
                    3066 ****     CALL    UPFREQ          ;send out frequency
    E465            3067 L8
                    3068 ****     RET
                    3069 ;        ENDIF
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3071 ;*****************************
                     3072 ;*       SND_MANAGER         *
                     3073 ;*****************************
                     3074
                     3075 ;See Users' Manual for description
                     3076
     E465            3077 L6
     E465            3078 SND_MANAGER
                     3079
                     3080 ;       * IX := addr of song #1 data area [S1DATA]
                     3081
     E465 0601       3082         LD      B,1             ;pt IX to byte 0 song data area for song # 1
     E467 CDE383     3083         CALL    PT_IX_TO_SxDATA
                     3084
     E46A            3085 L1                              ;LOOP until end of song data areas
     E46A 3E00       3086         LD      A,ENDSDATA      ;check for end of song data areas
     E46C DDBE00     3087         CP      [IX+0]          ;set Z flag if inactive
     E46F C8         3088         RET     Z               ;leave [Z set], if all data areas have been processed
                     3089
                     3090 ;       * process active song data areas
                     3091
     E470 CDE4CD     3092         CALL    PROCESS_DATA_AREA;update counters or call effect get next note
                     3093
                     3094 ;       * point IX to byte 0 next song data area
                     3095
     E473 11000A     3096         LD      DE,10
     E476 DD19       3097         ADD     IX,DE
     E478 18F0       3098         JR      L1              ;REPEAT LOOP
                     3099
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3101
                     3102 ;***************************
                     3103 ;*     UP_CH_DATA_PTRS     *
                     3104 ;***************************
                     3105
                     3106 ;For each active data area, starting with S1DATA and proceeding in order, load
                     3107 ;the associated channel data area pointer [PTR_TO_S_ON_x] with the address of
                     3108 ;byte 0.This routine is called by JUKE_BOX, when a song starts and
                     3109 ;PROCESS_DATA_AREA when the channel using a data area has changed as a result of
                     3110 ;calling LOAD_NEXT_NOTE [this happens when a song finishes and when it switches
                     3111 ;back and forth between noise and tone notes].
                     3112
E47A                 3113 UP_CH_DATA_PTRS
E47A DDE5            3114         PUSH    IX              ;save curent IX
E47C 21E3E6          3115         LD      HL,DUMAREA      ;set all 4 ch data ptrs to dummy inactive area
E47F 22FE70          3116         LD      [PTR_TO_S_ON_0],HL
E482 22FE72          3117         LD      [PTR_TO_S_ON_1],HL
E485 22FE74          3118         LD      [PTR_TO_S_ON_2],HL
E488 22FE76          3119         LD      [PTR_TO_S_ON_3],HL
E48B 0601            3120         LD      B,1             ;set IX to byte 0 S1DATA
E48D CDE383          3121         CALL    PT_IX_TO_SxDATA ;RETs with IX addr byte 0 song 1
                     3122
E490                 3123 L2                              ;LOOP until end of song data areas
E490 DD7E00          3124         LD      A,[IX+0]        ;test for end of song data araes
E493 B7              3125         OR      A           ;**** CP      ENDSDATA
E494 281F            3126         JR      Z,DONE_SNDMAN   ;leave loop if all data areas checked
                     3127
                     3128 ;        * if area active, set appropriate channel data area pointer
                     3129
E496 3C              3130         INC     A ;****   CP      INACTIVE        ;check for inactive data area: don't up date ptr if so
E497 2815            3131         JR      Z,L9            ;if PSW is non-zero area is active: update channel data ptrs
E499 DD7E00          3132         LD      A,[IX+0]        ;get CH# in A
E49C E6C0            3133         AND     0C0H            ;B7 - B6  in A  = CH#
E49E 07              3134         RLCA                    ;form CH#  * 2  in A, i.e., the offset from
E49F 07              3135         RLCA                    ;PTR_TO_S_ON_0  of the  channel data area pointer
E4A0 07              3136         RLCA                    ;that points  to channel  CH#
E4A1 5F              3137         LD      E,A             ;add offset to addr of PTR_TO_ S_ON_0
E4A2 1600            3138         LD      D,0
E4A4 21FE70          3139         LD      HL,PTR_TO_S_ON_0
E4A7 19              3140         ADD     HL,DE           ;HL points to  proper channel  data area pointer
E4A8 DDE5            3141         PUSH    IX              ;store this song data area's byte  0  addr there
E4AA D1              3142         POP     DE
E4AB 73              3143         LD      [HL],E
E4AC 23              3144         INC     HL
E4AD 72              3145         LD      [HL],D
                     3146
                     3147 ;        ENDIF
                     3148
                     3149 ;        * point IX to byte 0 next song data area
                     3150
E4AE                 3151 L9
E4AE 11000A          3152         LD      DE,10
E4B1 DD19            3153         ADD     IX,DE
E4B3 18DB            3154         JR      L2              ;REPEAT LOOP
                     3155
E4B5                 3156 DONE_SNDMAN
E4B5 DDE1            3157         POP     IX              ;restore IX
```

```
LOCATION OBJECT CODE LINE      SOURCE LINE

                     3158 ;●●● L12
    E4B7 C9          3159          RET
                     3160
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3162
                     3163 ;****************************
                     3164 ;*      LEAVE_EFFECT        *
                     3165 ;****************************
                     3166
                     3167 ;LEAVE_EFFECT, called by a special sound effect routine when it's finished,
                     3168 ;restores the SONGNO of the song to which the effect note belongs to B5 - B0 of
                     3169 ;byte 0 in the effect's data area, and loads bytes 1 and 2 with the address of
                     3170 ;the next note in the song.  The address of the 1 byte SONGNO (saved by the
                     3171 ;effect when first called) is passed in DE.  The 2 byte address of the next note
                     3172 ;in the song, also saved by the effect, is passed in HL.  IX is assumed to be
                     3173 ;pointing to byte 0 of the data area to which the song number is to be restored.
                     3174 ;Bits 7 and 6 of the saved SONGNO are ignored, and therefore may be used by the
                     3175 ;effect to store flag information during the course of the note.
                     3176
   E4B8              3177 __EFFECT_OVER
                     3178 ;### LEAVE_EFFECT
   E4B8 DD7501       3179         LD      [IX+1],L        ;LSB NEXT_NOTE_PTR := LSB addr next note in song
   E4BB DD7402       3180         LD      [IX+2],H        ;MSB NEXT_NOTE_PTR := MSB addr next note in song
   E4BE 1A           3181         LD      A,[DE]          ;A := x x SONGNO (i.e., the saved, original SONGNO)
   E4BF E63F         3182         AND     03FH            ;A := 0 0 SONGNO
   E4C1 47           3183         LD      B,A             ;B := 0 0 SONGNO (B now = original SONGNO)
   E4C2 DD7E00       3184         LD      A,[IX+0]        ;A := CH#   62 (all effect notes have SONGNO = 62)
   E4C5 E6C0         3185         AND     0C0H            ;A := CH# 0 0 0 0 0 0
   E4C7 B0           3186         OR      B               ;A := CH#   SONGNO
   E4C8 DD7700       3187         LD      [IX+0],A        ;restore the song number
   E4CB 1816         3188         JR      EFXOVER
                     3189
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3191
                     3192 ;***************************
                     3193 ;*   PROCESS_DATA_AREA     *
                     3194 ;***************************
                     3195
                     3196 ;See Users' Manual for description
                     3197 ;Terminology: SFX = address of sound effect routine
                     3198
    E4CD             3199 PROCESS_DATA_AREA
    E4CD CDE397      3200          CALL    AREA_SONG_IS     ;return area's SONGNO in A [and addr SFX in HL]
    E4D0 FEFF        3201          CP      INACTIVE         ;test for inactive code
    E4D2 C8          3202          RET     Z                ;RET, no processing if area inactive
                     3203
                     3204 ;         * if special effect, call it to process the data area
                     3205
    E4D3 FE3E        3206          CP      62               ;test for special sound effect
    E4D5 2005        3207          JR      NZ,L10           ;if PSW is zero data area used by sound effect
    E4D7 110007      3208          LD      DE,7             ;pt HLto SFX+7, starting adr of the effect's cod e
    E4DA 19          3209          ADD     HL,DE
    E4DB E9          3210          JP      [HL]             ;do 1 pass thru effect, RET from effect
                     3211
                     3212 ;          ENDIF
                     3213
                     3214 ;         * else process a non-effect note
                     3215
    E4DC             3216 L10
    E4DC CDE2F7      3217          CALL    ATN_SWEEP        ;process atn sweep data, if any
    E4DF CDE2C7      3218          CALL    FREQ_SWEEP       ;proc frq sweep data, if any, & note dura timer s
                     3219 *****    JR      NZ,L12           ;if PSW is zero note is over
    E4E2 C0          3220          RET     NZ
    E4E3             3221 EFXOVER
    E4E3 DD7E00      3222          LD      A,[IX+0]         ;A := CH#   SONGNO this note
    E4E6 F5          3223          PUSH    AF               ;save on stack
    E4E7 CDE4F2      3224          CALL    LOAD_NEXT_NOTE   ;load data for next note
    E4EA C1          3225          POP     BC               ;B := CH#   SONGNO previous note
    E4EB DD7E00      3226          LD      A,[IX+0]         ;A := CH#   SONGNO new note [may be inactive]
    E4EE B8          3227          CP      B                ;check against new note's CH#   SONGNO
                     3228 ****     JR      Z,L12            ;if PSW is non-zero change to/from tone/efx/noise
    E4EF C8          3229          RET     Z
    E4F0 1888        3230          JR      UP_CH_DATA_PTRS  ;to maintain data area priority system
                     3231
                     3232 ;         ENDIF
                     3233 ;         END SNDMAN
                     3234
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3236
                     3237 ;**************************
                     3238 ;*     LOAD_NEXT_NOTE     *
                     3239 ;**************************
                     3240
                     3241 ;see Users' Manual for description
                     3242 ;SFX refers to the beginning address of a special sound effect routine
                     3243
    E4F2             3244 LOAD_NEXT_NOTE
                     3245
                     3246 ;       * deactivate area, save SONGNO on stack
                     3247
    E4F2 DD7E00      3248         LD      A,[IX+0]        ;A := byte 0
    E4F5 E63F        3249         AND     00111111B       ;mask CH#, if any
    E4F7 F5          3250         PUSH    AF              ;save SONGNO on stack
    E4F8 DD3600FF    3251         LD      [IX+0],INACTIVE ;deactivate area
                     3252
                     3253 ;       * A := header new note
                     3254
    E4FC DD6E01      3255         LD      L,[IX+1]        ;HL := addr new note in ROM
    E4FF DD6602      3256         LD      H,[IX+2]
    E502 7E          3257         LD      A,[HL]          ;A := header new note
                     3258
                     3259 ;       * save header of new note in song on stack and load its data CASE note type
                     3260
    E503 47          3261         LD      B,A             ;save header new note in B
    E504 CB6F        3262         BIT     5,A             ;test for rest
    E506 281C        3263         JR      Z,L13           ;if PSW is non-zero note is a rest
                     3264
                     3265 ;*** REST
    E508 C5          3266         PUSH    BC              ;save header on stack
    E509 E61F        3267         AND     00011111B       ;mask all but duration bits
                     3268
                     3269 ;       * set up NEXT_NOTE_PTR
                     3270
    E50B 23          3271         INC     HL              ;HL = address of the header of the note after this note
    E50C DD7501      3272         LD      [IX+1],L        ;store in NEXT_NOTE_PTR
    E50F DD7402      3273         LD      [IX+2],H
                     3274
                     3275 ;       * move this note's data and fill in bytes where necessary
                     3276
    E512 DD3604F0    3277         LD      [IX+ATN],0F0H   ;set atn off
    E516 DD7705      3278         LD      [IX+NLEN],A     ;NLEN := 5 bit duration
    E519 DD360700    3279         LD      [IX+FSTEP],0    ;indicate freq not to be swept
    E51D DD360800    3280         LD      [IX+ASTEP],0    ;indicate atn not to be swept
    E521 C3E5EC      3281         JP      MODBO
                     3282
                     3283 ;       ENDIF
                     3284
    E524             3285 L13
    E524 CB67        3286         BIT     4,A             ;test for end
    E526 280E        3287         JR      Z,L14           ;if PSW is non-zero end of song
    E528 CB5F        3288         BIT     3,A             ;test for repeat
    E52A 2804        3289         JR      Z,ENDNOREP      ;if PSW is non-zero repeat song
                     3290 ;*** ENDREP
    E52C C1          3291         POP     BC              ;B := SONGNO
    E52D C3E[illegible] 3292      JP      JUKE_BOX        ;to reload 1st note of this song
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3293
                     3294 ;       RET                        ;to PROCESS_DATA_AREA, don't save header
                     3295 ;       ENDIF
                     3296
     E530            3297 ENDNOREP
     E530 3EFF       3298         LD      A,INACTIVE
     E532 F5         3299         PUSH    AF                 ;save inactive code to end song
     E533 C3E5EC     3300         JP      MODBO              ;to load byte 0
                     3301
                     3302 ;       ENDIF
                     3303 ;       - test for special sound effect
                     3304
     E536            3305 L14
     E536 E63C       3306         AND     00111100B          ;mask irrelevant bits
     E538 FE04       3307         CP      00000100B          ;test for B5 - B2 = 0001
     E53A 2027       3308         JR      NZ,L15             ;if PSW is zero note is a special effect
                     3309 ;*** EFFECT
     E53C FDE1       3310         POP     IY                 ;IY := SONGNO
     E53E FDE5       3311         PUSH    IY                 ;put SONGNO back on stack
     E540 C5         3312         PUSH    BC                 ;save header on stack NEXT_NOTE_PTR := SFX, DE := SFX
     E541 23         3313         INC     HL                 ;-pt HL to next byte [LSB addr SFX]
     E542 5E         3314         LD      E,[HL]             ;-E := LSB SFX
     E543 DD7301     3315         LD      [IX+1],E           ;-put LSB of SFX in byte 1 of SxDATA [NEXT_NOTE_PTR]
     E546 23         3316         INC     HL                 ;-pt HL to MSB SFX
     E547 56         3317         LD      D,[HL]             ;-D := MSB SFX
     E548 DD7202     3318         LD      [IX+2],D           ;-put MSB SFX in byte 2 of SxDATA
     E54B 23         3319         INC     HL                 ;point HL to next note [after this new note]
     E54C FDE5       3320         PUSH    IY                 ;A := SONGNO
     E54E F1         3321         POP     AF
     E54F D5         3322         PUSH    DE                 ;IY := SFX
     E550 FDE1       3323         POP     IY
     E552 11E558     3324         LD      DE,PASS1           ;create "CALL [IY]" with RET to PASS1 by storing
     E555 D5         3325         PUSH    DE                 ;PASS1 on the stack
     E556 FDE9       3326         JP      [IY]               ;1st 7 bytes SFX will save addr next note & SONGNO
     E558            3327 PASS1
     E558 110007     3328         LD      DE,7               ;in same fashion, create a "CALL (IY+7)"
     E55B FD19       3329         ADD     IY,DE              ;to allow SFX to load initial values
     E55D 11E5EC     3330         LD      DE,MODBO           ;RET to MODBO
     E560 D5         3331         PUSH    DE
     E561 FDE9       3332         JP      [IY]
                     3333
                     3334 ;       ENDIF
                     3335
                     3336 ;       - if here, note is type 0 - 3
                     3337
     E563            3338 L15
     E563 C5         3339         PUSH    BC                 ;save header on stack
     E564 78         3340         LD      A,B                ;A := fresh copy header
     E565 E603       3341         AND     00000011B          ;mask all but type number
     E567 B7         3342         OR      A                  ;test for type 0
     E568 2020       3343         JR      NZ,L16             ;if PSW is zero note is type 0: fixed freq and atn
                     3344
                     3345 ;*** TYPE0                              ;set up NEXT_NOTE_PTR
     E56A 23         3346         INC     HL                 ;next note [after this new note] is 4 bytes away,
     E56B 23         3347         INC     HL                 ;point HL to it
     E56C 23         3348         INC     HL
     E56D 23         3349         INC     HL
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E56E DD7501       3350          LD      [IX+1],L        ;put addr in NEXT_NOTE_PTR
    E571 DD7402       3351          LD      [IX+2],H
                      3352
                      3353 ;        * move new note data and fill in bytes where necessary
                      3354
    E574 2B           3355          DEC     HL              ;point HL back to 1st ROM data to move, NLEN
    E575 110005       3356          LD      DE,05           ;point DE to destination: bytes 5, 4, and 3
    E578 CDE603       3357          CALL    DE_TO_DEST
    E57B 010003       3358          LD      BC,3            ;move 3 bytes
    E57E EDB8         3359          LDDR
    E580 DD360700     3360          LD      [IX+FSTEP],0    ;set for no freq sweep
    E584 DD360800     3361          LD      [IX+ASTEP],0    ;set for no atn sweep
    E588 1862         3362          JR      MODBO
                      3363
                      3364 ;        ENDIF
                      3365
    E58A              3366 L16
    E58A FE01         3367          CP      1               ;test for type 1
    E58C 201A         3368          JR      NZ,L17          ;if PSW is zero note is type 1: swept
                      3369                                  ; freq, fixed attenuation
                      3370 ;*** TYPE1                           ; set up NEXT_NOTE_PTR
    E58E 110006       3371          LD      DE,6            ;note after this note is 6 bytes away,
    E591 19           3372          ADD     HL,DE           ;pt HL to it
    E592 DD7501       3373          LD      [IX+1],L        ;store in NEXT_NOTE_PTR
    E595 DD7402       3374          LD      [IX+2],H
                      3375
                      3376 ;        * move new note data and fill in bytes where necessary
                      3377
    E598 2B           3378          DEC     HL              ;point HL back to 1st ROM data to move, FSTEP
    E599 1C           3379          INC     E               ;E; point DE to destination: bytes 7 - 3
    E59A CDE603       3380          CALL    DE_TO_DEST
    E59D 010005       3381          LD      BC,5            ;move 5 bytes
    E5A0 EDB8         3382          LDDR
    E5A2 DD360800     3383          LD      [IX+ASTEP],0    ;set for no atn sweep
    E5A6 1844         3384          JR      MODBO
                      3385
                      3386 ;        ENDIF
                      3387
    E5A8              3388 L17
    E5A8 FE02         3389          CP      2               ;test for type 2
    E5AA 2026         3390          JR      NZ,TYPE3        ;if PSW is zero note is type 2: fixed freq, swept attenuation
                      3391 ;*** TYPE2                           ;set up NEXT_NOTE_PTR
    E5AC 110006       3392          LD      DE,6            ;pt HL to note after this note, since it's 6 bytes away,
    E5AF 19           3393          ADD     HL,DE           ;pt HL to it by adding 6
    E5B0 F1           3394          POP     AF              ;A := header this note [CH#   SONGNO]
    E5B1 F5           3395          PUSH    AF              ;put back on stack
    E5B2 E6C0         3396          AND     11000000B       ;mask SONGNO, leaving CH#
    E5B4 2001         3397          JR      NZ,L18          ;if PSW is zero this is a noise note, which is only 5 ROM bytes long
    E5B6 2B           3398          DEC     HL              ;so move HL back 1 byte
                      3399
                      3400 ;        ENDIF
                      3401
    E5B7              3402 L18
    E5B7 DD7501       3403          LD      [IX+1],L        ;put addr in NEXT_NOTE_PTR
    E5BA DD7402       3404          LD      [IX+2],H
                      3405
                      3406 ;        * move new note data and fill in b   s where necessary
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3407
    E5BD 2B          3408          DEC     HL              ;point HL back to 1st ROM data to move, APS
    E5BE 1E09        3409          LD      E,9             ;point DE to destination: bytes 9,8,5 - 3
    E5C0 CDE603      3410          CALL    DE_TO_DEST
    E5C3 010002      3411          LD      BC,2            ;move 2 bytes
    E5C6 EDB8        3412          LDDR                    ;when done, DE points to FSTEP, HL to ROM NLEN
    E5C8 AF          3413          XOR     A
    E5C9 12          3414          LD      [DE],A          ;FSTEP := 0 for no freq sweep
    E5CA 1B          3415          DEC     DE              ;pt DE to RAM NLEN
    E5CB 1B          3416          DEC     DE
    E5CC 0E03        3417          LD      C,3             ;move last 3 ROM bytes if this is a noise note, garbage
    E5CE EDB8        3418          LDDR                    ;will be loaded into byte 3, buts that's OK
    E5D0 181A        3419          JR      MODBO
                     3420
                     3421 ;        ENDIF
                     3422
                     3423 ;        if here, note is type 3: swept freq, swept attenuation
                     3424
    E5D2             3425 TYPE3                            ;set up NEXT_NOTE_PTR
    E5D2 110008      3426          LD      DE,8            ;note after this note is 8 bytes away,
    E5D5 19          3427          ADD     HL,DE           ;pt HL to it
    E5D6 DD7501      3428          LD      [IX+1],L        ;put addr in NEXT_NOTE_PTR
    E5D9 DD7402      3429          LD      [IX+2],H
                     3430
                     3431 ;        * move new note data and fill in bytes where necessary
                     3432
    E5DC 2B          3433          DEC     HL              ;point HL back to 1st ROM data to move, APS
    E5DD DDE5        3434          PUSH    IX              ;point DE to destination: bytes 9 - 3
    E5DF FDE1        3435          POP     IY              ;IY := addr byte 0 [and DE = 6]
    E5E1 1C          3436          INC     E               ;DE := 9
    E5E2 FD19        3437          ADD     IY,DE           ;IY := addr byte 9 [APS]
    E5E4 FDE5        3438          PUSH    IY
    E5E6 D1          3439          POP     DE              ;DE := addr APS
    E5E7 010007      3440          LD      BC,7            ;move 7 bytes
    E5EA EDB8        3441          LDDR
                     3442
                     3443 ;        * modify byte 0 basis header new note
                     3444
    E5EC             3445 MODBO
    E5EC DDE5        3446          PUSH    IX              ;pt HL to byte 0
    E5EE E1          3447          POP     HL
    E5EF F1          3448          POP     AF              ;A := header new note
    E5F0 C1          3449          POP     BC              ;B := SONGNO
    E5F1 FEFF        3450          CP      INACTIVE        ;test for inactive [song over, as detected above]
    E5F3 C8          3451          RET     Z
    E5F4 57          3452          LD      D,A             ;save header in D
    E5F5 E63F        3453          AND     3FH             ;Rid channel bits
    E5F7 FE04        3454          CP      04              ;Special effect
    E5F9 2002        3455          JR      NZ,L20_LOAD_NEX
    E5FB 063E        3456          LD      B,62
                     3457
    E5FD             3458 L20_LOAD_NEX:
    E5FD 7A          3459          LD      A,D             ;restore A to header
    E5FE E6C0        3460          AND     0C0H            ;A := CH# 0 0 0 0 0 0
    E600 B0          3461          OR      B               ;A := new CH#   SONGNO
    E601 77          3462          LD      [HL],A          ;store back in byte 0
                     3463
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3464 ;       ENDIF
                     3465
                     3466 ;### L19
    E602 C9          3467         RET
                     3468
    E603             3469 DE_TO_DEST                  ;DE passed = offset from byte 0, RETed w addr byte offset
    E603 DDE5        3470         PUSH    IX
    E605 FDE1        3471         POP     IY          ;IY := addr byte 0 [and DE = offset]
    E607 FD19        3472         ADD     IY,DE       ;IY := addr byte 0 + offset
    E609 FDE5        3473         PUSH    IY
    E608 D1          3474         POP     DE          ;DE := addr of destination byte in SxDATA
    E60C C9          3475         RET
                     3476
                     3477 ;       END LOADNEX
                     3478
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3480 ;-----------------------------------------------------------------
                     3481 ;
                     3482 ;  Out_to_sound_port
                     3483 ;               input:  a=sound port data to output
                     3484 ;               register usage: n.a.
                     3485 ;
    E60D             3486 OUT_TO_SOUND_PORT:
    E60D C5          3487         PUSH    BC                      ;SAVE BC
    E60E 47          3488         LD      B,A                     ;THE IDEA IS TO GET THE
    E60F 3AFC2F      3489         LD      A,[SOUNDPORT]           ;INDIRECT PORT ADDRESSING
    E612 4F          3490         LD      C,A                     ;THRU THE PORT TABLE IN EOS
    E613 ED41        3491         OUT     [C],B
    E615 78          3492         LD      A,B
    E616 C1          3493         POP     BC
    E617 C9          3494         RET
                     3495 ;
                     3496 ;*****************************************************************
                     3497 ;*****************************************************************
                     3498 ;*****************************************************************
                     3499 ;
                     3500
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3502         ;GLB      __QUERY_FILE,__SET_FILE,__MAKE_FILE
                     3503         ;GLB      __FILE_QUERY
                     3504         ;EXT      FCB_HEAD_ADDR,FCB_DATA_ADDR
                     3505         ;EXT      SCAN_FOR_FILE
                     3506         ;EXT      BLOCKS_REQ,USER_BUF,USER_NAME
                     3507         ;EXT      BUF_START,BUF_END
                     3508         ;EXT      STRCMP,BASECMP
                     3509         ;EXT      NEW_HOLE_SIZE,NEW_HOLE_START,HOLE_FILE_NAME
                     3510         ;EXT      EOS_YEAR,EOS_MONTH,EOS_DAY
                     3511         ;EXT      BLK_STRT_PTR,VOL_BLK_SZ      ;DLS(8/29/83)
                     3512         ;EXT      READ_BLOCK,WRITE_BLOCK,CHECK_IF_DIRECTORY
                     3513         ;EXT      DIR_BLOCK_NO,FOUND_AVAIL_ENTRY
                     3514         ;EXT      FILENAME_CMPS
                     3515
                     3516 ;-----------------------------------------------------------------
                     3517 ;
                     3518 ; __QUERY_FILE -- Read the file's directory entry. (USES STRCMP FOR FILE NAME COMPARISIONS)
                     3519 ; __FILE_QUERY -- SAME AS ABOVE BUT SETS UP SCAN_FOR_FILE FOR BASE COMPARES  ( USES BASECMP)
                     3520 ;
                     3521 ; CALLING PARAMETERS: Device number in A
                     3522 ;                      address of name string in DE
                     3523 ;                      address of buffer in HL
                     3524 ;
                     3525 ; EXIT PARAMETERS:  if no errors -- Z = 1; A = 0; BCDE = file's start block;
                     3526 ;                       directory entry in caller's buffer
                     3527 ;                   if errors -- Z = 0; A = error code; DE = junk;
                     3528 ;                       caller's buffer undefined
                     3529 ;
                     3530 ;-----------------------------------------------------------------
                     3531
E618                 3532 __FILE_QUERY:
E618 37              3533         SCF                      ; THIS ENTRY DOES NOTHING MORE THAN SET CARRY FLAG
E619 1802            3534         JR       QUEER
                     3535
E61B                 3536 __QUERY_FILE
E61B 37              3537         SCF                      ; THIS ENTRY CLEARS CARRY FLAG
E61C 3F              3538         CCF
E61D                 3539 QUEER:
E61D E5              3540         PUSH     HL              ;SAVE REG'S
E61E DDE5            3541         PUSH     IX
                     3542
E620 F5              3543         PUSH     AF
                     3544
E621 3801            3545         JR       C,SET_UP_A      ; DID WE COME IN AT FILE_QUERY
                     3546
E623 AF              3547         XOR      A                  ; USE STRCMP FOR FILE NAME COMPARE
E624                 3548 SET_UP_A:                           ;
E624 32FDD8          3549         LD       [FILENAME_CMPS],A  ; 0 = COMPARE COMPLETE FILE NAME
                     3550                                     ; NOT 0 - COMPARE ONLY THE BASE
E627 F1              3551         POP      AF
                     3552
E628 22FE06          3553         LD       [USER_BUF],HL   ;SAVE USER'S DATA ADDRESS
                     3554
E62B DD2AFDFD        3555         LD       IX,[FCB_HEAD_ADDR] ;POINT TO SYSTEM'S FCB
E62F 62              3556         LD       H,D             ;GET NAME POINTER IN HL
E630 6B              3557         LD       L,E
E631 CD    B         3558         CALL     SCAN_FOR_FILE   ;LOOK IN TH'   'RECTORY
```

```
LOCATION OBJECT CODE LINE      SOURCE LINE

    E634 2017        3559          JR      NZ,Q_ERROR        ;BRANCH IF ERROR RETURNED
                     3560
    E636 D5          3561          PUSH    DE                ;SAVE START BLOCK OF FILE
    E637 C5          3562          PUSH    BC
                     3563
    E638 ED5BFE06    3564          LD      DE,[USER_BUF]     ;GET BUFFER ADDRESS IN DE
    E63C DD6E21      3565          LD      L,[IX+FCB_POINTER] ;GET ADDR OF ENTRY IN HL
    E63F DD6622      3566          LD      H,[IX+FCB_POINTER+1]
    E642 010017      3567          LD      BC,DIR_ENT_LENGTH-3  ; SET BYTE COUNT TO LOAD ONLY VALID INFO THAT
                     3568                                       ; THAT WAS STORED ON THE DEVICE
                     3569
    E645 EDB0        3570          LDIR                      ;COPY DATA TO CALLER'S BUFFER
                     3571
    E647 C1          3572          POP     BC                ;GET FILE'S START BLOCK
    E648 D1          3573          POP     DE
                     3574
    E649 AF          3575          XOR     A                 ;SHOW NO ERROR
    E64A 32FDD8      3576          LD      [FILENAME_CMPS],A  ; DEFAULT FOR SCAN FOR FILE
    E64D             3577 Q_ERROR:
    E64D DDE1        3578          POP     IX                ;RESTORE REG'S
    E64F E1          3579          POP     HL
    E650 C9          3580          RET
                     3581
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    3583 ;---------------------------------------------------------------
                    3584 ;
                    3585 ; __SET_FILE -- Re-write the file's directory entry.
                    3586 ;
                    3587 ; CALLING PARAMETERS: Device number in A
                    3588 ;                      address of name string in DE
                    3589 ;                      address of buffer in HL
                    3590 ;
                    3591 ; EXIT PARAMETERS:  if no errors -- Z = 1; A = 0;
                    3592 ;                       directory entry updated
                    3593 ;                    if errors -- Z = 0; A = error code;
                    3594 ;                       directory entry unchanged
                    3595 ;
                    3596 ;---------------------------------------------------------------
                    3597
    E651            3598 __SET_FILE
    E651 C5         3599         PUSH    BC              ;SAVE REG'S
    E652 D5         3600         PUSH    DE
    E653 E5         3601         PUSH    HL
    E654 DDE5       3602         PUSH    IX
    E656 22FE06     3603         LD      [USER_BUF],HL   ;SAVE ADDR OF USER'S DATA
                    3604
    E659 DD2AFDFD   3605         LD      IX,[FCB_HEAD_ADDR] ;POINT TO SYSTEM'S FCB
                    3606
    E65D 62         3607         LD      H,D             ;GET STRING ADDRESS IN HL
    E65E 6B         3608         LD      L,E
    E65F CDEF0B     3609         CALL    SCAN_FOR_FILE   ;GET THE DIR ENTRY
    E662 2026       3610         JR      NZ,S_ERROR      ;BRANCH IF ERROR
                    3611
    E664 2AFE06     3612         LD      HL,[USER_BUF]   ;GET BUFFER ADDRESS
    E667 DD5E21     3613         LD      E,[IX+FCB_POINTER] ;GET ENTRY ADDR IN DE
    E66A DD5622     3614         LD      D,[IX+FCB_POINTER+1]
    E66D 010017     3615         LD      BC,DIR_ENT_LENGTH-3   ;SET BYTE COUNT
    E670 EDB0       3616         LDIR                    ;COPY THEIR BUFFER TO BLOCK BUFFER
                    3617
    E672            3618 TAPE_TIMED_OUT:
    E672 DD7E17     3619         LD      A,[IX+FCB_DEVICE] ;SET DEVICE NUMBER
    E675 2AFDFF     3620         LD      HL,[FCB_DATA_ADDR] ;GET ADDR OF MY BUFFER
    E678 DD5E19     3621         LD      E,[IX+FCB_BLOCK] ;GET BLOCK ADDRESS IN BCDE
    E67B DD561A     3622         LD      D,[IX+FCB_BLOCK+1]
    E67E DD4E1B     3623         LD      C,[IX+FCB_BLOCK+2]
    E681 DD461C     3624         LD      B,[IX+FCB_BLOCK+3]
    E684 CDF1E6     3625         CALL    WRITE_BLOCK      ;RE-WRITE THE DIRECTORY BLOCK
                    3626
    E687 2001       3627         JR      NZ,WRITE_ERRORS
    E689 AF         3628         XOR     A               ;SHOW NO ERRORS
                    3629
    E68A            3630 WRITE_ERRORS:
    E68A            3631 S_ERROR
                    3632
    E68A DDE1       3633         POP     IX              ;RESTORE REG'S
    E68C E1         3634         POP     HL
    E68D D1         3635         POP     DE
    E68E C1         3636         POP     BC
    E68F C9         3637         RET
                    3638
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3640 ;---------------------------------------------------------------
                     3641 ;
                     3642 ; __MAKE_FILE: creates a file in the directory.
                     3643 ;
                     3644 ; ENTRY PARAMETERS: called with device ID in A; address of name
                     3645 ;                string in HL; file size (in bytes) in BC DE.
                     3646 ;
                     3647 ;  NOTE:  IF FILE SIZE = 0 (BC DE), THEN THE REMAINDER OF THE TAPE IS
                     3648 ;         ALLOCATED TO THE FILE
                     3649 ;
                     3650 ; EXIT PARAMETERS: CONDITION FLAGS
                     3651 ;                     Z   -   NO ERRORS
                     3652 ;                    NZ   -   ERRORS
                     3653 ;                             A = ERROR CODE
                     3654 ;
                     3655 ;      ALL REGISTERS ARE PRESERVED EXCEPT AF
                     3656 ;
                     3657 ;---------------------------------------------------------------
                     3658
           <0000>    3659 FOUND_ENTRY     EQU     0       ; BIT INDICATING DELETED FILE FOUND FOR OVERLAYING
                     3660
    E690             3661 __MAKE_FILE
    E690 FDE5        3662         PUSH    IY              ;SAVE REGISTERS
    E692 DDE5        3663         PUSH    IX
    E694 E5          3664         PUSH    HL
    E695 D5          3665         PUSH    DE
    E696 C5          3666         PUSH    BC
                     3667
    E697 DD2AFDFD    3668         LD      IX,[FCB_HEAD_ADDR] ;POINT TO SYSTEM FCB
    E69B DD361800    3669         LD      [IX+FCB_MODE],0     ; INIT IT IN CASE IT WAS SET
                     3670
    E69F DD7717      3671         LD      [IX+FCB_DEVICE],A ;SAVE DEVICE NUMBER
    E6A2 22FE10      3672         LD      [USER_NAME],HL  ;SAVE POINTER TO NAME
                     3673
    E6A5 78          3674         LD      A,B             ; CHECK IF TO ALLOCATE REST OF TAPE
    E6A6 B1          3675         OR      C
    E6A7 200A        3676         JR      NZ,GOT_FILE_SIZE        ; NOPE, GOT SIZE
                     3677
    E6A9 7B          3678         LD      A,E
    E6AA B2          3679         OR      D
    E6AB 2006        3680         JR      NZ,GOT_FILE_SIZE
    E6AD DDCB18EE    3681         SET     MODE_REMAINDER_BIT,[IX+FCB_MODE]  ; FLAG IT
                     3682
    E6B1 1820        3683         JR      SET_UP_DIR
                     3684
                     3685
                     3686 ; SINCE WE ARE CALLED WITH A BYTE COUNT, DIVIDE IT BY 1024
                     3687 ; (SHIFT RIGHT 10 BITS) TO GET A BLOCK COUNT.
                     3688
    E6B3             3689 GOT_FILE_SIZE:
    E6B3 5A          3690         LD      E,D             ;DO A QUICK SHIFT BY 8
    E6B4 51          3691         LD      D,C
    E6B5 48          3692         LD      C,B
    E6B6 0600        3693         LD      B,0
    E6B8 CB39        3694         SRL     C               ;SHIFT ONE MORE
    E6BA CB1A        3695         RR      D
    E6BC CB1B        3696         RR      E
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E6BE CB39          3697         SRL     C               ;THEN ONE LAST TIME
    E6C0 CB1A          3698         RR      D
    E6C2 CB1B          3699         RR      E
                       3700
                       3701 ; FOR THIS TO BE A LEGAL FILE SIZE, IT MUST FIT INTO DE.
                       3702
    E6C4 79            3703         LD      A,C             ;CHECK FOR TOO BIG
    E6C5 B7            3704         OR      A
    E6C6 C2E992        3705         JP      NZ,TOO_BIG      ;JUMP IF TOO BEEG, SENOR
    E6C9 13            3706         INC     DE              ;ROUND UP IN CASE THERE WAS A FRACTION
    E6CA 7B            3707         LD      A,E             ;CHECK FOR TOO BIG AGAIN
    E6CB B2            3708         OR      D
    E6CC CAE992        3709         JP      Z,TOO_BIG
    E6CF ED53FE0C      3710         LD      [BLOCKS_REQ],DE ;ELSE SAVE THE BLOCK COUNT
                       3711
    E6D3               3712 SET_UP_DIR:
    E6D3 110000        3713         LD      DE,0                    ; ZERO OUT HI ADDRESS
    E6D6 ED53FE0E      3714         LD      [BLOCKS_REQ+2],DE
    E6DA ED53FDD9      3715         LD      [DIR_BLOCK_NO],DE
    E6DE AF            3716         XOR     A
    E6DF 32FDDB        3717         LD      [FOUND_AVAIL_ENTRY],A
                       3718
    E6E2 DD361901      3719         LD      [IX+FCB_BLOCK],1 ;INIT NUMBER OF FIRST DIR BLOCK
    E6E6 DD771A        3720         LD      [IX+FCB_BLOCK+1],A  ; ZERO IT OUT
    E6E9 DD771B        3721         LD      [IX+FCB_BLOCK+2],A
    E6EC DD771C        3722         LD      [IX+FCB_BLOCK+3],A
                       3723
    E6EF DD360D01      3724         LD      [IX+FCB_START_BLOCK],1 ;INIT NUMBER OF FIRST DIR BLOCK
    E6F3 DD770E        3725         LD      [IX+FCB_START_BLOCK+1],A
    E6F6 DD770F        3726         LD      [IX+FCB_START_BLOCK+2],A
    E6F9 DD7710        3727         LD      [IX+FCB_START_BLOCK+3],A
                       3728
    E6FC ED5BFDFF      3729         LD      DE,[FCB_DATA_ADDR] ;INIT POINTER TO BUFFER
    E700 DD7321        3730         LD      [IX+FCB_POINTER],E
    E703 DD7222        3731         LD      [IX+FCB_POINTER+1],D
                       3732
    E706 ED53FE08      3733         LD      [BUF_START],DE  ;ALSO INIT HERE WITH SAME
    E70A 210400        3734         LD      HL,1024         ;CALC ADDR OF END OF BUFFER TOO
    E70D 19            3735         ADD     HL,DE
    E70E 22FE0A        3736         LD      [BUF_END],HL    ;AND SAVE IT
                       3737
                       3738 ;THIS LABEL NOT USED IN THIS ROUTINE, NOT DECLARED GLOBAL
                       3739 ;REMOVED BECAUSE IT CONFLICTS WITH IDENTICAL LABEL ELSEWHERE
                       3740 ;READ_TIMEOUT:
    E711 DD7E17        3741         LD      A,[IX+FCB_DEVICE] ;GET PARAMETERS FROM FCB
    E714 DD6E21        3742         LD      L,[IX+FCB_POINTER]
    E717 DD6622        3743         LD      H,[IX+FCB_POINTER+1]
    E71A DD5E19        3744         LD      E,[IX+FCB_BLOCK]
    E71D DD561A        3745         LD      D,[IX+FCB_BLOCK+1]
    E720 DD4E1B        3746         LD      C,[IX+FCB_BLOCK+2]
    E723 DD461C        3747         LD      B,[IX+FCB_BLOCK+3]
    E726 CDF17B        3748         CALL    READ_BLOCK      ;THEN READ THE FIRST BLOCK
                       3749
    E729 C2E989        3750         JP      NZ,MAKE_ERROR   ;BRANCH IF ERROR
                       3751
    E72C               3752 SET_UP_FCB:
    E72C     \FDFF     3753         LD      IY,[FCB_DATA_ADDR] ;POINT    THE BUFFER
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3754 *
    E730 FD7E0C      3755          LD      A,[IY+VOL_DIRSIZE] ;GET THE DIR SIZE BYTE
    E733 E67F        3756          AND     7FH                ;ZERO THE PERM BIT
                     3757 ;        INC     A                  ;BECAUSE WE START IN BLOCK 1
                     3758 ;        DEC     A                  ;TO SOLVE PICKET FENCE PROBLEM
    E735 DD771D      3759          LD      [IX+FCB_LAST_BLOCK],A ;SET THE LAST BLOCK #
    E738 DD361E00    3760          LD      [IX+FCB_LAST_BLOCK+1],0 ;OTHER BYTES WILL BE ZERO
    E73C DD361F00    3761          LD      [IX+FCB_LAST_BLOCK+2],0
    E740 DD362000    3762          LD      [IX+FCB_LAST_BLOCK+3],0
                     3763
    E744 CDF035      3764          CALL    CHECK_IF_DIRECTORY      ; CHECK IF DIRECTORY EXISTS
    E747 C2E989      3765          JP      NZ,MAKE_ERROR
                     3766
    E74A 0626        3767          LD      B,ENT_PER_BLOCK-1 ;INIT ENTRY COUNT, ALLOW FOR VOL ID
                     3768
    E74C             3769 NEXT_ENT
    E74C DD6E21      3770          LD      L,[IX+FCB_POINTER] ;GET CURRENT POINTER
    E74F DD6622      3771          LD      H,[IX+FCB_POINTER+1]
    E752 11001A      3772          LD      DE,DIR_ENT_LENGTH ;GET LENGTH OF AN ENTRY
    E755 19          3773          ADD     HL,DE              ;ADVANCE POINTER TO NEXT ENTRY
    E756 DD7521      3774          LD      [IX+FCB_POINTER],L ;AND SAVE IT AGAIN
    E759 DD7422      3775          LD      [IX+FCB_POINTER+1],H
                     3776
    E75C             3777 CHECK_ENT
    E75C DD6E21      3778          LD      L,[IX+FCB_POINTER] ;GET POINTER INTO IY THE LONG WAY
    E75F DD6622      3779          LD      H,[IX+FCB_POINTER+1]
    E762 E5          3780          PUSH    HL
    E763 FDE1        3781          POP     IY                 ;!!!
                     3782
    E765 FD7E0C      3783          LD      A,[IY+DIR_ATTR] ;GET THE ATTRIBUTE BYTE FOR THIS ENTRY
    E768 4F          3784          LD      C,A                ;SAVE IT HERE FOR NOW
    E769 CB47        3785          BIT     ATTR_HOLE_BIT,A ; CHECK IF HOLE
    E76B C2E807      3786          JP      NZ,FOUND_HOLE     ;BRANCH IF IT'S A HOLE!!! <<------
                     3787
    E76E DDCB186E    3788          BIT     MODE_REMAINDER_BIT,[IX+FCB_MODE]  ; DO WE ALLOCATE REST OF TAPE?
    E772 2037        3789          JR      NZ,D_FILE          ; YUP, DON'T BOTHER CHECKING
                     3790
    E774 CB51        3791          BIT     ATTR_DEL_BIT,C  ;IS IT A DELETED FILE?
    E776 2824        3792          JR      Z,ACTIVE_FILE   ;NOPE, AN ACTIVE ONE
                     3793
                     3794 *        LET'S LOOK FOR DELETED FILE TO USE
                     3795
    E778 FD6E11      3796          LD      L,[IY+DIR_MAX_LENGTH]   ; GET ALLOCATED SIZE
    E77B FD6612      3797          LD      H,[IY+DIR_MAX_LENGTH+1]
                     3798
    E77E ED5BFE0C    3799          LD      DE,[BLOCKS_REQ]         ; MIN NO OF BLOCKS NEEDED
    E782 B7          3800          OR      A                       ;clear carry flag
    E783 ED52        3801          SBC     HL,DE                   ; WILL WE FIT?]
    E785 3824        3802          JR      C,D_FILE                ; NOPE, LOOK FOR ANOTHER ENTRY
                     3803
    E787 21FDDB      3804          LD      HL,FOUND_AVAIL_ENTRY    ; FLAG BYTE
    E78A CB46        3805          BIT     FOUND_ENTRY,[HL]        ; DO WE ALREADY HAVE FIRST FIT?
    E78C 201D        3806          JR      NZ,D_FILE               ; YUP, DON'T EVEN CONSIDER THIS ONE
                     3807
    E78E CBC6        3808          SET     FOUND_ENTRY,[HL]
    E790 DD5E19      3809          LD      E,[IX+FCB_BLOCK]        ;GET BLOCK NO OF THIS DIRECTORY
    E793 DD561A      3810          LD      D,[IX+FCB_BLOCK+1]
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3811
    E796 ED53FDD9    3812          LD      [DIR_BLOCK_NO],DE       ; SAVE IT
    E79A 180F        3813          JR      D_FILE
                     3814
                     3815
                     3816 ; IF THIS IS A REAL FILE, WE CAN SEE IF ITS NAME MATCHES THE
                     3817 ; ONE WE ARE ABOUT TO MAKE -- IT'S AN ERROR IF SO.
                     3818
    E79C             3819 ACTIVE_FILE:
    E79C DD5E21      3820          LD      E,[IX+FCB_POINTER] ;POINT TO THIS ENTRY
    E79F DD5622      3821          LD      D,[IX+FCB_POINTER+1]
    E7A2 2AFE10      3822          LD      HL,[USER_NAME]  ;POINT TO DESIRED NAME
    E7A5 CDF054      3823          CALL    BASECMP         ;COMPARE THE BASE OF THE FILENAMES
    E7A8 CAE996      3824          JP      Z,FILE_EXISTS   ;BRANCH IF THEY'RE THE SAME
    E7AB             3825 D_FILE
    E7AB 109F        3826          DJNZ    NEXT_ENT        ;ELSE GO TO NEXT ENTRY (IF IT EXISTS)
                     3827
    E7AD 2AFE08      3828          LD      HL,[BUF_START]  ;IF NOT RESET FCB FOR NEXT BLOCK
    E7B0 DD7521      3829          LD      [IX+FCB_POINTER],L
    E7B3 DD7422      3830          LD      [IX+FCB_POINTER+1],H
    E7B6 DD3419      3831          INC     [IX+FCB_BLOCK]  ;INC THE BLOCK -- NO CARRY OUT
                     3832
                     3833 ; DO 4-BYTE COMPARE OF FCB_BLOCK VS FCB_LAST_BLOCK TO SEE
                     3834 ; IF THERE'S MORE FILE LEFT.
                     3835
    E7B9 DD7E20      3836          LD      A,[IX+FCB_LAST_BLOCK+3] ;COMPARE MS BYTES
    E7BC DDBE1C      3837          CP      [IX+FCB_BLOCK+3]
    E7BF DAE7FD      3838          JP      C,FULL_DIR      ;BRANCH IF BLOCK IS LARGER
    E7C2 DD7E1F      3839          LD      A,[IX+FCB_LAST_BLOCK+2] ;COMPARE NEXT BYTES
    E7C5 DDBE1B      3840          CP      [IX+FCB_BLOCK+2]
    E7C8 DAE7FD      3841          JP      C,FULL_DIR      ;BRANCH IF BLOCK IS LARGER
    E7CB DD7E1E      3842          LD      A,[IX+FCB_LAST_BLOCK+1] ;COMPARE NEXT BYTES
    E7CE DDBE1A      3843          CP      [IX+FCB_BLOCK+1]
    E7D1 DAE7FD      3844          JP      C,FULL_DIR      ;BRANCH IF BLOCK IS LARGER
    E7D4 DD7E1D      3845          LD      A,[IX+FCB_LAST_BLOCK+0] ;COMPARE LS BYTES
    E7D7 DDBE19      3846          CP      [IX+FCB_BLOCK+0]
    E7DA DAE7FD      3847          JP      C,FULL_DIR      ;BRANCH IF BLOCK IS LARGER
                     3848 ; IF WE FALL OUT, FCB_BLOCK <= FCB_LAST_BLOCK.
    E7DD             3849 DO_READ_AGAIN:
                     3850
    E7DD DD7E17      3851          LD      A,[IX+FCB_DEVICE] ;SET UP THE PARAMETERS
    E7E0 DD6E21      3852          LD      L,[IX+FCB_POINTER]
    E7E3 DD6622      3853          LD      H,[IX+FCB_POINTER+1]
    E7E6 DD5E19      3854          LD      E,[IX+FCB_BLOCK]
    E7E9 DD561A      3855          LD      D,[IX+FCB_BLOCK+1]
    E7EC DD4E1B      3856          LD      C,[IX+FCB_BLOCK+2]
    E7EF DD461C      3857          LD      B,[IX+FCB_BLOCK+3]
    E7F2 CDF17B      3858          CALL    READ_BLOCK      ;AND READ THE NEXT BLOCK OF ENTRIES
    E7F5 C2E989      3859          JP      NZ,MAKE_ERROR           ; NOPE, A REAL ERROR
                     3860
    E7F8             3861 NO_TIMEOUT:
    E7F8 0627        3862          LD      B,ENT_PER_BLOCK ;SET THE ENTRY COUNTER
    E7FA C3E75C      3863          JP      CHECK_ENT       ;AND GO CHECK THE NEW ENTRY
                     3864
    E7FD             3865 FULL_DIR:
    E7FD 21FDDB      3866          LD      HL,FOUND_AVAIL_ENTRY
    E800    6        3867          BIT     FOUND_ENTRY,[HL]        ;    'E WE FOUND AN ENTRY?
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E802 CAE99A      3868         JP      Z,FULL_DIR_EXIT          ; NOPE, EXIT
    E805 1807        3869         JR      USE_ENTRY                ; YUP, CREATE NEW ENTRY THERE
                     3870
                     3871 ; ARRIVE HERE WHEN WE'VE FOUND THE HOLE ENTRY IN THE DIRECTORY.
                     3872 ; IY IS STILL POINTING TO THE ENTRY IN THE BUFFER.
                     3873
    E807             3874 FOUND_HOLE
    E807 21FDDB      3875         LD      HL,FOUND_AVAIL_ENTRY
    E80A CB46        3876         BIT     FOUND_ENTRY,[HL]         ; HAVE WE FOUND A DELETED ENTRY WE COULD USE?
    E80C 2864        3877         JR      Z,USE_HOLE               ; NOPE, STICK IT IN THE END
                     3878
    E80E             3879 USE_ENTRY:
    E80E CB86        3880         RES     FOUND_ENTRY,[HL]         ; RESET FOR NEXT TIME
                     3881
    E810 DD6E19      3882         LD      L,[IX+FCB_BLOCK]         ; GET THE CURRENT BLOCK
    E813 DD661A      3883         LD      H,[IX+FCB_BLOCK+1]
                     3884
    E816 ED5BFDD9    3885         LD      DE,[DIR_BLOCK_NO]        ; BLOCK NO WHERE WE HAVE ENTRY TO USE
    E81A B7          3886         OR      A                        ;clear carry flag
    E81B ED52        3887         SBC     HL,DE                    ; SAME BLOCK?
                     3888
    E81D 2AFE08      3889         LD      HL,[BUF_START]           ; POINT TO START OF BLOCK
    E820 DD7521      3890         LD      [IX+FCB_POINTER],L
    E823 DD7422      3891         LD      [IX+FCB_POINTER+1],H
                     3892
    E826 2824        3893         JR      Z,GOT_BLOCK              ; YUP, DON'T NEED TO READ IN ANOTHER BLOCK
                     3894
    E828 DD7319      3895         LD      [IX+FCB_BLOCK],E         ; GET THIS BLOCK
    E82B DD721A      3896         LD      [IX+FCB_BLOCK+1],D
                     3897
    E82E 010000      3898         LD      BC,0
    E831 DD7E17      3899         LD      A,[IX+FCB_DEVICE]        ; DEVICE ID
                     3900
    E834 CDF17B      3901         CALL    READ_BLOCK
    E837 C2E989      3902         JP      NZ,MAKE_ERROR
    E83A 1810        3903         JR      GOT_BLOCK                ; READ IN NEW DIRECTORY
                     3904
    E83C             3905 NEXT_DIR:
    E83C             3906 TOO_SMALL:
    E83C DD6E21      3907         LD      L,[IX+FCB_POINTER]       ; GET POINTER TO DIR ENTRY
    E83F DD6622      3908         LD      H,[IX+FCB_POINTER+1]
                     3909
    E842 11001A      3910         LD      DE,DIR_ENT_LENGTH        ; LENGTH OF ENTRY
                     3911
    E845 19          3912         ADD     HL,DE                    ; POINT TO NEXT ENTRY
                     3913
    E846 DD7521      3914         LD      [IX+FCB_POINTER],L
    E849 DD7422      3915         LD      [IX+FCB_POINTER+1],H
                     3916
    E84C             3917 GOT_BLOCK:
    E84C E5          3918         PUSH    HL
    E84D FDE1        3919         POP     IY
                     3920
    E84F FDCB0C56    3921         BIT     ATTR_DEL_BIT,[IY+DIR_ATTR]       ; IS IT DELETED?
    E853 28E7        3922         JR      Z,NEXT_DIR                       ; NOPE, LOOK FOR ANOTHER DLETED ENTRY
                     3923
    E855 FD6E11      3924         LD      L,[IY+DIR_MAX_LENGTH]            ; GET ALLOCATED AMOUNT OF DELETED FILE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E858 FD6612      3925          LD      H,[IY+DIR_MAX_LENGTH+1]
                     3926
    E85B ED5BFE0C    3927          LD      DE,[BLOCKS_REQ]                 ; MIN  AMOUNT NEEDED
                     3928
    E85F E5          3929          PUSH    HL
    E860 B7          3930          OR      A                               ;clear carry flag
    E861 ED52        3931          SBC     HL,DE                           ; WILL IT FIT?
    E863 E1          3932          POP     HL
                     3933
    E864 38D6        3934          JR      C,TOO_SMALL                     ; NOPE, TRY ANOTHER ONE
                     3935
    E866 22FE0C      3936          LD      [BLOCKS_REQ],HL                  ; FORCE ALLOCATED AMOUNT
                     3937
    E869 CDE98F      3938          CALL    LOAD_NEW_ENTRY_INFO
                     3939
    E86C C2E989      3940          JP      NZ,MAKE_ERROR
                     3941
    E86F C3E971      3942          JP      TIME_TO_WRITE
                     3943
                     3944 ; COMPARE FCB_BLOCK TO FCB_LAST_BLOCK.  GO TO MORE_BLOCKS
                     3945 ; IF FCB_LAST_BLOCK > FCB_BLOCK.
                     3946
    E872             3947 USE_HOLE:
    E872 DD7E1C      3948          LD      A,[IX+FCB_BLOCK+3]
    E875 DD8E20      3949          CP      [IX+FCB_LAST_BLOCK+3]
    E878 381E        3950          JR      C,MORE_BLOCKS
    E87A DD7E1B      3951          LD      A,[IX+FCB_BLOCK+2]
    E87D DD8E1F      3952          CP      [IX+FCB_LAST_BLOCK+2]
    E880 3816        3953          JR      C,MORE_BLOCKS
    E882 DD7E1A      3954          LD      A,[IX+FCB_BLOCK+1]
    E885 DD8E1E      3955          CP      [IX+FCB_LAST_BLOCK+1]
    E888 380E        3956          JR      C,MORE_BLOCKS
    E88A DD7E19      3957          LD      A,[IX+FCB_BLOCK]
    E88D DD8E1D      3958          CP      [IX+FCB_LAST_BLOCK]
    E890 3806        3959          JR      C,MORE_BLOCKS
                     3960
                     3961 ; FALL THROUGH IF THIS IS THE LAST BLOCK OF THE FILE.
                     3962
    E892 78          3963          LD      A,B             ;LOOK AT ENTRY COUNTER
    E893 FE01        3964          CP      1
    E895 CAE99E      3965          JP      Z,TAPE_FULL     ;ERROR IF NO ENTRIES LEFT VACANT
                     3966
    E898             3967 MORE_BLOCKS
    E898 FD6E11      3968          LD      L,[IY+DIR_MAX_LENGTH] ;GET THE SIZE OF THE HOLE
    E89B FD6612      3969          LD      H,[IY+DIR_MAX_LENGTH+1]
                     3970
    E89E DDCB186E    3971          BIT     MODE_REMAINDER_BIT,[IX+FCB_MODE]  ; NEED TO ALLOCATE REST OF TAPE?
    E8A2 280C        3972          JR      Z,CHECK_HOLE_SIZE                 ; NOPE
                     3973
    E8A4 7C          3974          LD      A,H                     ; YUP, CHECK IF THERE ARE ANY BLOCKS LEFT
    E8A5 B5          3975          OR      L
    E8A6 CAE99E      3976          JP      Z,TAPE_FULL             ; TAPE FULL
                     3977
    E8A9 22FE0C      3978          LD      [BLOCKS_REQ],HL         ; NOT FULL, REQUEST REST OF TAPE
    E8AC DDCB18AE    3979          RES     MODE_REMAINDER_BIT,[IX+FCB_MODE]
                     3980
    E8B0             3981 CHECK_HOLE_SIZE:
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E8B0 ED5BFE0C    3982         LD      DE,[BLOCKS_REQ] ;GET THE REQUESTED SIZE
    E8B4 B7          3983         OR      A
    E8B5 ED52        3984         SBC     HL,DE           ;COMPARE THEM
    E8B7 DAE99E      3985         JP      C,TAPE_FULL     ;BRANCH IF THE HOLE IS TOO SMALL
    E8BA 22FE1A      3986         LD      [NEW_HOLE_SIZE],HL ;ELSE SAVE THE RESULT
                     3987
                     3988
                     3989 ; ADD BLOCKS_REQ TO DIR_START_BLOCK TO FIND THE HOLE'S NEW
                     3990 ; START BLOCK.  NOTICE HOW STOOOPID THE Z80 IS WITH ADDITION!
    E8BD 21FE0C      3991         LD      HL,BLOCKS_REQ   ;POINT TO THE NUMBER TO ADD
    E8C0 FD7E0D      3992         LD      A,[IY+DIR_START_BLOCK]
    E8C3 86          3993         ADD     A,[HL]
    E8C4 23          3994         INC     HL
    E8C5 32FE16      3995         LD      [NEW_HOLE_START],A
    E8C8 FD7E0E      3996         LD      A,[IY+DIR_START_BLOCK+1]
    E8CB 8E          3997         ADC     A,[HL]
    E8CC 23          3998         INC     HL
    E8CD 32FE17      3999         LD      [NEW_HOLE_START+1],A
    E8D0 FD7E0F      4000         LD      A,[IY+DIR_START_BLOCK+2]
    E8D3 8E          4001         ADC     A,[HL]
    E8D4 23          4002         INC     HL
    E8D5 32FE18      4003         LD      [NEW_HOLE_START+2],A
    E8D8 FD7E10      4004         LD      A,[IY+DIR_START_BLOCK+3]
    E8DB 8E          4005         ADC     A,[HL]
    E8DC 23          4006         INC     HL
    E8DD 32FE19      4007         LD      [NEW_HOLE_START+3],A
                     4008
    E8E0 C5          4009         PUSH    BC              ;SAVE ENTRY COUNT (IN B)
                     4010
    E8E1 CDE98F      4011         CALL    LOAD_NEW_ENTRY_INFO
                     4012
    E8E4 C1          4013         POP     BC              ;RESTORE ENTRY COUNT
                     4014
    E8E5 C2E989      4015         JP      NZ,MAKE_ERROR
                     4016
    E8E8 DD6E21      4017         LD      L,[IX+FCB_POINTER] ;GET CURRENT POINTER
                     4018
                     4019
    E8EB DD6622      4020         LD      H,[IX+FCB_POINTER+1]
    E8EE 11001A      4021         LD      DE,DIR_ENT_LENGTH ;GET LENGTH OF AN ENTRY
    E8F1 19          4022         ADD     HL,DE           ;ADVANCE POINTER TO NEXT ENTRY
    E8F2 DD7521      4023         LD      [IX+FCB_POINTER],L ;AND SAVE IT AGAIN
    E8F5 DD7422      4024         LD      [IX+FCB_POINTER+1],H
                     4025
    E8F8 1044        4026         DJNZ    NOT_END         ;JUMP IF THERE ARE MORE ENTRIES
                     4027
                     4028 ;THIS LABEL NOT USED IN THIS ROUTINE, NOT DECLARED GLOBAL
                     4029 ;REMOVED BECAUSE IT CONFLICTS WITH IDENTICAL LABEL ELSEWHERE
                     4030 ;WRITE_AGAIN:
    E8FA DD7E17      4031         LD      A,[IX+FCB_DEVICE] ;GET PARAMETERS FROM FCB
    E8FD 2AFE08      4032         LD      HL,[BUF_START]
    E900 DD5E19      4033         LD      E,[IX+FCB_BLOCK]
    E903 DD561A      4034         LD      D,[IX+FCB_BLOCK+1]
    E906 DD4E1B      4035         LD      C,[IX+FCB_BLOCK+2]
    E909 DD461C      4036         LD      B,[IX+FCB_BLOCK+3]
                     4037
    E90C CDF1E6      4038         CALL    WRITE_BLOCK
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4039
    E90F C2E989      4040          JP       NZ,MAKE_ERROR                    ; NOPE, A REAL LIVE ERROR
                     4041
                     4042
    E912             4043 RESET_THE_POINTER:
    E912 2AFE08      4044          LD       HL,[BUF_START] ;RESET THE POINTER
    E915 DD7521      4045          LD       [IX+FCB_POINTER],L
    E918 DD7422      4046          LD       [IX+FCB_POINTER+1],H
                     4047
    E91B DD3419      4048          INC      [IX+FCB_BLOCK]  ;INC BLOCK -- NO CARRY OUT
                     4049
    E91E             4050 TIMEOUT_IN_READ:
    E91E DD7E17      4051          LD       A,[IX+FCB_DEVICE] ;GET PARAMETERS FROM FCB
    E921 DD6E21      4052          LD       L,[IX+FCB_POINTER]
    E924 DD6622      4053          LD       H,[IX+FCB_POINTER+1]
    E927 DD5E19      4054          LD       E,[IX+FCB_BLOCK]
    E92A DD561A      4055          LD       D,[IX+FCB_BLOCK+1]
    E92D DD4E1B      4056          LD       C,[IX+FCB_BLOCK+2]
    E930 DD461C      4057          LD       B,[IX+FCB_BLOCK+3]
    E933 CDF17B      4058          CALL     READ_BLOCK       ;AND READ THE NEXT BLOCK
                     4059
    E936 2051        4060          JR       NZ,MAKE_ERROR       ; NOPE, ERROR!!!!!!
                     4061
                     4062
    E938 DD6E21      4063          LD       L,[IX+FCB_POINTER] ;GET POINTER INTO IY THE LONG WAY
    E93B DD6622      4064          LD       H,[IX+FCB_POINTER+1]
                     4065
    E93E             4066 NOT_END:
    E93E E5          4067          PUSH     HL
    E93F FDE1        4068          POP      IY                ;!!!
                     4069
                     4070 *
                     4071 *  STILL TRYING TO CALC THE NEW START ADDR
                     4072 *
    E941 2AFE1A      4073          LD       HL,[NEW_HOLE_SIZE]
    E944 FD7511      4074          LD       [IY+DIR_MAX_LENGTH],L          ; BLOCKS LEFT ON TAPE
    E947 FD7412      4075          LD       [IY+DIR_MAX_LENGTH+1],H
                     4076
    E94A 3AFE16      4077          LD       A,[NEW_HOLE_START]
    E94D FD7700      4078          LD       [IY+DIR_START_BLOCK],A
    E950 3AFE17      4079          LD       A,[NEW_HOLE_START+1]
    E953 FD770E      4080          LD       [IY+DIR_START_BLOCK+1],A
    E956 3AFE18      4081          LD       A,[NEW_HOLE_START+2]
    E959 FD770F      4082          LD       [IY+DIR_START_BLOCK+2],A
    E95C 3AFE19      4083          LD       A,[NEW_HOLE_START+3]
    E95F FD7710      4084          LD       [IY+DIR_START_BLOCK+3],A
                     4085
    E962 FD360C01    4086          LD       [IY+DIR_ATTR],ATTR_HOLE          ; SET HOLE UP
                     4087
    E966 FDE5        4088          PUSH     IY
    E968 D1          4089          POP      DE
                     4090
    E969 21F428      4091          LD       HL,HOLE_FILE_NAME                ;
    E96C 01000C      4092          LD       BC,12
    E96F EDB0        4093          LDIR
                     4094
    E971             4095 TIME_TO_WRITE:
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    E971 DD7E17      4096         LD      A,[IX+FCB_DEVICE] ;GET PARAMETERS FROM FCB
    E974 2AFE08      4097         LD      HL,[BUF_START]
    E977 DD5E19      4098         LD      E,[IX+FCB_BLOCK]
    E97A DD561A      4099         LD      D,[IX+FCB_BLOCK+1]
    E97D DD4E1B      4100         LD      C,[IX+FCB_BLOCK+2]
    E980 DD461C      4101         LD      B,[IX+FCB_BLOCK+3]
    E983 CDF1E6      4102         CALL    WRITE_BLOCK       ;WRITE THE BLOCK OUT
                     4103
    E986 2001        4104         JR      NZ,MAKE_ERROR       ; IT'S A BOO BOO!!
    E988 AF          4105         XOR     A                 ;SHOW NO ERROR
                     4106
    E989             4107 LETS_GET_OUT:
    E989             4108 MAKE_ERROR:
                     4109                                   ;RESTORE THE REGISTERS
    E989 B7          4110         OR      A
    E98A C1          4111         POP     BC
    E98B D1          4112         POP     DE
    E98C E1          4113         POP     HL
    E98D DDE1        4114         POP     IX
    E98F FDE1        4115         POP     IY
    E991 C9          4116         RET
                     4117
    E992             4118 TOO_BIG
    E992 3E08        4119         LD      A,TOO_BIG_ERR     ;SHOW AN ERROR
    E994 18F3        4120         JR      MAKE_ERROR
                     4121
                     4122
    E996             4123 FILE_EXISTS
    E996 3E06        4124         LD      A,FILE_EXISTS_ERR
    E998 18EF        4125         JR      MAKE_ERROR
                     4126
    E99A             4127 FULL_DIR_EXIT:
    E99A 3E0C        4128         LD      A,FULL_DIR_ERR
    E99C 18EB        4129         JR      MAKE_ERROR
                     4130
    E99E             4131 TAPE_FULL
    E99E 3E0D        4132         LD      A,FULL_TAPE_ERR
    E9A0 18E7        4133         JR      MAKE_ERROR
                     4134 *
                     4135 *
                     4136 *    SUBROUTINE NAME: GET FILE NAME LENGTH
                     4137 *
                     4138 *       ENTRY: HL- POINTS TO TEXT STRING
                     4139 *       EXIT:  Z=1=FOUND AND PROPER SIZE (1-12)
                     4140 *                BC=BYTE COUNT
                     4141 *                A=TRASHED
                     4142 *
                     4143 *              Z=0=ERROR
                     4144 *                BC=TRASH
                     4145 *                A=FILE_NAME_TOO_LONG
                     4146 * DLS(8/28/83)
                     4147 *
    E9A2             4148 GET_FILE_NM_LEN:
    E9A2 E5          4149                 PUSH    HL                ;SAVE THE POINTER TO FILE NAME
    E9A3 060C        4150                 LD      B,12              ;SCAN UP TO 12 BYTE
    E9A5 0E01        4151                 LD      C,1               ;SET COUNT TO 1
    E9A7             4152 SRCH_LOOP:
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   E9A7 7E          4153                LD      A,[HL]          ;GET FN(I)
   E9A8 FE03        4154                CP      03              ;ETX
   E9AA 2809        4155                JR      Z,GOT_IT
                    4156 *
   E9AC 0C          4157                INC     C
   E9AD 23          4158                INC     HL              ;ADVANCE POINTER TO FILE NAME STRING
   E9AE 10F7        4159                DJNZ    SRCH_LOOP
                    4160 *
   E9B0             4161 ERR_GFN
   E9B0 3E0E        4162                LD      A,FILE_NM_ERR
   E9B2 B7          4163                OR      A               ;SHOW ERROR
   E9B3 E1          4164                POP     HL
   E9B4 C9          4165                RET
                    4166 *
   E9B5             4167 GOT_IT
   E9B5 79          4168                LD      A,C
   E9B6 FE01        4169                CP      1               ;ETX ALONE IS NOT VALID
   E9B8 28F6        4170                JR      Z,ERR_GFN
   E9BA 0600        4171                LD      B,0
   E9BC AF          4172                XOR     A               ;SHOW OK
   E9BD E1          4173                POP     HL
   E9BE C9          4174                RET
                    4175
                    4176
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4178
    E9BF             4179 LOAD_NEW_ENTRY_INFO:
    E9BF FDE5        4180          PUSH    IY              ;GET NAME ADDRESS INTO DE
    E9C1 D1          4181          POP     DE
                     4182
    E9C2 2AFE10      4183          LD      HL,[USER_NAME]  ;GET USER'S STRING ADDRESS
                     4184 *
    E9C5 CDE9A2      4185          CALL    GET_FILE_NM_LEN ; DLS(8/28/83)
    E9C8 C2E9FF      4186          JP      NZ,MAKE_ERR_1
                     4187 *
    E9CB EDB0        4188          LDIR                    ;COPY USER'S NAME INTO DIR ENTRY
                     4189
    E9CD 3E10        4190          LD      A,ATTR_USER     ;SET THE DEFAULT ATTRIBUTE
    E9CF FD770C      4191          LD      [IY+DIR_ATTR],A
                     4192
    E9D2 ED4BFE0C    4193          LD      BC,[BLOCKS_REQ]  ;GET NEW FILE'S MAX SIZE
    E9D6 FD7111      4194          LD      [IY+DIR_MAX_LENGTH],C ;PUT INTO DIR
    E9D9 FD7012      4195          LD      [IY+DIR_MAX_LENGTH+1],B
                     4196
                     4197
    E9DC FD361301    4198          LD      [IY+DIR_USED_LENGTH],1 ;INIT COUNT OF BLOCKS USED
    E9E0 FD361400    4199          LD      [IY+DIR_USED_LENGTH+1],0
                     4200
    E9E4 FD361500    4201          LD      [IY+DIR_LAST_COUNT],0  ;INIT BYTECOUNT IN LAST BLOCK
    E9E8 FD361600    4202          LD      [IY+DIR_LAST_COUNT+1],0
                     4203
    E9EC 3AFDE0      4204          LD      A,[EOS_YEAR]    ;INSERT THE DATE
    E9EF FD7717      4205          LD      [IY+DIR_YEAR],A
    E9F2 3AFDE1      4206          LD      A,[EOS_MONTH]
    E9F5 FD7718      4207          LD      [IY+DIR_MONTH],A
    E9F8 3AFDE2      4208          LD      A,[EOS_DAY]
    E9FB FD7719      4209          LD      [IY+DIR_DAY],A
                     4210
    E9FE AF          4211          XOR     A
                     4212
    E9FF             4213 MAKE_ERR_1:
    E9FF C9          4214          RET
                     4215
                     4216 ;
                     4217 ;*****************************************************************
                     4218 ;*****************************************************************
                     4219 ;*****************************************************************
                     4220 ;
                     4221
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4223         ;GLB      __OPEN_FILE,__CLOSE_FILE,__RESET_FILE
                     4224         ;EXT      FCB_HEAD_ADDR,FCB_DATA_ADDR
                     4225         ;EXT      BUF_START,FMGR_DIR_ENT
                     4226         ;EXT      __QUERY_FILE,MODE_CHECK,READ_BLOCK,WRITE_BLOCK
                     4227         ;EXT      __SET_FILE
                     4228         ;EXT      FILE_NAME_ADDR
                     4229
                     4230 ;---------------------------------------------------------------
                     4231 ;
                     4232 ; __OPEN_FILE -- Sets up an FCB for the caller to access a file.
                     4233 ;
                     4234 ; CALLING PARAMETERS: device number in A; address of name string
                     4235 ;                     in HL; mode in B.
                     4236 ;
                     4237 ; EXIT PARAMETERS:  if no error -- Z = 1; A = file number
                     4238 ;                   if error -- Z = 0; A = error code; B = junk
                     4239 ;
                     4240 ;---------------------------------------------------------------
                     4241
                     4242 ; NEXT_FCB CHANGED TO O_NEXT_FCB TO PREVENT CONFLICT WITH
                     4243 ; IDENTICAL LABEL ELSEWHERE
                     4244
EA00                 4245 __OPEN_FILE
EA00 FDE5            4246         PUSH     IY              ;SAVE REGISTERS
EA02 E5              4247         PUSH     HL
EA03 DDE5            4248         PUSH     IX
EA05 D5              4249         PUSH     DE
                     4250
EA06 F5              4251         PUSH     AF              ;SAVE DEVICE NUMBER
EA07 C5              4252         PUSH     BC              ;SAVE MODE
                     4253
                     4254 ; FIRST, WE HAVE TO LOCATE A FREE FCB.
EA08 DD2AFDFD        4255         LD       IX,[FCB_HEAD_ADDR] ;GET POINTER TO FIRST FCB'S HEAD
EA0C 110023          4256         LD       DE,FCB_LENGTH
EA0F DD19            4257         ADD      IX,DE           ; SKIP OVER IT -- BELONGS TO SYSTEM
                     4258
EA11 FD2AFDFF        4259         LD       IY,[FCB_DATA_ADDR] ;GET POINTER TO FIRST FCB'S BODY
EA15 110400          4260         LD       DE,1024
EA18 FD19            4261         ADD      IY,DE           ; SKIP OVER IT TOO
                     4262
EA1A 0601            4263         LD       B,1             ;SET FCB NUMBER
EA1C                 4264 O_NEXT_FCB
EA1C DD7E18          4265         LD       A,[IX+FCB_MODE] ;GET THE MODE BYTE FROM FCB
EA1F B7              4266         OR       A               ;CHECK IT
EA20 281C            4267         JR       Z,GOT_ONE       ;BRANCH IF THIS ONE IS FREE
                     4268
EA22 110023          4269         LD       DE,FCB_LENGTH   ;ELSE  SKIP OVER IT
EA25 DD19            4270         ADD      IX,DE
EA27 110400          4271         LD       DE,1024
EA2A FD19            4272         ADD      IY,DE
                     4273
EA2C 04              4274         INC      B               ;INC THE FCB NUMBER
EA2D 78              4275         LD       A,B
EA2E FE03            4276         CP       NUM_FCBS        ;ARE THERE MORE FCBS TO CHECK?
EA30 38EA            4277         JR       C,O_NEXT_FCB    ;LOOP UNTIL WE'VE SEEN THEM ALL
EA32                 4278 NO_FCBS
EA32                 4279         POP      BC              ;RESTORE R
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    EA33 F1          4280         POP     AF
    EA34 D1          4281         POP     DE
    EA35 DDE1        4282         POP     IX
    EA37 E1          4283         POP     HL
    EA38 FDE1        4284         POP     IY
                     4285
    EA3A 3E07        4286         LD      A,NO_FCB_ERR     ;SET THE ERROR CODE
    EA3C B7          4287         OR      A                ;SET THE CONDITIONS
    EA3D C9          4288         RET
                     4289
    EA3E             4290 GOT_ONE
    EA3E FD22FE08    4291         LD      [BUF_START],IY   ;SAVE THE FCB BUFFER'S ADDRESS
                     4292
    EA42 F1          4293         POP     AF               ;GET THE MODE
    EA43 DD7718      4294         LD      [IX+FCB_MODE],A  ;PUT IT INTO FCB
    EA46 F1          4295         POP     AF               ;GET THE DEVICE
    EA47 DD7717      4296         LD      [IX+FCB_DEVICE],A ;PUT IT INTO FCB
                     4297
    EA4A C5          4298         PUSH    BC               ;SAVE THE FCB NUMBER
                     4299
    EA4B E5          4300         PUSH    HL
    EA4C D1          4301         POP     DE               ; ADDRESS OF NAME STRING
                     4302
    EA4D DDE5        4303         PUSH    IX               ;POINT TO FCB NAME (ASSUME OFFSET = 0)
    EA4F E1          4304         POP     HL
    EA50 DD7E17      4305         LD      A,[IX+FCB_DEVICE] ;GET DEVICE NUMBER
    EA53 CDE618      4306         CALL    __QUERY_FILE     ;GET THE FILE'S DIR ENTRY
    EA56 C2EAF8      4307         JP      NZ,OP_ERR        ;BRANCH IF THERE WAS AN ERROR
                     4308                                  ;*** SET PARAMS ***
    EA59 CDF0D9      4309         CALL    MODE_CHECK       ;SEE IF MODE MATCHES ATTRIBUTES
    EA5C C2EAF8      4310         JP      NZ,OP_ERR        ;BRANCH IF NOT
                     4311
    EA5F DD7E13      4312         LD      A,[IX+FCB_USED_LENGTH]  ;CALC. & SET LAST BLOCK
    EA62 DD8600      4313         ADD     A,[IX+FCB_FIRST_BLOCK]
    EA65 DD771D      4314         LD      [IX+FCB_LAST_BLOCK],A
    EA68 DD7E0E      4315         LD      A,[IX+FCB_FIRST_BLOCK+1]
    EA6B DD8E14      4316         ADC     A,[IX+FCB_USED_LENGTH+1]
    EA6E DD771E      4317         LD      [IX+FCB_LAST_BLOCK+1],A
    EA71 DD7E0F      4318         LD      A,[IX+FCB_FIRST_BLOCK+2]
    EA74 CE00        4319         ADC     A,0
    EA76 DD771F      4320         LD      [IX+FCB_LAST_BLOCK+2],A
    EA79 DD7E10      4321         LD      A,[IX+FCB_FIRST_BLOCK+3]
    EA7C CE00        4322         ADC     A,0
    EA7E DD7720      4323         LD      [IX+FCB_LAST_BLOCK+3],A
                     4324
    EA81 DD7E1D      4325         LD      A,[IX+FCB_LAST_BLOCK+0]
    EA84 D601        4326         SUB     1
    EA86 DD771D      4327         LD      [IX+FCB_LAST_BLOCK+0],A
                     4328
    EA89 DD7E1E      4329         LD      A,[IX+FCB_LAST_BLOCK+1]
    EA8C DE00        4330         SBC     A,0
    EA8E DD771E      4331         LD      [IX+FCB_LAST_BLOCK+1],A
                     4332
    EA91 DD7E1F      4333         LD      A,[IX+FCB_LAST_BLOCK+2]
    EA94 DE00        4334         SBC     A,0
    EA96 DD771F      4335         LD      [IX+FCB_LAST_BLOCK+2],A
                     4336
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    EA99 DD7E20      4337          LD      A,[IX+FCB_LAST_BLOCK+3]
    EA9C DE00        4338          SBC     A,0
    EA9E DD7720      4339          LD      [IX+FCB_LAST_BLOCK+3],A
                     4340
    EAA1 ED5BFE08    4341          LD      DE,[BUF_START]  ;INIT BUFFER POINTER
    EAA5 DD7321      4342          LD      [IX+FCB_POINTER],E
    EAA8 DD7222      4343          LD      [IX+FCB_POINTER+1],D
                     4344
    EAAB DD7E0D      4345          LD      A,[IX+FCB_FIRST_BLOCK] ;SET BLOCK TO FIRST_BLOCK
    EAAE DD7719      4346          LD      [IX+FCB_BLOCK],A
    EAB1 DD7E0E      4347          LD      A,[IX+FCB_FIRST_BLOCK+1]
    EAB4 DD771A      4348          LD      [IX+FCB_BLOCK+1],A
    EAB7 DD7E0F      4349          LD      A,[IX+FCB_FIRST_BLOCK+2]
    EABA DD771B      4350          LD      [IX+FCB_BLOCK+2],A
    EABD DD7E10      4351          LD      A,[IX+FCB_FIRST_BLOCK+3]
    EAC0 DD771C      4352          LD      [IX+FCB_BLOCK+3],A
                     4353
    EAC3 DD7E18      4354          LD      A,[IX+FCB_MODE] ;LOOK AT THE MODE
    EAC6 E607        4355          AND     MODE_MODE
    EAC8 FE02        4356          CP      MODE_WRITE      ;IS IT WRITE MODE?
    EACA 2827        4357          JR      Z,OPEN5         ;JUMP IF SO -- NO PRE-READ NECESSARY
                     4358
    EACC             4359 READ_TIMEOUT:
                     4360 *                       If FCB_USED_LENGTH = 1 (file size of one block)
    EACC AF          4361          XOR     A                       ;load Acc with zero
    EACD DDBE14      4362          CP      [IX+FCB_USED_LENGTH+1]  ;High order must be zero
    EAD0 200A        4363          JR      NZ,NOT_SIZE_1           ;If not then past bitset
    EAD2 3C          4364          INC     A                       ;Inc Acc to a 1
    EAD3 DDBE13      4365          CP      [IX+FCB_USED_LENGTH]    ;Compare with low order
    EAD6 2004        4366          JR      NZ,NOT_SIZE_1           ;If not equal then exit
                     4367
                     4368 *                       Then set last block bit in mode
    EAD8 DDCB18FE    4369          SET     MODE_LAST_BLOCK_BIT,[IX+FCB_MODE] ;Set it
                     4370 *                       Endif
    EADC             4371 NOT_SIZE_1:
                     4372
    EADC DD7E17      4373          LD      A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
    EADF 2AFE08      4374          LD      HL,[BUF_START]  ;GET BUFFER ADDRESS
    EAE2 DD5E19      4375          LD      E,[IX+FCB_BLOCK] ;GET BLOCK NUMBER
    EAE5 DD561A      4376          LD      D,[IX+FCB_BLOCK+1]
    EAE8 DD4E1B      4377          LD      C,[IX+FCB_BLOCK+2]
    EAEB DD461C      4378          LD      B,[IX+FCB_BLOCK+3]
    EAEE CDF17B      4379          CALL    READ_BLOCK      ;READ THE FIRST BLOCK
                     4380
    EAF1 2005        4381          JR      NZ,OP_ERR       ; IT'S AN ERROR
                     4382
    EAF3             4383 OPEN5
    EAF3 AF          4384          XOR     A               ;SHOW NO ERROR
    EAF4 C1          4385          POP     BC              ;GET THE FILE NUMBER IN B
    EAF5 78          4386          LD      A,B
                     4387
    EAF6 1805        4388          JR      REST_REGS
                     4389 ;        POP     DE              ;RESTORE REGISTERS
                     4390 ;        POP     IX
                     4391 ;        POP     HL
                     4392 ;        POP     IY
                     4393 ;        RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4394
    EAF8             4395 OP_ERR
    EAF8 DD361800    4396         LD      [IX+FCB_MODE],MODE_UNUSED ;MAKE THE FCB UNUSED AGAIN
                     4397
    EAFC C1          4398         POP     BC              ;RESTORE REGISTERS
    EAFD             4399 REST_REGS:
    EAFD D1          4400         POP     DE
    EAFE DDE1        4401         POP     IX
    EB00 E1          4402         POP     HL
    EB01 FDE1        4403         POP     IY
    EB03 C9          4404         RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4406 ;---------------------------------------------------------------------
                     4407 ;
                     4408 ; __CLOSE_FILE -- does the necessary clean-up and marks the FCB
                     4409 ;                as unused.
                     4410 ;
                     4411 ; ENTRY PARAMETERS -- file number in A.
                     4412 ;
                     4413 ; EXIT PARAMETERS -- if no error -- Z = 1; A = 0
                     4414 ;                    if error -- Z = 0; A = error code
                     4415 ;
                     4416 ;---------------------------------------------------------------------
                     4417
    EB04             4418 __CLOSE_FILE
    EB04 DDE5        4419         PUSH    IX              ;SAVE REGISTERS
    EB06 E5          4420         PUSH    HL
    EB07 D5          4421         PUSH    DE
    EB08 C5          4422         PUSH    BC
                     4423
    EB09 B7          4424         OR      A               ;CHECK FILE NUMBER
    EB0A 2850        4425         JR      Z,C_ERROR       ;BRANCH IF ZERO
    EB0C FE03        4426         CP      NUM_FCBS        ;CHECK FOR MAX RANGE
    EB0E 304C        4427         JR      NC,C_ERROR      ;BRANCH IF TOO LARGE
                     4428
    EB10 47          4429         LD      B,A             ;PUT IT INTO B
    EB11 DD2AFDFD    4430         LD      IX,[FCB_HEAD_ADDR] ;GET ADDRESS OF 0TH FCB
    EB15 2AFDFF      4431         LD      HL,[FCB_DATA_ADDR] ;GET ADDRESS OF 0TH BUFFER
    EB18             4432 CLOSE2
    EB18 110023      4433         LD      DE,FCB_LENGTH   ;SET OFFSET
    EB1B DD19        4434         ADD     IX,DE           ;ADVANCE POINTER TO NEXT FCB
    EB1D 110400      4435         LD      DE,1024
    EB20 19          4436         ADD     HL,DE           ;ADVANCE POINTER TO NEXT BUFFER
    EB21 10F5        4437         DJNZ    CLOSE2          ;UNTIL WE GET TO THE ONE WE WANT
                     4438
    EB23 DD7E18      4439         LD      A,[IX+FCB_MODE] ;LOOK AT THE MODE BYTE
    EB26 B7          4440         OR      A
    EB27 2833        4441         JR      Z,C_ERROR       ;BRANCH IF FCB NOT IN USE
                     4442
    EB29 E640        4443         AND     MODE_DIRTY      ;ZERO UNWANTED BITS
    EB2B 2825        4444         JR      Z,CLOSE3        ;BRANCH IF BUFFER NOT DIRTY
                     4445
    EB2D DD22FD73    4446         LD      [FILE_NAME_ADDR],IX   ; POINTER TO FILE NAME IN FCB
    EB31             4447 WRITE_TIMEOUT:
    EB31 DD7E17      4448         LD      A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
    EB34 DD5E19      4449         LD      E,[IX+FCB_BLOCK] ;GET THE BLOCK NUMBER
    EB37 DD561A      4450         LD      D,[IX+FCB_BLOCK+1]
    EB3A DD4E1B      4451         LD      C,[IX+FCB_BLOCK+2]
    EB3D DD461C      4452         LD      B,[IX+FCB_BLOCK+3]
                     4453                                 ;HL STILL HAS ADDRESS OF BUFFER
    EB40 CDF1E6      4454         CALL    WRITE_BLOCK      ;FLUSH THE BLOCK TO TAPE
    EB43 2020        4455         JR      NZ,C_ERROR2     ;BRANCH IF ERROR
                     4456
    EB45             4457 CLOSE4:
    EB45 DD7E17      4458         LD      A,[IX+FCB_DEVICE]  ; GET DEVICE ID
    EB48 DDE5        4459         PUSH    IX
    EB4A E1          4460         POP     HL
    EB4B ED5BFD73    4461         LD      DE,[FILE_NAME_ADDR]
                     4462
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    EB4F CDE651      4463         CALL     __SET_FILE
                     4464
    EB52             4465 CLOSE3
    EB52 DD361800    4466         LD       [IX+FCB_MODE],MODE_UNUSED ;MARK THE FCB EMPTY
                     4467
    EB56 C1          4468         POP      BC              ;RESTORE REGISTERS
    EB57 D1          4469         POP      DE
    EB58 E1          4470         POP      HL
    EB59 DDE1        4471         POP      IX
    EB5B C9          4472         RET
    EB5C             4473 C_ERROR
    EB5C C1          4474         POP      BC              ;RESTORE REGISTERS
    EB5D D1          4475         POP      DE
    EB5E E1          4476         POP      HL
    EB5F DDE1        4477         POP      IX
    EB61 3E09        4478         LD       A,BAD_FNUM_ERR  ;SET THE ERROR
    EB63 B7          4479         OR       A               ;SET THE CONDITIONS
    EB64 C9          4480         RET
    EB65             4481 C_ERROR2
    EB65 C1          4482         POP      BC              ;RESTORE REGISTERS
    EB66 D1          4483         POP      DE
    EB67 E1          4484         POP      HL
    EB68 DDE1        4485         POP      IX
    EB6A B7          4486         OR       A               ;SET CONDITIONS
    EB6B C9          4487         RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4489 ;---------------------------------------------------------------
                     4490 ;
                     4491 ; __RESET_FILE -- rewinds the file back to the first byte.
                     4492 ;
                     4493 ; ENTRY PARAMETERS: file number in A
                     4494 ;
                     4495 ; EXIT PARAMETERS: no error -- Z = 1; A = 0
                     4496 ;                  if error -- Z = 0; A = error code
                     4497 ;
                     4498 ;---------------------------------------------------------------
                     4499
    EB6C             4500 __RESET_FILE
    EB6C DDE5        4501         PUSH    IX              ;SAVE REGISTERS
    EB6E D5          4502         PUSH    DE
    EB6F C5          4503         PUSH    BC
    EB70 E5          4504         PUSH    HL
                     4505
    EB71 B7          4506         OR      A               ;CHECK FILE NUMBER
    EB72 CAEC07      4507         JP      Z,R_ERROR       ;BRANCH IF ZERO
    EB75 FE03        4508         CP      NUM_FCBS        ;CHECK FOR MAX RANGE
    EB77 D2EC07      4509         JP      NC,R_ERROR      ;BRANCH IF TOO LARGE
                     4510
    EB7A 47          4511         LD      B,A             ;PUT IT INTO B
    EB7B DD2AFDFD    4512         LD      IX,[FCB_HEAD_ADDR] ;GET ADDRESS OF 0TH FCB
    EB7F 2AFDFF      4513         LD      HL,[FCB_DATA_ADDR] ;GET ADDRESS OF 0TH BUFFER
    EB82             4514 RES1
    EB82 110023      4515         LD      DE,FCB_LENGTH   ;SET OFFSET
    EB85 DD19        4516         ADD     IX,DE           ;ADVANCE POINTER TO NEXT FCB
    EB87 110400      4517         LD      DE,1024         ;SET BUFFER SIZE
    EB8A 19          4518         ADD     HL,DE           ;ADVANCE POINTER TO NEXT BUFFER
    EB8B 10F5        4519         DJNZ    RES1            ;UNTIL WE GET TO THE ONE WE WANT
                     4520
    EB8D DD7E18      4521         LD      A,[IX+FCB_MODE] ;LOOK AT THE MODE BYTE
    EB90 B7          4522         OR      A
    EB91 2874        4523         JR      Z,R_ERROR       ;BRANCH IF FCB NOT IN USE
                     4524
    EB93 E640        4525         AND     MODE_DIRTY      ;ZERO UNWANTED BITS
    EB95 2818        4526         JR      Z,RES2          ;BRANCH IF BUFFER NOT DIRTY
                     4527
    EB97             4528 TIMED_OUT:
    EB97 DD7E17      4529         LD      A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
    EB9A DD5E19      4530         LD      E,[IX+FCB_BLOCK] ;GET THE BLOCK NUMBER
    EB9D DD561A      4531         LD      D,[IX+FCB_BLOCK+1]
    EBA0 DD4E1B      4532         LD      C,[IX+FCB_BLOCK+2]
    EBA3 DD461C      4533         LD      B,[IX+FCB_BLOCK+3]
                     4534                                 ;HL STILL HAS ADDRESS OF BUFFER
    EBA6 CDF1E6      4535         CALL    WRITE_BLOCK     ;FLUSH THE BLOCK TO TAPE
    EBA9 2065        4536         JR      NZ,R_ERROR2     ;BRANCH IF ERROR
                     4537
    EBAB             4538 RESET_BIT:
    EBAB DDCB18B6    4539         RES     MODE_DIRTY_BIT,[IX+FCB_MODE] ;CLEAR THE DIRTY BIT
    EBAF             4540 RES2
    EBAF DDCB18BE    4541         RES     MODE_LAST_BLOCK_BIT,[IX+FCB_MODE]  ; BACK TO FIRST BLOCK
    EBB3 DD7E0D      4542         LD      A,[IX+FCB_FIRST_BLOCK] ;SET BLOCK TO FIRST_BLOCK
    EBB6 DD7719      4543         LD      [IX+FCB_BLOCK],A
    EBB9 DD7E        4544         LD      A,[IX+FCB_FIRST_BLOCK+1]
    EBBC             4545         LD      [IX+FCB_BLOCK+1],A
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    EBBF DD7E0F      4546          LD      A,[IX+FCB_FIRST_BLOCK+2]
    EBC2 DD771B      4547          LD      [IX+FCB_BLOCK+2],A
    EBC5 DD7E10      4548          LD      A,[IX+FCB_FIRST_BLOCK+3]
    EBC8 DD771C      4549          LD      [IX+FCB_BLOCK+3],A
                     4550
    EBCB DD7521      4551          LD      [IX+FCB_POINTER+0],L                    ;reset pointer for new (first) block
    EBCE DD7422      4552          LD      [IX+FCB_POINTER+1],H
                     4553
    EBD1 DD7E18      4554          LD      A,[IX+FCB_MODE] ;GET THE MODE AGAIN
    EBD4 E607        4555          AND     MODE_MODE       ;LOOK AT MODE TYPE ONLY
    EBD6 FE02        4556          CP      MODE_WRITE      ;ARE WE WRITING?
    EBD8 2012        4557          JR      NZ,READ_IT       ;NOPE, NEED TO PRE-READ
                     4558
    EBDA DD361500    4559          LD      [IX+FCB_LAST_COUNT],0
    EBDE DD361600    4560          LD      [IX+FCB_LAST_COUNT+1],0
                     4561
    EBE2 DD361301    4562          LD      [IX+FCB_USED_LENGTH],1
    EBE6 DD361400    4563          LD      [IX+FCB_USED_LENGTH+1],0
                     4564
    EBEA 1814        4565          JR      NO_READ
                     4566
    EBEC             4567 READ_IT:
    EBEC DD7E17      4568          LD      A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
    EBEF DD5E19      4569          LD      E,[IX+FCB_BLOCK] ;GET THE BLOCK NUMBER
    EBF2 DD561A      4570          LD      D,[IX+FCB_BLOCK+1]
    EBF5 DD4E18      4571          LD      C,[IX+FCB_BLOCK+2]
    EBF8 DD461C      4572          LD      B,[IX+FCB_BLOCK+3]
                     4573                                  ;HL STILL HAS ADDRESS OF BUFFER
    EBFB CDF17B      4574          CALL    READ_BLOCK      ;READ THE FIRST BLOCK
                     4575
    EBFE 2010        4576          JR      NZ,R_ERROR2     ;BRANCH ON ERROR
                     4577
    EC00             4578 NO_READ
    EC00 E1          4579          POP     HL              ;RESTORE THE REGISTERS
    EC01 C1          4580          POP     BC
    EC02 D1          4581          POP     DE
    EC03 DDE1        4582          POP     IX
    EC05 AF          4583          XOR     A               ;SHOW NO ERROR
    EC06 C9          4584          RET
                     4585
    EC07             4586 R_ERROR
    EC07 E1          4587          POP     HL              ;RESTORE THE REGISTERS
    EC08 C1          4588          POP     BC
    EC09 D1          4589          POP     DE
    EC0A DDE1        4590          POP     IX
    EC0C 3E09        4591          LD      A,BAD_FNUM_ERR  ;SHOW THE ERROR
    EC0E B7          4592          OR      A               ;SET CONDITIONS
    EC0F C9          4593          RET
    EC10             4594 R_ERROR2
    EC10 E1          4595          POP     HL              ;RESTORE REGISTERS
    EC11 C1          4596          POP     BC
    EC12 D1          4597          POP     DE
    EC13 DDE1        4598          POP     IX
    EC15 B7          4599          OR      A               ;SET CONDITION
    EC16 C9          4600          RET
                     4601 ;
                     4602 ;=========================================================================
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4603 ;************************************************************************
                     4604 ;************************************************************************
                     4605 ;
                     4606
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4608
                     4609  ;Packed some code to allow room for
                     4610
                     4611        ;GLB    __READ_FILE,__WRITE_FILE
                     4612        ;EXT    BUF_START,BUF_END,USER_BUF,BYTES_REQ,BYTES_TO_GO
                     4613        ;EXT    FNUM
                     4614        ;EXT    FCB_HEAD_ADDR,FCB_DATA_ADDR
                     4615        ;EXT    READ_BLOCK,WRITE_BLOCK
                     4616        ;EXT    MODE_CHECK
                     4617
                     4618 ;---------------------------------------------------------------
                     4619 ;
                     4620 ; __READ_FILE -- read some data from a file into the user's buffer.
                     4621 ;
                     4622 ; ENTRY PARAMETERS: device number in A; buffer address in HL;
                     4623 ;                   desired number of bytes in BC.
                     4624 ;
                     4625 ; EXIT PARAMETERS: no errors -- Z = 1; A = 0
                     4626 ;                   if errors -- Z = 0; A = error code;
                     4627 ;                   BC = number of bytes transferred
                     4628 ;
                     4629 ;---------------------------------------------------------------
                     4630
                     4631 ; GET_OUT CHANGED TO R_GET_OUT TO PREVENT CONFLICT WITH
                     4632 ; IDENTICAL LABEL ELSEWHERE
                     4633
EC17                 4634 __READ_FILE
EC17 C5              4635        PUSH    BC              ;SAVE REGISTERS
EC18 D5              4636        PUSH    DE
EC19 E5              4637        PUSH    HL
EC1A DDE5            4638        PUSH    IX
                     4639
EC1C 22FE06          4640        LD      [USER_BUF],HL   ;SAVE USER'S BUFFER ADDRESS
EC1F ED43FE02        4641        LD      [BYTES_REQ],BC  ;SAVE REQUESTED COUNT
EC23 32FE01          4642        LD      [FNUM],A        ;SAVE FILE NUMBER
EC26 ED43FE04        4643        LD      [BYTES_TO_GO],BC ;INIT COUNTER (COUNTS DOWN)
                     4644
EC2A B7              4645        OR      A               ;CHECK FILE NUMBER
EC2B CAED7F          4646        JP      Z,READ_ERR2     ;BRANCH IF ZERO
EC2E FE03            4647        CP      NUM_FCBS        ;CHECK FOR MAX RANGE
EC30 D2ED7F          4648        JP      NC,READ_ERR2    ;BRANCH IF TOO LARGE
                     4649
EC33 47              4650        LD      B,A             ;PUT IT INTO B
EC34 DD2AFDFD        4651        LD      IX,[FCB_HEAD_ADDR] ;GET ADDRESS OF OTH FCB
EC38 2AFDFF          4652        LD      HL,[FCB_DATA_ADDR] ;GET ADDRESS OF OTH BUFFER
EC3B                 4653 READ1
EC3B 110023          4654        LD      DE,FCB_LENGTH   ;ADVANCE POINTER TO NEXT FCB
EC3E DD19            4655        ADD     IX,DE
EC40 110400          4656        LD      DE,1024         ;ADVANCE POINTER TO NEXT BUFFER
EC43 19              4657        ADD     HL,DE
EC44 10F5            4658        DJNZ    READ1           ;UNTIL WE GET TO THE ONE WE WANT
                     4659
EC46 22FE08          4660        LD      [BUF_START],HL  ;SAVE ADDRESS OF MY BUFFER
EC49 19              4661        ADD     HL,DE           ;POINT HL PAST END OF MY BUFFER
EC4A 22FE0A          4662        LD      [BUF_END],HL    ;SAVE IT TOO
                     4663
                     4664                                ;*** CHECK FOR MODE ***
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    EC4D DDE5        4665         PUSH    IX
    EC4F E1          4666         POP     HL
                     4667
    EC50 CDF009      4668         CALL    MODE_CHECK       ;SEE IF THE MODE IS OK
    EC53 C2ED81      4669         JP      NZ,READ_ERR1     ;BRANCH IF NOT
                     4670
    EC56             4671 READ3
    EC56 DDCB187E    4672         BIT     MODE_LAST_BLOCK_BIT,[IX+FCB_MODE] ;LOOK AT LAST BLOCK BIT
    EC5A C2ED2B      4673         JP      NZ,READ_LAST     ;BRANCH IF THIS IS THE LAST FILE BLOCK
                     4674
    EC5D 2AFE0A      4675         LD      HL,[BUF_END]     ;GET END ADDRESS OF BUFFER
    EC60 DD5E21      4676         LD      E,[IX+FCB_POINTER] ;GET POINTER INTO BUFFER
    EC63 DD5622      4677         LD      D,[IX+FCB_POINTER+1]
    EC66 B7          4678         OR      A
    EC67 ED52        4679         SBC     HL,DE            ;CALC. BYTES LEFT IN BUFFER
    EC69 2825        4680         JR      Z,_O_BYTES_LEFT2
                     4681
    EC6B 44          4682         LD      B,H              ;SAVE BYTES LEFT IN CASE WE NEED IT
    EC6C 4D          4683         LD      C,L
                     4684
    EC6D ED5BFE04    4685         LD      DE,[BYTES_TO_GO] ;GET BYTES LEFT TO SEND
    EC71 B7          4686         OR      A
    EC72 ED52        4687         SBC     HL,DE            ;COMPARE TO WHAT WE HAVE
    EC74 D2ECEA      4688         JP      NC,ENOUGH        ;BRANCH IF WE HAVE ENOUGH DATA HERE
                     4689
                     4690 ; IF WE FALL THROUGH HERE, WE KNOW WE WILL HAVE TO GET ANOTHER
                     4691 ; BLOCK AFTER THIS ONE.  FIRST, LET'S TRANSFER WHAT WE HAVE.
                     4692
    EC77 2AFE04      4693         LD      HL,[BYTES_TO_GO] ;UPDATE NUMBER OF BYTES TO GO
    EC7A B7          4694         OR      A
    EC7B ED42        4695         SBC     HL,BC            ;SUBTRACT NUM OF BYTES WE ARE SENDING HERE
    EC7D 22FE04      4696         LD      [BYTES_TO_GO],HL
                     4697
    EC80 DD6E21      4698         LD      L,[IX+FCB_POINTER] ;POINT INTO MY BUFFER
    EC83 DD6622      4699         LD      H,[IX+FCB_POINTER+1]
    EC86 ED5BFE06    4700         LD      DE,[USER_BUF]    ;POINT INTO THE CALLER'S BUFFER
                     4701                                  ;BC HAS COUNT FROM ABOVE
    EC8A EDB0        4702         LDIR                     ;GIVE WHAT WE HAVE
                     4703
    EC8C ED53FE06    4704         LD      [USER_BUF],DE    ;POINT TO NEXT SPOT
                     4705
    EC90             4706 _O_BYTES_LEFT2:
    EC90 2AFE08      4707         LD      HL,[BUF_START]   ;RESET THE POINTER
    EC93 DD7521      4708         LD      [IX+FCB_POINTER],L
    EC96 DD7422      4709         LD      [IX+FCB_POINTER+1],H
                     4710
    EC99 DD3419      4711         INC     [IX+FCB_BLOCK]   ;INC THE BLOCK NUMBER
    EC9C 2000        4712         JR      NZ,READ4         ;BRANCH IF NO CARRY OUT
    EC9E DD341A      4713         INC     [IX+FCB_BLOCK+1]
    ECA1 2008        4714         JR      NZ,READ4
    ECA3 DD341B      4715         INC     [IX+FCB_BLOCK+2]
    ECA6 2003        4716         JR      NZ,READ4
    ECA8 DD341C      4717         INC     [IX+FCB_BLOCK+3]
                     4718
    ECAB             4719 READ4
                     4720
    ECAB [           4721         LD      A,[IX+FCB_DEVICE]        'EVICE ID
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    ECAE 2AFE08      4722          LD      HL,[BUF_START]          ; GET ADDRESS OF USERS BUFFER
    ECB1 DD5E19      4723          LD      E,[IX+FCB_BLOCK]        ; GET BLOCK NUMBER
    ECB4 DD561A      4724          LD      D,[IX+FCB_BLOCK+1]
    ECB7 DD4E1B      4725          LD      C,[IX+FCB_BLOCK+2]
    ECBA DD461C      4726          LD      B,[IX+FCB_BLOCK+3]
                     4727
    ECBD CDF17B      4728          CALL    READ_BLOCK          ;READ THE NEXT BLOCK
                     4729
    ECC0 C2ED81      4730          JP      NZ,READ_ERR1        ;BRANCH IF ERROR
                     4731
    ECC3             4732 NOT_TAPE1:
    ECC3 DD7E20      4733          LD      A,[IX+FCB_LAST_BLOCK+3] ;SEE IF THIS IS THE LAST BLOCK
    ECC6 DDBE1C      4734          CP      [IX+FCB_BLOCK+3]
    ECC9 201C        4735          JR      NZ,JP_TO_READ3      ;BRANCH IF NOT
    ECCB DD7E1F      4736          LD      A,[IX+FCB_LAST_BLOCK+2]
    ECCE DDBE1B      4737          CP      [IX+FCB_BLOCK+2]
    ECD1 2014        4738          JR      NZ,JP_TO_READ3
    ECD3 DD7E1E      4739          LD      A,[IX+FCB_LAST_BLOCK+1]
    ECD6 DDBE1A      4740          CP      [IX+FCB_BLOCK+1]
    ECD9 200C        4741          JR      NZ,JP_TO_READ3
    ECDB DD7E1D      4742          LD      A,[IX+FCB_LAST_BLOCK]
    ECDE DDBE19      4743          CP      [IX+FCB_BLOCK]
    ECE1 2004        4744          JR      NZ,JP_TO_READ3
    ECE3 DDCB18FE    4745          SET     MODE_LAST_BLOCK_BIT,[IX+FCB_MODE] ;ELSE SET THE LAST BLOCK BIT
    ECE7             4746 JP_TO_READ3:
    ECE7 C3EC56      4747          JP      READ3               ;AND DEAL WITH THE NEXT BLOCK
                     4748
                     4749 ; ARRIVE HERE IF THIS BLOCK CONTAINS ENOUGH BYTES TO SATISFY
                     4750 ; THIS CALL.
                     4751
    ECEA             4752 ENOUGH:
    ECEA DD7E20      4753          LD      A,[IX+FCB_LAST_BLOCK+3] ;SEE IF THIS IS THE LAST BLOCK
    ECED DDBE1C      4754          CP      [IX+FCB_BLOCK+3]
    ECF0 2018        4755          JR      NZ,READ_LAST1       ;BRANCH IF NOT
    ECF2 DD7E1F      4756          LD      A,[IX+FCB_LAST_BLOCK+2]
    ECF5 DDBE1B      4757          CP      [IX+FCB_BLOCK+2]
    ECF8 2010        4758          JR      NZ,READ_LAST1
    ECFA DD7E1E      4759          LD      A,[IX+FCB_LAST_BLOCK+1]
    ECFD DDBE1A      4760          CP      [IX+FCB_BLOCK+1]
    ED00 2008        4761          JR      NZ,READ_LAST1
    ED02 DD7E1D      4762          LD      A,[IX+FCB_LAST_BLOCK]
    ED05 DDBE19      4763          CP      [IX+FCB_BLOCK]
    ED08 281D        4764          JR      Z,LAST_BLOCK
                     4765
    ED0A             4766 READ_LAST1:
    ED0A ED4BFE04    4767          LD      BC,[BYTES_TO_GO]    ; GET BYTE COUNT TO MOVE
                     4768
                     4769
    ED0E DD6E21      4770          LD      L,[IX+FCB_POINTER] ;POINT INTO MY BUFFER
    ED11 DD6622      4771          LD      H,[IX+FCB_POINTER+1]
    ED14 ED5BFE06    4772          LD      DE,[USER_BUF]       ;POINT TO THE USER'S BUFFER
    ED18 EDB0        4773          LDIR                        ;MOVE THE DATA
                     4774
    ED1A DD7521      4775          LD      [IX+FCB_POINTER],L ;UPDATE MY POINTER
    ED1D DD7422      4776          LD      [IX+FCB_POINTER+1],H
                     4777
                     4778
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   ED20 DDE1         4779          POP     IX              ;RESTORE REGISTERS
   ED22 E1           4780          POP     HL
   ED23 D1           4781          POP     DE
   ED24 C1           4782          POP     BC
   ED25 AF           4783          XOR     A               ;SHOW NO ERROR
   ED26 C9           4784          RET
                     4785
                     4786 ; ARRIVE HERE IF THIS IS THE LAST BLOCK OF THE FILE.
   ED27              4787 LAST_BLOCK:
   ED27 DDCB18FE     4788          SET     MODE_LAST_BLOCK_BIT,[IX+FCB_MODE]
                     4789
   ED2B              4790 READ_LAST
   ED2B 2AFE08       4791          LD      HL,[BUF_START]  ;POINT TO MY BUFFER
   ED2E DD5E15       4792          LD      E,[IX+FCB_LAST_COUNT] ;GET # BYTES IN IT
   ED31 DD5616       4793          LD      D,[IX+FCB_LAST_COUNT+1]
   ED34 19           4794          ADD     HL,DE           ;CALC ADDR OF LAST BYTE + 1
   ED35 DD5E21       4795          LD      E,[IX+FCB_POINTER] ;GET CURRENT POINTER
   ED38 DD5622       4796          LD      D,[IX+FCB_POINTER+1]
   ED3B B7           4797          OR      A               ;CLEAR CARRY
   ED3C ED52         4798          SBC     HL,DE           ;CALC # BYTES LEFT IN BUFFER
   ED3E 44           4799          LD      B,H             ;SAVE IT IN CASE IT'S NEEDED
   ED3F 4D           4800          LD      C,L
   ED40 282A         4801          JR      Z,_O_BYTES_LEFT
                     4802
   ED42 ED5BFE04     4803          LD      DE,[BYTES_TO_GO] ;GET REQUESTED COUNT
   ED46 B7           4804          OR      A               ;CLEAR CARRY (UGLY Z80)
   ED47 ED52         4805          SBC     HL,DE           ;SEE IF WE HAVE ENOUGH BYTES
   ED49 308F         4806          JR      NC,READ_LAST1   ;BRANCH IF SO -- NO SWEAT
                     4807
   ED4B 2AFE04       4808          LD      HL,[BYTES_TO_GO] ;UPDATE NUMBER OF BYTES TO GO
   ED4E B7           4809          OR      A
   ED4F ED42         4810          SBC     HL,BC           ;SUBTRACT NUM OF BYTES WE ARE SENDING HERE
   ED51 22FE04       4811          LD      [BYTES_TO_GO],HL
                     4812
   ED54 DD6E21       4813          LD      L,[IX+FCB_POINTER] ;GET POINTER TO MY BUFFER
   ED57 DD6622       4814          LD      H,[IX+FCB_POINTER+1]
   ED5A ED5BFE06     4815          LD      DE,[USER_BUF]   ;GET POINTER TO USER'S BUFFER
                     4816                                  ;BC HAS COUNT FROM ABOVE
   ED5E EDB0         4817          LDIR                    ;MOVE WHAT WE HAVE TO USER
                     4818
   ED60 2AFE02       4819          LD      HL,[BYTES_REQ]  ;GET # BYTES REQUESTED
   ED63 ED4BFE04     4820          LD      BC,[BYTES_TO_GO] ;GET # BYTES WE DIDN'T DO
   ED67 B7           4821          OR      A
   ED68 ED42         4822          SBC     HL,BC           ;CALC. # OF BYTES WE DID DO
   ED6A 44           4823          LD      B,H             ;PUT IT INTO BC
   ED6B 4D           4824          LD      C,L
                     4825
   ED6C              4826 _O_BYTES_LEFT:
   ED6C 3E0A         4827          LD      A,EOF_ERR       ;SHOW AN ERROR
   ED6E              4828 R_GET_OUT:
   ED6E DDE1         4829          POP     IX              ;RESTORE REGISTERS
   ED70 E1           4830          POP     HL
   ED71 D1           4831          POP     DE
   ED72 33           4832          INC     SP              ;DISCARD OLD BC
   ED73 33           4833          INC     SP
                     4834
   ED74 B7           4835          OR      A
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    ED75 C9          4836          RET
                     4837
    ED76             4838 RQ_STAT_ERR:
    ED76 DDE1        4839          POP     IX
    ED78 E1          4840          POP     HL
    ED79 D1          4841          POP     DE
    ED7A C1          4842          POP     BC
                     4843
    ED7B 3E15        4844          LD      A,RQ_TP_STAT_ERR
    ED7D B7          4845          OR      A
    ED7E C9          4846          RET
                     4847
    ED7F             4848 READ_ERR2:
    ED7F 3E09        4849          LD      A,BAD_FNUM_ERR
                     4850
    ED81             4851 READ_ERR1
    ED81 2AFE02      4852          LD      HL,[BYTES_REQ]  ;GET # BYTES REQUESTED
    ED84 ED4BFE04    4853          LD      BC,[BYTES_TO_GO] ;GET # BYTES WE DIDN'T DO
    ED88 B7          4854          OR      A
    ED89 ED42        4855          SBC     HL,BC           ;CALC. # OF BYTES WE DID DO
    ED8B 44          4856          LD      B,H             ;PUT IT INTO BC
    ED8C 4D          4857          LD      C,L
                     4858
    ED8D 18DF        4859          JR      R_GET_OUT
                     4860
                     4861 *        POP     IX              ;RESTORE REGISTERS
                     4862 *        POP     HL
                     4863 *        POP     DE
                     4864 *        INC     SP              ;DISCARD OLD BC
                     4865 *        INC     SP
                     4866 *        OR      A               ;SET CONDITION CODES
                     4867 *        RET
                     4868 *
                     4869 *READ_ERR2
                     4870 *        LD      HL,[BYTES_REQ]  ;GET # BYTES REQUESTED
                     4871 *        LD      BC,[BYTES_TO_GO] ;GET # BYTES WE DIDN'T DO
                     4872 *        OR      A
                     4873 *        SBC     HL,BC           ;CALC. # OF BYTES WE DID DO
                     4874 *        LD      B,H             ;PUT IT INTO BC
                     4875 *        LD      C,L
                     4876 *
                     4877 *        POP     IX              ;RESTORE REGISTERS
                     4878 *        POP     HL
                     4879 *        POP     DE
                     4880 *        INC     SP              ;DISCARD OLD BC
                     4881 *        INC     SP
                     4882 *        LD      A,BAD_FNUM_ERR  ;SET THE ERROR CODE
                     4883 *        OR      A               ;SET CONDITIONS
                     4884 *        RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4886 ;---------------------------------------------------------------
                     4887 ;
                     4888 ; __WRITE_FILE -- write some bytes to a file.
                     4889 ;
                     4890 ; ENTRY PARAMETERS: file number in A; byte count in BC; pointer
                     4891 ;                  to data in HL.
                     4892 ;
                     4893 ; EXIT PARAMETERS: no errors -- Z = 1; A = 0
                     4894 ;                   if errors -- Z = 0; A = error code
                     4895 ;
                     4896 ;---------------------------------------------------------------
                     4897
    ED8F             4898 __WRITE_FILE
    ED8F C5          4899          PUSH    BC              ;SAVE REGISTERS
    ED90 D5          4900          PUSH    DE
    ED91 E5          4901          PUSH    HL
    ED92 DDE5        4902          PUSH    IX
                     4903
    ED94 22FE06      4904          LD      [USER_BUF],HL   ;SAVE USER'S BUFFER ADDRESS
    ED97 ED43FE02    4905          LD      [BYTES_REQ],BC  ;SAVE REQUESTED COUNT
    ED9B 32FE01      4906          LD      [FNUM],A        ;SAVE FILE NUMBER
    ED9E ED43FE04    4907          LD      [BYTES_TO_GO],BC ;INIT COUNTER (COUNTS DOWN)
                     4908
    EDA2 B7          4909          OR      A               ;CHECK FILE NUMBER
    EDA3 CAEEBC      4910          JP      Z,WRITE_ERR2    ;BRANCH IF ZERO
    EDA6 FE03        4911          CP      NUM_FCBS        ;CHECK FOR MAX RANGE
    EDA8 D2EEBC      4912          JP      NC,WRITE_ERR2   ;BRANCH IF TOO LARGE
                     4913
    EDAB 47          4914          LD      B,A             ;PUT IT INTO B
    EDAC DD2AFDFD    4915          LD      IX,[FCB_HEAD_ADDR] ;GET ADDRESS OF 0TH FCB
    EDB0 110023      4916          LD      DE,FCB_LENGTH   ;SET OFFSET
    EDB3             4917 WRITE1
    EDB3 DD19        4918          ADD     IX,DE           ;ADVANCE POINTER TO NEXT FCB
    EDB5 10FC        4919          DJNZ    WRITE1          ;UNTIL WE GET TO THE ONE WE WANT
                     4920
    EDB7 47          4921          LD      B,A             ;POINT HL AT MY BUFFER
    EDB8 110400      4922          LD      DE,1024
    EDBB 2AFDFF      4923          LD      HL,[FCB_DATA_ADDR]
    EDBE             4924 WRITE2
    EDBE 19          4925          ADD     HL,DE
    EDBF 10FD        4926          DJNZ    WRITE2
                     4927
    EDC1 22FE08      4928          LD      [BUF_START],HL  ;SAVE ADDRESS OF MY BUFFER
    EDC4 19          4929          ADD     HL,DE           ;POINT HL PAST END OF MY BUFFER
    EDC5 22FE0A      4930          LD      [BUF_END],HL   ;SAVE IT TOO
                     4931
                     4932                                  ;*** CHECK FOR MODE ***
    EDC8 DDE5        4933          PUSH    IX
    EDCA E1          4934          POP     HL
                     4935
    EDCB CDF0D9      4936          CALL    MODE_CHECK
    EDCE C2EEBE      4937          JP      NZ,WRITE_ERR1   ;JUMP IF WRONG MODE
                     4938
    EDD1             4939 WRITE3
    EDD1 DD7E14      4940          LD      A,[IX+FCB_USED_LENGTH+1]
    EDD4 DDBE12      4941          CP      [IX+FCB_MAX_LENGTH+1]
    EDD7 3?          4942          JR      C,SET_UP
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    EDD9 DD7E13      4943          LD      A,[IX+FCB_USED_LENGTH+0]
    EDDC DDBE11      4944          CP      [IX+FCB_MAX_LENGTH]
    EDDF 3804        4945          JR      C,SET_UP
                     4946
    EDE1 DDCB18FE    4947          SET     MODE_LAST_BLOCK_BIT,[IX+FCB_MODE]
    EDE5             4948 SET_UP:
    EDE5 2AFE0A      4949          LD      HL,[BUF_END]     ;GET LAST ADDR + 1 OF BUFFER
    EDE8 DD5E21      4950          LD      E,[IX+FCB_POINTER] ;GET CURRENT POINTER
    EDEB DD5622      4951          LD      D,[IX+FCB_POINTER+1]
    EDEE B7          4952          OR      A                ;CLEAR CARRY
    EDEF ED52        4953          SBC     HL,DE            ;CALC. ROOM LEFT IN BUFFER
    EDF1 44          4954          LD      B,H              ;SAVE IT IN CASE WE NEED IT
    EDF2 4D          4955          LD      C,L
    EDF3 ED5BFE04    4956          LD      DE,[BYTES_TO_GO] ;GET BYTES LEFT TO TRANSFER
    EDF7 B7          4957          OR      A
    EDF8 ED52        4958          SBC     HL,DE            ;SEE IF WE HAVE ROOM FOR IT ALL
    EDFA D2EE93      4959          JP      NC,ROOM_OK       ;BRANCH IF YES
                     4960
                     4961 ; WE FALL THROUGH HERE IF THE NUMBER OF BYTES TO BE WRITTEN
                     4962 ; IS MORE THAN THE SPACE WE HAVE IN THE CURRENT BLOCK.
                     4963 ; FILL THE BLOCK, WRITE IT OUT, THEN DECIDE WHAT TO DO.
                     4964
    EDFD 78          4965          LD      A,B              ; CHECK IF ANY BYTES AVAILABLE
    EDFE B1          4966          OR      C
    EDFF 281B        4967          JR      Z,CHECK_LAST_BLOCK  ; NOPE, EMPTY BUFFER FIRST
                     4968
    EE01 2AFE04      4969          LD      HL,[BYTES_TO_GO] ;GET THE COUNT
    EE04 B7          4970          OR      A
    EE05 ED42        4971          SBC     HL,BC            ;SUB THE ROOM IN BUFFER TO UPDATE IT
    EE07 22FE04      4972          LD      [BYTES_TO_GO],HL
                     4973
    EE0A DD5E21      4974          LD      E,[IX+FCB_POINTER] ;GET POINTER INTO BUFFER
    EE0D DD5622      4975          LD      D,[IX+FCB_POINTER+1]
    EE10 2AFE06      4976          LD      HL,[USER_BUF]    ;GET POINTER TO USER DATA
                     4977                                   ;BC STILL HAS COUNT FROM ABOVE
    EE13 ED80        4978          LDIR                     ;MOVE DATA INTO THE BUFFER
                     4979
    EE15 22FE06      4980          LD      [USER_BUF],HL    ; SAVE WHERE WE LEFT OFF
                     4981
                     4982
    EE18 DDCB18F6    4983          SET     MODE_DIRTY_BIT,[IX+FCB_MODE] ;SET THE DIRTY BIT
                     4984
    EE1C             4985 CHECK_LAST_BLOCK:
    EE1C DDCB187E    4986          BIT     MODE_LAST_BLOCK_BIT,[IX+FCB_MODE] ;CHECK FOR LAST BLOCK
    EE20 2809        4987          JR      Z,MORE_FILE      ;JUMP IF THIS ISN'T LAST BLOCK
                     4988
                     4989 ; IF THIS IS THE LAST BLOCK (AND IT'S NOW FULL) THERE'S NOTHING
                     4990 ; MORE WE CAN DO.  WE LEAVE THE DATA IN PLACE (IT WILL GET
                     4991 ; FLUSHED WHEN THE FILE IS CLOSED) AND REPORT AN EOF ERROR,
                     4992 ; MEANING THAT WE COULDN'T WRITE ALL THE DATA.
                     4993
    EE22 DDE1        4994          POP     IX               ;RESTORE REGISTERS
    EE24 E1          4995          POP     HL
    EE25 D1          4996          POP     DE
    EE26 C1          4997          POP     BC
    EE27 3E0A        4998          LD      A,EOF_ERR        ;SHOW THE ERROR
    EE29 B7          4999          OR      A                ;SET CONDITIONS
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

   EE2A C9          5000          RET
                    5001
                    5002 ; WE COME HERE IF WE HAVE A FULL BUFFER AND THERE'S MORE
                    5003 ; DATA TO WRITE AND THERE ARE MORE BLOCKS TO THE FILE.
                    5004
   EE2B             5005 MORE_FILE
   EE2B DD3413      5006          INC     [IX+FCB_USED_LENGTH]    ; INCREMENT SECTORS WRITTEN TO
   EE2E 2003        5007          JR      NZ,MORE_FILE1
   EE30 DD3414      5008          INC     [IX+FCB_USED_LENGTH+1]
   EE33             5009 MORE_FILE1:
   EE33 DD7E17      5010          LD      A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
   EE36 2AFE08      5011          LD      HL,[BUF_START]  ;GET BUFFER ADDRESS
   EE39 DD5E19      5012          LD      E,[IX+FCB_BLOCK] ;GET BLOCK NUMBER
   EE3C DD561A      5013          LD      D,[IX+FCB_BLOCK+1]
   EE3F DD4E1B      5014          LD      C,[IX+FCB_BLOCK+2]
   EE42 DD461C      5015          LD      B,[IX+FCB_BLOCK+3]
   EE45 CDF1E6      5016          CALL    WRITE_BLOCK     ;WRITE OUT THE FULL BLOCK
                    5017
   EE48 C2EEBE      5018          JP      NZ,WRITE_ERR1   ;BRANCH IF ERROR
                    5019
   EE4B             5020 CLEAR_DIRTY_BIT:
   EE4B DDCB18B6    5021          RES     MODE_DIRTY_BIT,[IX+FCB_MODE] ;CLEAR THE DIRTY BIT
   EE4F 2AFE08      5022          LD      HL,[BUF_START]  ;RESET THE BUFFER POINTER
   EE52 DD7521      5023          LD      [IX+FCB_POINTER],L
   EE55 DD7422      5024          LD      [IX+FCB_POINTER+1],H
                    5025
   EE58 DD361500    5026          LD      [IX+FCB_LAST_COUNT],0
   EE5C DD361600    5027          LD      [IX+FCB_LAST_COUNT+1],0
                    5028
   EE60 DD3419      5029          INC     [IX+FCB_BLOCK]  ;INC THE BLOCK NUMBER
   EE63 2000        5030          JR      NZ,WRITE4       ;BRANCH IF NO CARRY OUT
   EE65 DD341A      5031          INC     [IX+FCB_BLOCK+1]
   EE68 2008        5032          JR      NZ,WRITE4
   EE6A DD341B      5033          INC     [IX+FCB_BLOCK+2]
   EE6D 2003        5034          JR      NZ,WRITE4
   EE6F DD341C      5035          INC     [IX+FCB_BLOCK+3]
                    5036
   EE72             5037 WRITE4
   EE72 DD7E18      5038          LD      A,[IX+FCB_MODE] ;LOOK AT THE MODE BYTE
   EE75 E607        5039          AND     MODE_MODE       ;GET RID OF EXTRA BITS
   EE77 FE03        5040          CP      MODE_UPDATE     ;DO WE HAVE TO PRE_READ THE NEXT BLOCK?
   EE79 C2EDD1      5041          JP      NZ,WRITE3       ;LOOP TO TOP IF NOT
                    5042
   EE7C             5043 WRITE_AGAIN:
   EE7C DD7E17      5044          LD      A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
   EE7F 2AFE08      5045          LD      HL,[BUF_START]  ;GET BUFFER ADDRESS
   EE82 DD5E19      5046          LD      E,[IX+FCB_BLOCK] ;GET BLOCK NUMBER
   EE85 DD561A      5047          LD      D,[IX+FCB_BLOCK+1]
   EE88 DD4E1B      5048          LD      C,[IX+FCB_BLOCK+2]
   EE8B DD461C      5049          LD      B,[IX+FCB_BLOCK+3]
   EE8E CDF17B      5050          CALL    READ_BLOCK      ;PRE-READ THE NEXT FILE BLOCK
                    5051
   EE91 202B        5052          JR      NZ,WRITE_ERR1   ;BRANCH IF ERROR
                    5053
                    5054
                    5055 ; ARRIVE HERE IF THE DATA BEING WRITTEN WILL NOT FILL THE BUFFER.
                    5056 ; EVERYTHING IS VERY EASY -- JUST COPY IT TO THE USER'S ADDRESS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5057 ; INTO THE BUFFER AND GO AWAY.
                     5058
    EE93             5059 ROOM_OK
    EE93 ED4BFE04    5060          LD      BC,[BYTES_TO_GO] ;GET THE BYTE COUNT
    EE97 DD6E15      5061          LD      L,[IX+FCB_LAST_COUNT]
    EE9A DD6616      5062          LD      H,[IX+FCB_LAST_COUNT+1]
    EE9D 09          5063          ADD     HL,BC
    EE9E DD7515      5064          LD      [IX+FCB_LAST_COUNT],L
    EEA1 DD7416      5065          LD      [IX+FCB_LAST_COUNT+1],H
    EEA4 DD5E21      5066          LD      E,[IX+FCB_POINTER] ;GET POINTER INTO BUFFER
    EEA7 DD5622      5067          LD      D,[IX+FCB_POINTER+1]
    EEAA 2AFE06      5068          LD      HL,[USER_BUF]   ;GET POINTER TO USER'S DATA
    EEAD ED80        5069          LDIR                    ;MOVE THE DATA
                     5070
    EEAF DD7321      5071          LD      [IX+FCB_POINTER],E ;UPDATE THE POINTER
    EEB2 DD7222      5072          LD      [IX+FCB_POINTER+1],D
    EEB5 DDCB18F6    5073          SET     MODE_DIRTY_BIT,[IX+FCB_MODE] ;SET THE DIRTY BIT
                     5074
    EEB9             5075 ROOM1_OK:
    EEB9 AF          5076          XOR     A
    EEBA 1803        5077          JR      PAST_WRITE_ERR
    EEBC             5078 WRITE_ERR2:
    EEBC 3E09        5079          LD      A,BAD_FNUM_ERR
    EEBE             5080 WRITE_ERR1:
    EEBE B7          5081          OR      A
    EEBF             5082 PAST_WRITE_ERR:
    EEBF DDE1        5083          POP     IX              ;RESTORE REGISTERS
    EEC1 E1          5084          POP     HL
    EEC2 D1          5085          POP     DE
    EEC3 C1          5086          POP     BC
    EEC4 C9          5087          RET
                     5088 *WRITE_ERR1
                     5089 *        POP     IX              ;RESTORE REGISTERS
                     5090 *        POP     HL
                     5091 *        POP     DE
                     5092 *        POP     BC
                     5093 *        OR      A               ;SET CONDITIONS
                     5094 *        RET
                     5095 *WRITE_ERR2
                     5096 *        POP     IX              ;RESTORE REGISTERS
                     5097 *        POP     HL
                     5098 *        POP     DE
                     5099 *        POP     BC
                     5100 *        LD      A,BAD_FNUM_ERR  ;SHOW AN ERROR
                     5101 *        OR      A               ;SET CONDITIONS
                     5102 *        RET
                     5103 ;
                     5104 ;*****************************************************************
                     5105 ;*****************************************************************
                     5106 ;*****************************************************************
                     5107 ;
                     5108
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5110         ;GLB      __SET_DATE,__GET_DATE
                     5111         ;EXT      EOS_YEAR,EOS_MONTH,EOS_DAY
                     5112
                     5113
                     5114 ;---------------------------------------------------------------
                     5115 ;
                     5116 ; __SET_DATE -- Set the current date.
                     5117 ;
                     5118 ; ENTRY PARAMETERS: B = day (1..31)
                     5119 ;                    C = month (1..12)
                     5120 ;                    D = year (83..99)
                     5121 ;
                     5122 ; EXIT PARAMETERS: none
                     5123 ;
                     5124 ;---------------------------------------------------------------
                     5125
   EEC5              5126 __SET_DATE
   EEC5 F5           5127         PUSH     AF              ;SAVE THE CURRENT A
                     5128
   EEC6 78           5129         LD       A,B             ;STORE THE DAY
   EEC7 32FDE2       5130         LD       [EOS_DAY],A
   EECA 79           5131         LD       A,C             ;STORE THE MONTH
   EECB 32FDE1       5132         LD       [EOS_MONTH],A
   EECE 7A           5133         LD       A,D             ;STORE THE YEAR
   EECF 32FDE0       5134         LD       [EOS_YEAR],A
                     5135
   EED2 F1           5136         POP      AF              ;RESTORE A
   EED3 C9           5137         RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5139 ;---------------------------------------------------------------
                     5140 ;
                     5141 ; __GET_DATE -- Reads the current date.
                     5142 ;
                     5143 ; ENTRY PARAMETERS: none
                     5144 ;
                     5145 ; EXIT PARAMETERS: no errors -- Z = 1; A = 0
                     5146 ;                               B = day
                     5147 ;                               C = month
                     5148 ;                               D = year
                     5149 ;
                     5150 ;                    error (date never set) -- Z = 0; A = error code
                     5151 ;                               B =  0
                     5152 ;                               C =  0
                     5153 ;                               D =  0
                     5154 ;
                     5155 ;---------------------------------------------------------------
                     5156
EED4                 5157 __GET_DATE
EED4 3AFDE2          5158         LD      A,[EOS_DAY]
EED7 47              5159         LD      B,A             ;GET THE DAY
EED8 3AFDE1          5160         LD      A,[EOS_MONTH]
EED8 4F              5161         LD      C,A             ;GET THE MONTH
EEDC 3AFDE0          5162         LD      A,[EOS_YEAR]
EEDF 57              5163         LD      D,A             ;GET THE YEAR
                     5164
EEE0 B0              5165         OR      B
EEE1 B1              5166         OR      C               ;CHECK FOR ALL = 0
EEE2 2802            5167         JR      Z,GET_ERROR     ;BRANCH IF ERROR
EEE4 AF              5168         XOR     A               ;SHOW NO ERROR
EEE5 C9              5169         RET
                     5170
EEE6                 5171 GET_ERROR
EEE6 3E04            5172         LD      A,NO_DATE_ERR   ;SHOW AN ERROR OCURRED
EEE8 B7              5173         OR      A               ;CLEAR Z FLAG
EEE9 C9              5174         RET
                     5175 ;
                     5176 ;***************************************************************
                     5177 ;***************************************************************
                     5178 ;***************************************************************
                     5179 ;
                     5180
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5182 ;
                     5183 ;  Rev. 1   07oct515p  RPD  made __DELETE_FILE check for protected file
                     5184 ;
                     5185
                     5186         ;GLB     __FMGR_INIT,SCAN_FOR_FILE,STRCMP,MODE_CHECK
                     5187         ;GLB     READ_BLOCK,WRITE_BLOCK,BASECMP,__WRITE_BLOCK
                     5188         ;GLB     __MODE_CHECK,__READ_BLOCK,__SCAN_FOR_FILE
                     5189         ;EXT     __RD_1_BLOCK,__WR_1_BLOCK,__REQUEST_STATUS
                     5190         ;EXT     __QUERY_FILE,__SET_FILE
                     5191         ;EXT     QUERY_BUFFER,FILENAME_CMPS
                     5192         ;EXT     FCB_HEAD_ADDR,FCB_DATA_ADDR
                     5193         ;EXT     USER_NAME,BUF_START
                     5194         ;EXT     RETRY_COUNT
                     5195         ;EXT     FILE_COUNT,MOD_FILE_COUNT
                     5196
                     5197
                     5198 ;-----------------------------------------------------------------
                     5199 ;
                     5200 ; __FMGR_INIT: sets up everything to start with.
                     5201 ;
                     5202 ; ENTRY PARAMETERS: DE contains address to place FCB buffers,
                     5203 ;                   and HL contains address to place FCB headers.
                     5204 ;
                     5205 ; EXIT PARAMETERS: none.
                     5206 ;
                     5207 ;-----------------------------------------------------------------
                     5208
EEEA                 5209 __FMGR_INIT
EEEA ED53FDFF        5210         LD      [FCB_DATA_ADDR],DE ;SAVE BUFFER ADDRESS
EEEE 22FDFD          5211         LD      [FCB_HEAD_ADDR],HL ;SAVE HEADER ADDRESS
                     5212
EEF1 C5              5213         PUSH    BC              ;SAVE REGISTERS
EEF2 D5              5214         PUSH    DE
EEF3 DDE5            5215         PUSH    IX
                     5216
EEF5 0603            5217         LD      B,NUM_FCBS      ;SET TO # FCBS
EEF7 110023          5218         LD      DE,FCB_LENGTH   ;SET FOR ADDITION
EEFA DD2AFDFD        5219         LD      IX,[FCB_HEAD_ADDR] ;GET ADDR OF OTH FCB
                     5220
EEFE                 5221 INIT_LOOP
EEFE DD361800        5222         LD      [IX+FCB_MODE],MODE_UNUSED ;SHOW THE FCB IS EMPTY
EF02 DD19            5223         ADD     IX,DE           ;ADVANCE POINTER TO NEXT FCB
EF04 10F8            5224         DJNZ    INIT_LOOP       ;LOOP FOR ALL FCBS
                     5225
EF06 DDE1            5226         POP     IX              ;RESTORE REGISTERS
EF08 D1              5227         POP     DE
EF09 C1              5228         POP     BC
EF0A C9              5229         RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5231 ;---------------------------------------------------------------
                     5232 ;
                     5233 ; SCAN_FOR_FILE -- Look in the directory for a file name.
                     5234 ;                 Reads each block of the directory into a buffer
                     5235 ;                 (in the system's FCB) and scans through for a
                     5236 ;                 match with the caller's string.
                     5237 ;
                     5238 ; ENTRY PARAMETERS: address of name string in HL; device # in A
                     5239 ;          ASSUMES FILENAME_CMP SET UP FOR FILE COMPARES
                     5240 ;          0 - USE STRCMP ,  NOT 0 - USE BASECMP  FOR FILE NAME COMPARISIONS
                     5241 ;
                     5242 ; EXIT PARAMETERS: if match found -- Z = 1; A = 0; file's start
                     5243 ;                 block in BCDE; directory block left in buffer,
                     5244 ;                 address of entry left in FCB_POINTER.
                     5245 ;
                     5246 ;                    if no match found -- Z = 0; A = error code;
                     5247 ;                 BCDE = junk; other stuff = junk.
                     5248 ;
                     5249 ;---------------------------------------------------------------
                     5250
EF08                 5251 __SCAN_FOR_FILE:
EF08                 5252 SCAN_FOR_FILE
EF08 E5              5253          PUSH    HL              ;SAVE REGISTERS
EF0C DDE5            5254          PUSH    IX
EF0E FDE5            5255          PUSH    IY
                     5256
EF10 22FE10          5257          LD      [USER_NAME],HL  ;SAVE USER'S STRING ADDRESS
                     5258
EF13 DD2AFDFD        5259          LD      IX,[FCB_HEAD_ADDR] ;GET ADDRESS OF SYSTEM'S FCB
                     5260
                     5261 ; First, set up as much of the FCB as we can.
EF17 DD7717          5262          LD      [IX+FCB_DEVICE],A ;SET THE DEVICE ADDRESS PASSED TO US
                     5263
EF1A DD361901        5264          LD      [IX+FCB_BLOCK],1 ;SET BLOCK TO 1
EF1E DD361A00        5265          LD      [IX+FCB_BLOCK+1],0
EF22 DD361B00        5266          LD      [IX+FCB_BLOCK+2],0
EF26 DD361C00        5267          LD      [IX+FCB_BLOCK+3],0
                     5268
EF2A ED58FDFF        5269          LD      DE,[FCB_DATA_ADDR] ;SET BUFFER ADDRESS
EF2E DD7321          5270          LD      [IX+FCB_POINTER],E
EF31 DD7222          5271          LD      [IX+FCB_POINTER+1],D
                     5272
EF34                 5273 RD_TMEOUT:
EF34 21FDD4          5274          LD      HL,FILE_COUNT
EF37 3600            5275          LD      [HL],0
                     5276
EF39 DD7E17          5277          LD      A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
EF3C 2AFDFF          5278          LD      HL,[FCB_DATA_ADDR] ;GET BUFFER ADDRESS
EF3F DD5E19          5279          LD      E,[IX+FCB_BLOCK] ;GET BLOCK NUMBER
EF42 DD561A          5280          LD      D,[IX+FCB_BLOCK+1]
EF45 DD4E1B          5281          LD      C,[IX+FCB_BLOCK+2]
EF48 DD461C          5282          LD      B,[IX+FCB_BLOCK+3]
EF4B CDF17B          5283          CALL    READ_BLOCK      ;READ FIRST BLOCK OF DIRECTORY
EF4E C2FO26          5284          JP      NZ,SCAN_ERROR   ;BRANCH IF ERROR
                     5285
EF51                 5286 GET_BUFF_ADDR:
                     5287
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    EF51 E5          5288         PUSH    HL              ;GET BUFFER ADDRESS INTO IY TOO
    EF52 FDE1        5289         POP     IY
                     5290
    EF54 FD7E0C      5291         LD      A,[IY+VOL_DIRSIZE] ;GET THE DIRECTORY SIZE
    EF57 E67F        5292         AND     07FH            ;ZERO PERM BIT
                     5293 ;       INC     A               ;ADD 1 'CAUSE WE START IN BLOCK 1
                     5294 ;       DEC     A               ;SUB 1 'CAUSE OF PICKET FENCE PROBLEM
    EF59 DD771D      5295         LD      [IX+FCB_LAST_BLOCK],A ;SET LAST BLOCK IN FCB
    EF5C DD361E00    5296         LD      [IX+FCB_LAST_BLOCK+1],0 ;(ALL MSBYTES WILL BE 0)
    EF60 DD361F00    5297         LD      [IX+FCB_LAST_BLOCK+2],0
    EF64 DD362000    5298         LD      [IX+FCB_LAST_BLOCK+3],0
                     5299
    EF68 CDF035      5300         CALL    CHECK_IF_DIRECTORY      ; CHECK IF DIRECTORY EXIST
    EF6B C2F026      5301         JP      NZ,SCAN_ERROR
                     5302
    EF6E DD7E21      5303         LD      A,[IX+FCB_POINTER] ;ADVANCE POINTER TO NEXT ENTRY
    EF71 C61A        5304         ADD     A,DIR_ENT_LENGTH
    EF73 DD7721      5305         LD      [IX+FCB_POINTER],A
    EF76 DD7E22      5306         LD      A,[IX+FCB_POINTER+1]
    EF79 CE00        5307         ADC     A,0
    EF7B DD7722      5308         LD      [IX+FCB_POINTER+1],A
                     5309
    EF7E 21FDD4      5310         LD      HL,FILE_COUNT
    EF81 34          5311         INC     [HL]
                     5312
    EF82 0626        5313         LD      B,ENT_PER_BLOCK-1 ;SET DIR ENTRY COUNT FOR FIRST BLOCK
    EF84 184B        5314         JR      SCAN2           ;JUMP INTO THE LOOP
                     5315
    EF86             5316 SCAN1
    EF86 DD3419      5317         INC     [IX+FCB_BLOCK]  ;INC THE BLOCK NUMBER
                     5318                                 ;WE CAN ASSUME THERE'S NO CARRY OUT
                     5319
                     5320 ; SEE IF THERE IS MORE DIRECTORY: I.E. IS FCB_BLOCK > FCB_LAST_BLOCK?
                     5321 ; IF TRUE, GO TO NO_ENT.
                     5322
    EF89 DD7E20      5323         LD      A,[IX+FCB_LAST_BLOCK+3]
    EF8C DDBE1C      5324         CP      [IX+FCB_BLOCK+3]
    EF8F DAF02C      5325         JP      C,NO_ENT        ;BRANCH IF NO MORE DIRECTORY
    EF92 DD7E1F      5326         LD      A,[IX+FCB_LAST_BLOCK+2]
    EF95 DDBE1B      5327         CP      [IX+FCB_BLOCK+2]
    EF98 DAF02C      5328         JP      C,NO_ENT        ;BRANCH IF NO MORE DIRECTORY
    EF9B DD7E1E      5329         LD      A,[IX+FCB_LAST_BLOCK+1]
    EF9E DDBE1A      5330         CP      [IX+FCB_BLOCK+1]
    EFA1 DAF02C      5331         JP      C,NO_ENT        ;BRANCH IF NO MORE DIRECTORY
    EFA4 DD7E1D      5332         LD      A,[IX+FCB_LAST_BLOCK]
    EFA7 DDBE19      5333         CP      [IX+FCB_BLOCK]
    EFAA DAF02C      5334         JP      C,NO_ENT        ;BRANCH IF NO MORE DIRECTORY
                     5335
    EFAD             5336 RD_TMEOUT1:
    EFAD DD7E17      5337         LD      A,[IX+FCB_DEVICE] ;GET THE DEVICE NUMBER
    EFB0 2AFDFF      5338         LD      HL,[FCB_DATA_ADDR] ;GET BUFFER ADDRESS
    EFB3 DD5E19      5339         LD      E,[IX+FCB_BLOCK] ;GET BLOCK NUMBER
    EFB6 DD561A      5340         LD      D,[IX+FCB_BLOCK+1]
    EFB9 DD4E1B      5341         LD      C,[IX+FCB_BLOCK+2]
    EFBC DD461C      5342         LD      B,[IX+FCB_BLOCK+3]
    EFBF CDE1?       5343         CALL    READ_BLOCK      ;READ THE NEXT BLOCK
    EFC2 C           5344         JP      NZ,SCAN_ERROR   ;BRANCH IF
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5345
EFC5                 5346 GET_BUFF_ADDR1:
                     5347
                     5348
EFC5 ED5BFDFF        5349          LD      DE,[FCB_DATA_ADDR] ;GET BUFFER ADDRESS
EFC9 DD7321          5350          LD      [IX+FCB_POINTER],E ;PUT IT INTO THE FCB
EFCC DD7222          5351          LD      [IX+FCB_POINTER+1],D
                     5352
EFCF 0627            5353          LD      B,ENT_PER_BLOCK ;SET ENTRY COUNT
                     5354
EFD1                 5355 SCAN2
EFD1 21FDD4          5356          LD      HL,FILE_COUNT
EFD4 34              5357          INC     [HL]
                     5358
EFD5 DD5E21          5359          LD      E,[IX+FCB_POINTER] ;GET POINTER INTO DE
EFD8 DD5622          5360          LD      D,[IX+FCB_POINTER+1]
                     5361
EFDB 2E0C            5362          LD      L,DIR_ATTR          ; OFFSET (FOR ATTR) INTO DIR ENTRY
EFDD 2600            5363          LD      H,0                 ; NOT USED
EFDF 19              5364          ADD     HL,DE               ; SET UP HL TO POINT TO IT
EFE0 CB46            5365          BIT     ATTR_HOLE_BIT,[HL]  ; IS IT THE HOLE?
EFE2 2048            5366          JR      NZ,NO_ENT           ; YUP, DON'T BOTHER SEARCHING ANY MORE
EFE4 CB56            5367          BIT     ATTR_DEL_BIT,[HL]   ; IS THIS FILE DELETED?
EFE6 2013            5368          JR      NZ,FILE_DELETED     ; YUP, DON'T EVEN BOTHER CHECK FILE NAMES
                     5369
EFE8 2AFE10          5370          LD      HL,[USER_NAME]
                     5371
EFEB 3AFDD8          5372          LD      A,[FILENAME_CMPS]   ;
EFEE B7              5373          OR      A                   ; IS IT ZERO?
EFEF 2805            5374          JR      Z,FULL_FILENAME     ; YUP, COMPARE THE FULL FILE NAME
EFF1 CDF054          5375          CALL    BASECMP             ; NOPE, COMPARE JUST THE BASE
EFF4 1803            5376          JR      ERROR_CHECK
EFF6                 5377 FULL_FILENAME:
EFF6 CDF053          5378          CALL    STRCMP              ; COMPARE THE FULL FILE NAME (BASE PLUS EXTENSION)
EFF9                 5379 ERROR_CHECK:
EFF9 2815            5380          JR      Z,SCAN3             ;BRANCH IF YES
                     5381
EFFB                 5382 FILE_DELETED:
EFFB DD7E21          5383          LD      A,[IX+FCB_POINTER] ;ADVANCE POINTER TO NEXT ENTRY
EFFE C61A            5384          ADD     A,DIR_ENT_LENGTH
F000 DD7721          5385          LD      [IX+FCB_POINTER],A
F003 DD7E22          5386          LD      A,[IX+FCB_POINTER+1]
F006 CE00            5387          ADC     A,0
F008 DD7722          5388          LD      [IX+FCB_POINTER+1],A
                     5389
F00B 10C4            5390          DJNZ    SCAN2               ;LOOP IF MORE ENTRIES TO CHECK IN THIS BLOCK
                     5391
F00D C3EF86          5392          JP      SCAN1               ;ELSE GET NEXT BLOCK
                     5393
F010                 5394 SCAN3
F010 D5              5395          PUSH    DE                  ; GET DIR ENTRY PTR INTO IY
F011 FDE1            5396          POP     IY
                     5397
F013 FD5E0D          5398          LD      E,[IY+DIR_START_BLOCK+0]
F016 FD560E          5399          LD      D,[IY+DIR_START_BLOCK+1]
F019 FD4E0F          5400          LD      C,[IY+DIR_START_BLOCK+2]
F01C FD4610          5401          LD      B,[IY+DIR_START_BLOCK+3]
```

```
LOCATION OBJECT CODE LINE      SOURCE LINE

                     5402
   F01F FDE1         5403          POP     IY              ;RESTORE REGISTERS
   F021 DDE1         5404          POP     IX
   F023 E1           5405          POP     HL
   F024 AF           5406          XOR     A               ;SHOW NO ERROR
   F025 C9           5407          RET
                     5408
   F026              5409 SCAN_ERROR
   F026 FDE1         5410          POP     IY              ;RESTORE REGISTERS
   F028 DDE1         5411          POP     IX
   F02A E1           5412          POP     HL
   F02B C9           5413          RET
                     5414
   F02C              5415 NO_ENT
   F02C FDE1         5416          POP     IY              ;RESTORE REGISTERS
   F02E DDE1         5417          POP     IX
   F030 E1           5418          POP     HL
   F031 3E05         5419          LD      A,NO_FILE_ERR
   F033 B7           5420          OR      A
   F034 C9           5421          RET
                     5422
                     5423
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5425
                     5426 ******************************************************************************
                     5427 ******************************************************************************
                     5428 *
                     5429 *     CHECK_IF_DIRECTORY VERIFIES THE EXISTENCE OF A DIRECTORY
                     5430 *               ON THE MEDIA IN THE DEVICE
                     5431 *
                     5432 *      INPUT:   IY  -  POINTS TO THE VOLUME ENTRY (FIRST ENTRY INTO THE DIRECTORY)
                     5433 *
                     5434 *      OUTPUT:  CONDITION FLAGS
                     5435 *                   Z - DIRECTORY EXISTS
                     5436 *                  NZ - NO DIRECTORY
                     5437 *                         A = ERROR CODE
                     5438 *
                     5439 *
                     5440 *               ALL REGISTERS PRESERVED EXCEPT AF
                     5441 *
                     5442 ******************************************************************************
                     5443 ******************************************************************************
                     5444
                     5445          ;GLB     CHECK_IF_DIRECTORY
                     5446          ;EXT     DIRECTORY_CHECK
                     5447
F035                 5448 CHECK_IF_DIRECTORY:
F035 FDE5            5449          PUSH     IY
F037 C5              5450          PUSH     BC
F038 E5              5451          PUSH     HL
                     5452
F039 21F3E7          5453          LD       HL,DIRECTORY_CHECK       ; 4 UNIQUE BYTES PLACE IN VOLUME ENTRY
                     5454                                            ; AT CREATION TIME (BYTES DEFINED IN EOS_MKDIR)
                     5455
F03C 0604            5456          LD       B,4                      ; COMPARE 4 BYTES
F03E                 5457 NEXT_CHECK:
F03E 7E              5458          LD       A,[HL]                   ; GET THE FIRST UNIQUE BYTE
F03F FDBE0D          5459          CP       [IY+VOL_DIR_CHECK]       ; SAME AS ON TAPE?
F042 2008            5460          JR       NZ,NOT_DIR               ; NOPE
                     5461
F044 23              5462          INC      HL                       ; YUP, CHECK NEXT BYTE
F045 FD23            5463          INC      IY
F047 10F5            5464          DJNZ     NEXT_CHECK
                     5465
F049 AF              5466          XOR      A                        ; DIRECTORY EXISTS
                     5467
F04A                 5468 GET_OUT_HERE:
F04A E1              5469          POP      HL
F04B C1              5470          POP      BC
F04C FDE1            5471          POP      IY
                     5472
F04E C9              5473          RET
                     5474
F04F                 5475 NOT_DIR:
F04F 3E18            5476          LD       A,NO_DIR_ERR
F051 18F7            5477          JR       GET_OUT_HERE
                     5478
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5480 ;---------------------------------------------------------------
                     5481 ;
                     5482 ; STRCMP -- compare two strings (file names).
                     5483 ;
                     5484 ; ENTRY PARAMETERS: address of name string in HL; address of
                     5485 ;                   name within dir entry in DE.  NOTE: string
                     5486 ;                   pointed to by HL must be terminated with ETX.
                     5487 ;                   Maximum of NAME_LENGTH characters are compared.
                     5488 ;
                     5489 ; EXIT PARAMETERS: Z = 1; A = 0 if match or
                     5490 ;                   Z = 0; A = error code if no match.
                     5491 ;
                     5492 ;---------------------------------------------------------------
                     5493
           <F054>    5494 BASECMP: EQU     $+1             ; EXECUTES SCF (37H) - COMPARE ONLY THE BASE
                     5495                                   ;  OF THE FILE NAME
F053                 5496 STRCMP:                          ; "AND" INSTRUCTION CLEARS CARRY - COMPARE BASE &
F053 E637            5497          AND     037H            ;     EXTENSION
F055 E5              5498          PUSH    HL              ;PRESERVE REG'S
F056 D5              5499          PUSH    DE
F057 C5              5500          PUSH    BC
F058 F5              5501          PUSH    AF              ; SAVE CARRY FLAGGING EXECUTED ABOVE
                     5502
F059 060C            5503          LD      B,NAME_LENGTH   ;INIT MAX COUNT
F05B                 5504 NEXT_CHAR
F05B 7E              5505          LD      A,[HL]          ;GET CHAR FROM STRING
F05C FE03            5506          CP      ETX             ;HAVE WE REACHED END OF STRING?
F05E 2817            5507          JR      Z,EOS           ;BRANCH IF YES
F060 1A              5508          LD      A,[DE]          ;GET CHAR FROM DIR ENTRY
F061 BE              5509          CP      [HL]            ;COMPARE TO STRING
F062 2007            5510          JR      NZ,MATCHLESS    ;BRANCH IF DIFFERENT
F064 23              5511          INC     HL              ;ELSE INC. POINTERS
F065 13              5512          INC     DE
F066 10F3            5513          DJNZ    NEXT_CHAR       ;LOOP IF MORE CHARS TO LOOK AT
F068 C1              5514          POP     BC              ; POP AF INTO BC
F069 1812            5515          JR      MATCH           ;ALL 12 MATCH IF WE GET HERE!
F06B                 5516 MATCHLESS
F06B F1              5517          POP     AF              ; COMPARE BASE OR BASE&EXTENSION
F06C 3014            5518          JR      NC,NO_MATCH     ; NEEDED TO COMPARE THE WHOLE FILE NAME
                     5519                                  ; ONLY COMPARE THE BASE
F06E 23              5520          INC     HL              ;POINT TO NEXT CHAR OF NAME STRING
F06F 13              5521          INC     DE              ;  AND DIR ENTRY NAME
F070 7E              5522          LD      A,[HL]
F071 FE03            5523          CP      ETX             ; IS IT ETX?
F073 200D            5524          JR      NZ,NO_MATCH     ; NOPE, DOES NOT MATCH
F075 1801            5525          JR      AROUND_POP
                     5526
F077                 5527 EOS
F077 C1              5528          POP     BC              ; AF INTO BC TO PRESERVE AF
F078                 5529 AROUND_POP:
F078 1A              5530          LD      A,[DE]          ;LOOK AT CHAR IN DIR ENTRY
                     5531                                  ;DLS(8/29/83) CHANGED SPACE TO ETX.
F079 FE03            5532          CP      ETX             ;END OF FILE NAME?
F07B 2005            5533          JR      NZ,NO_MATCH     ;BRANCH IF NOT
F07D                 5534 MATCH
F07D C1              5535          POP     BC              ;RESTORE REG'S
F07E                 5536          POP     DE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F07F E1          5537          POP     HL
    F080 AF          5538          XOR     A               ;SHOW THE MATCH
    F081 C9          5539          RET
    F082             5540 NO_MATCH
    F082 C1          5541          POP     BC              ;RESTORE REG'S
    F083 D1          5542          POP     DE
    F084 E1          5543          POP     HL
    F085 3E08        5544          LD      A,MATCH_ERR
    F087 B7          5545          OR      A
    F088 C9          5546          RET                     ;RETURN WITH NZ CONDITION
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5548
                     5549 ************************************************************************
                     5550 *
                     5551 *
                     5552 *   THIS ROUTINE CHECKS TO SEE IF THE FILENAME (POINTED TO BY HL)
                     5553 *   IS ALREADY OPEN AND EXIST IN THE FCB
                     5554 *
                     5555 *               INPUT:   HL      POINTER TO FILENAME (ETX TERMINATED)
                     5556 *
                     5557 *               OUTPUT:  CONDITION FLAGS
                     5558 *                        NZ      NOT IN FCB (NOT OPENED)
                     5559 *                                A = ERROR CODE
                     5560 *                                HL = ORIGINAL POINTER TO FILENAME
                     5561 *                         Z      IN FCB
                     5562 *                                A = FILE NUMBER
                     5563 *                                B = MODE
                     5564 *                               HL = POINTER TO START OF FCB  *
                     5565 *
                     5566 ************************************************************************
                     5567    ..
                     5568 ; GET_OUT              U_GET_OUT
                     5569 ; NEXT_FCB CHANGED TO U_NEXT_FCB TO PREVENT CONFLICT WITH
                     5570 ; IDENTICAL LABEL ELSEWHERE
                     5571
                     5572       ;GLB     __CHECK_FCB
                     5573       ;EXT     FILE_NUMBR
                     5574
FO89                 5575 __CHECK_FCB:
FO89 DDE5            5576       PUSH     IX
FO8B D5              5577       PUSH     DE
                     5578 ;     PUSH     BC
FO8C E5              5579       PUSH     HL
                     5580
FO8D 2AFDFD          5581       LD       HL,[FCB_HEAD_ADDR]
FO90 AF              5582       XOR      A                          ; ZERO OUT A
FO91                 5583 U_NEXT_FCB:
FO91 3C              5584       INC      A                          ; CONTAINS THE FILE NUMBER
FO92 32FDD7          5585       LD       [FILE_NUMBR],A             ; (THE POSITION  ON THE TAPE)
                     5586
FO95 110023          5587       LD       DE,FCB_LENGTH              ; NUMBER OF BYTES IN EACH FCB
FO98 19              5588       ADD      HL,DE                      ; POINT TO NEXT FCB ENTRY
                     5589
FO99 E5              5590       PUSH     HL
FO9A DDE1            5591       POP      IX                         ; IX POINTS TO START OF FCB
                     5592
FO9C DD7E18          5593       LD       A,[IX+FCB_MODE]            ; GET MODE
FO9F E607            5594       AND      MODE_MODE
FOA1 FE00            5595       CP       0                          ; IT IS UNUSED?
                     5596
FOA3 2812            5597       JR       Z,END_FCB                  ; YUP, LOOK AT NEXT FCB
                     5598
FOA5 EB              5599       EX       DE,HL                      ; DE = POINTER TO FCB NAME STRING
FOA6 E1              5600       POP      HL
FOA7 E5              5601       PUSH     HL                         ; HL = POINTER TO FILENAME TO SEARCH FOR
                     5602
FOA8 D5              5603       PUSH     DE                         ; DE = POINTER TO FCB FILE NAME
                     5604
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    FOA9 01000C      5605         LD      BC,NAME_LENGTH          ; NO. OF CHARACTERS IN NAME
                     5606                                         ; HL = FILE NAME STRING
    FOAC             5607 NEXT_CHAR1:
    FOAC 1A          5608         LD      A,[DE]                  ; DE = FCB FILE NAME
    FOAD FE03        5609         CP      ETX                     ; IS IT ETX?
                     5610
    FOAF 2813        5611         JR      Z,ITS_ETX1              ; YUP
                     5612
    FOB1 EDA1        5613         CPI                             ; DO CHARS MATCH ([HL] = A)?
                     5614
    FOB3 13          5615         INC     DE
    FOB4 28F6        5616         JR      Z,NEXT_CHAR1            ; YUP,CHECK NEXT CHAR OF FILE NAME
                     5617
    FOB6 E1          5618         POP     HL                      ; pointer to FCB
    FOB7             5619 END_FCB:
    FOB7 3AFDD7      5620         LD      A,[FILE_NUMBR]          ; ARE WE AT 2ND FILE(FCB)?
    FOBA FE02        5621         CP      NUM_FCBS-1              ; NO. OF FCB'S - SYSTEM FCB
                     5622
    FOBC 20D3        5623         JR      NZ,U_NEXT_FCB           ; NOPE, NOT YET,
                     5624
    FOBE             5625 NO_MATCH1:
    FOBE E1          5626         POP     HL                      ; HL = ORIGINAL POINTER TO FILE NAME
                     5627
    FOBF 3E05        5628         LD      A,NO_FILE_ERR           ; NO MATCH FOUND
    FOC1 B7          5629         OR      A
    FOC2 1811        5630         JR      GET_OUT_ERR
                     5631
    FOC4             5632 ITS_ETX1:
    FOC4 7E          5633         LD      A,[HL]                  ; GET ASCII CHAR
    FOC5 FE03        5634         CP      ETX                     ; IS IT ETX?
    FOC7 E1          5635         POP     HL                      ;*pointer to FCB
    FOC8 20F4        5636         JR      NZ,NO_MATCH1            ; NOPE
                     5637
                     5638 ;       PUSH    HL                      ; copy pointer to FCB so can index to mode
                     5639 ;       POP     IX
                     5640
    FOCA DD7E18      5641         LD      A,[IX+FCB_MODE]         ; GET MODE THIS FILE WAS ORIGINALLY
    FOCD E607        5642         AND     MODE_MODE               ; OPENED FOR AND MASK OFF FLAG BITS
    FOCF 47          5643         LD      B,A                     ; PASS IT BACK IN B
                     5644
    FOD0 AF          5645         XOR     A                       ; INDICATE Z CONDITION
    FOD1 3AFDD7      5646         LD      A,[FILE_NUMBR]          ; GET FILE NUMBER
                     5647
    FOD4             5648 U_GET_OUT:
    FOD4 D1          5649         POP     DE                      ;*clean off pointer to name
                     5650 ;       POP     BC
    FOD5             5651 GET_OUT_ERR:
    FOD5 D1          5652         POP     DE
    FOD6 DDE1        5653         POP     IX
                     5654
    FOD8 C9          5655         RET
                     5656
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

                    5658
                    5659 ;---------------------------------------------------------------
                    5660 ;
                    5661 ; MODE_CHECK: sees if an attribute byte matches a permission request.
                    5662 ;
                    5663 ; ENTRY PARAMETERS: IX- FCB POINTER
                    5664 ;                   HL- DIR ENTRY
                    5665 ; EXIT PARAMETERS:  Z=1=OK
                    5666 ;                   Z=0=NO WAY
                    5667 ;                   A=TRASH
                    5668 ;                   IX AND HL PRESERVED
                    5669 ; DLS(8/30/83)
                    5670 ;---------------------------------------------------------------
                    5671
                    5672 ; OK CHANGED TO U_OK TO PREVENT CONFLICT WITH IDENTICAL
                    5673 ;LABEL ELSEWHERE
                    5674
   FOD9             5675 __MODE_CHECK:
   FOD9             5676 MODE_CHECK:
                    5677
   FOD9 FDE5        5678          PUSH    IY                      ;SAVE REG IY
   FOD8 E5          5679          PUSH    HL                      ;SAVE DIR ENTRY
   FODC DD7E18      5680          LD      A,[IX+FCB_MODE]         ;GET THE MODE THE USER PASSED IN
                    5681 *
                    5682 *  RANGE CHECK THIS REQUEST
                    5683 *
   FODF E607        5684          AND     MODE_MODE               ; AND OFF TEST BITS
   FOE1 FE00        5685          CP      0                       ;MAKE SURE HE'S IN RANGE
   FOE3 281F        5686          JR      Z,RANGE_NONO
   FOE5 FE05        5687          CP      MODE_MAX+1
   FOE7 3018        5688          JR      NC,RANGE_NONO
                    5689 *
   FOE9 21F10A      5690          LD      HL,ATTRIB_TBL           ;INDEX INTO THE MASK TABLE
   FOEC 85          5691          ADD     A,L
   FOED 6F          5692          LD      L,A
   FOEE 3001        5693          JR      NC,ADJUST
   FOF0 24          5694          INC     H
   FOF1             5695 ADJUST
   FOF1 FDE1        5696          POP     IY                      ;GET THE DIR ENTRY POINTER INTO IY
   FOF3 FDE5        5697          PUSH    IY
   FOF5 FD7E0C      5698          LD      A,[IY+DIR_ATTR]         ;GRAB HIS ATTRIBUTE BYTE
   FOF8 A6          5699          AND     [HL]                    ;COMPARE IT TO THE USER'S REQUEST
   FOF9 2005        5700          JR      NZ,PROTECT_FAULT
   FOFB             5701 U_OK
   FOFB E1          5702          POP     HL
   FOFC FDE1        5703          POP     IY
   FOFE AF          5704          XOR     A                       ;INDICATE ALL IS WELL
   FOFF C9          5705          RET
   F100             5706 PROTECT_FAULT
   F100 3E14        5707          LD      A,PRT_ERR
   F102 1802        5708          JR      PROTECT
   F104             5709 RANGE_NONO
   F104 3E11        5710          LD      A,RANGE_ERR
   F106             5711 PROTECT
   F106 E1          5712          POP     HL
   F107 FDE1        5713          POP     IY
   F109 B7          5714          OR      A                       ;IND[illegible]'S NEG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F10A C9          5715         RET
           <F10A>    5716 ATTRIB_TBL  EQU    $-1
    F10B 20          5717         DEFB    ATTR_READ_PROT
    F10C 40          5718         DEFB    ATTR_WRITE_PROT
    F10D 80          5719         DEFB    ATTR_PERMANENT
    F10E 02          5720         DEFB    ATTR_EXECUTE
                     5721
                     5722
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5724
                     5725 ;*****************************************************************************
                     5726 ;*******************  EOS_RENAME_FILE  ***********************************
                     5727 ;
                     5728 ;   ON ENTRY:  A = DEVICE NUMBER (8=TAPE1)
                     5729 ;             DE = POINTER TO OLD FILENAME
                     5730 ;             HL = POINTER TO NEW FILENAME
                     5731 ;
                     5732 ;   ON EXIT:   Z = 1, FILE RENAMED, A = 0
                     5733 ;             Z = 0, FILE NOT RENAMED, A = ERROR CODE
                     5734 ;
                     5735
                     5736         ;GLB     __RENAME_FILE
                     5737
F10F                 5738 __RENAME_FILE:
F10F C5              5739         PUSH    BC
F110 F5              5740         PUSH    AF                 ; AF = DEVICE ID
F111 D5              5741         PUSH    DE                 ; DE = POINTER TO OLD FILENAME
F112 E5              5742         PUSH    HL                 ; HL = POINTER TO NEW FILENAME
F113 EB              5743         EX      DE,HL              ; FIRST CHECK IF A FILE EXIST W/NEW FILE NAME
F114 21FDA0          5744         LD      HL,QUERY_BUFFER
F117 CDE61B          5745         CALL      __QUERY_FILE
F11A 282C            5746         JR      Z,NOT_RENAMED      ; YUP, THE NEW FILE NAME ALREADY EXISTS
F11C E1              5747         POP     HL
F11D D1              5748         POP     DE
F11E F1              5749         POP     AF
F11F F5              5750         PUSH    AF
F120 D5              5751         PUSH    DE
F121 E5              5752         PUSH    HL
                     5753
F122 21FDA0          5754         LD      HL,QUERY_BUFFER
F125 CDE61B          5755         CALL      __QUERY_FILE       ; NOW CHECK IF OLD FILE EXISTS
F128 201E            5756         JR      NZ,NOT_RENAMED     ; NOPE
F12A 11FDA0          5757         LD      DE,QUERY_BUFFER    ; DE = DESTINATION
F12D E1              5758         POP     HL                 ; HL = PTR TO NEW FILENAME
F12E E5              5759         PUSH    HL
F12F 01000C          5760         LD      BC,12              ; NUMBER OF BYTES TO REPLACE
F132 EDB0            5761         LDIR
F134 E1              5762         POP     HL
F135 D1              5763         POP     DE                 ; DE = POINTER TO OLD FILENAME
F136 F1              5764         POP     AF                 ; A = DEVICE ID
F137 F5              5765         PUSH    AF
F138 D5              5766         PUSH    DE                 ; RESTORE STACK
F139 E5              5767         PUSH    HL
F13A 21FDA0          5768         LD      HL,QUERY_BUFFER
F13D CDE651          5769         CALL      __SET_FILE       ; CHANGE FILENAME
F140 2006            5770         JR      NZ,NOT_RENAMED
F142 AF              5771         XOR     A                  ; GOOD RETURN
F143 E1              5772         POP     HL
F144 D1              5773         POP     DE
F145 C1              5774         POP     BC                 ; REALLY AF BUT NEED TO PRESERVE FLAGS FOR RET
F146 C1              5775         POP     BC
F147 C9              5776         RET
F148                 5777 NOT_RENAMED:
                     5778       ; LD      A,RENAME_ERR       ; ERROR RETURN
F148 B7              5779         OR      A
F149 E               5780         POP     HL
```

```
LOCATION OBJECT CODE LINE      SOURCE LINE

    F14A D1          5781      POP     DE
    F14B C1          5782      POP     BC          ; REALLY AF BUT NEED TO PRESERVE ERR MSG.
    F14C C1          5783      POP     BC
    F14D C9          5784      RET
                     5785
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5787
                     5788 ;
                     5789 ;*******************************************************************************
                     5790 ;******************  EOS_DELETE_FILE  ******************************************
                     5791 ;
                     5792 ;   ON ENTRY:  A = DEVICE NUMBER (8=TAPE1)
                     5793 ;             HL = POINTER TO FILENAME
                     5794 ;
                     5795 ;   ON EXIT:   Z = 1, FILE DELETED, A = 0
                     5796 ;              Z = 0, FILE NOT DELETED, A = ERROR CODE
                     5797 ;
                     5798
                     5799         ;GLB     __DELETE_FILE
                     5800
  F14E               5801 __DELETE_FILE:
  F14E D5            5802         PUSH    DE
  F14F E5            5803         PUSH    HL                  ; HL = POINTER TO FILENAME
  F150 F5            5804         PUSH    AF                  ; A  = DEVICE ID
  F151 EB            5805         EX      DE,HL               ; DE = POINTER TO FILENAME
  F152 21FDA0        5806         LD      HL,QUERY_BUFFER     ; HL = POINTER TO BUFFER
  F155 CDE61B        5807         CALL      __QUERY_FILE
  F158 201D          5808         JR      NZ,OTHER_ERR
  F15A 3AFDAC        5809         LD      A,[QUERY_BUFFER+FCB_ATTR] ; GET ATTRIBUTE BYTE
  F15D CB7F          5810         BIT     7,A                 ;see if permanent protected
  F15F 2013          5811         JR      NZ,NOT_DELETED
  F161 F604          5812         OR      ATTR_DELETED                ; SET "DELETED" BIT
  F163 32FDAC        5813         LD      [QUERY_BUFFER+FCB_ATTR],A
  F166 F1            5814         POP     AF                  ; DEVICE ID
  F167 D1            5815         POP     DE                  ; GET FILENAME POINTER
  F168 D5            5816         PUSH    DE
  F169 F5            5817         PUSH    AF
  F16A CDE651        5818         CALL      __SET_FILE
  F16D 2008          5819         JR      NZ,OTHER_ERR
  F16F AF            5820         XOR     A                   ; GOOD RETURN
  F170 E1            5821         POP     HL                  ; REALLY AF BUT PRESERVE NEW AF
  F171 E1            5822         POP     HL
  F172 D1            5823         POP     DE
  F173 C9            5824         RET
  F174               5825 NOT_DELETED:
  F174 3E10          5826         LD      A,DELETE_ERR        ; ERROR RETURN
  F176 B7            5827         OR      A
  F177               5828 OTHER_ERR:
  F177 E1            5829         POP     HL                  ; REALLY AF BUT PRESERVE ERR CONDITION
  F178 E1            5830         POP     HL
  F179 D1            5831         POP     DE
  F17A C9            5832         RET
                     5833
                     5834
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    5836 ********************************************************************************
                    5837 ********************************************************************************
                    5838 *
                    5839 *
                    5840 *       READ_BLOCK:
                    5841 *               1)  READS 1 BLOCK OFF OF SPECIFIED DEVICE .
                    5842 *               2)  REQUESTS FOR THE STATUS
                    5843 *               3)  AND SENDS ANOTHER READ TO THE DEVICE
                    5844 *
                    5845 *       INPUT:  SAME AS FOR __RD_1_BLOCK
                    5846 *               A       DEVICE ID
                    5847 *                       LOW NIBBLE - DEVICE ADDRESS
                    5848 *                       HI NIBBLE  - SECONDARY DEVICE ID
                    5849 *
                    5850 *               HL      DESTINATION IN RAM
                    5851 *
                    5852 *               REGISTER PAIR  BC DE
                    5853 *                       SECTOR NUMBER ON DEVICE
                    5854 *
                    5855 *
                    5856 *       OUTPUT: CONDITION FLAGS
                    5857 *                       Z:  NO ERRORS
                    5858 *                      NZ:  ERROR OCCURED
                    5859 *                           A - ERROR CODE
                    5860 *
                    5861 *       ALL REGISTERS ARE PRESERVED EXCEPT FOR AF
                    5862 *
                    5863 ********************************************************************************
                    5864 ********************************************************************************
                    5865
F17B                5866 __READ_BLOCK:
F17B                5867 READ_BLOCK:
F17B C5             5868         PUSH    BC
F17C D5             5869         PUSH    DE
F17D E5             5870         PUSH    HL
F17E DDE5           5871         PUSH    IX
F180 FDE5           5872         PUSH    IY
F182 F5             5873         PUSH    AF
                    5874
F183 3E02           5875         LD      A,ERROR_RETRY           ; RETRY COUNT FOR ERRORS
F185 32FDD6         5876         LD      [RETRY_COUNT],A
F188                5877 READ_LOOP:
F188 F1             5878         POP     AF                      ; GET DEVICE ID BACK IN A
F189 F5             5879         PUSH    AF
                    5880
F18A CDFA9E         5881         CALL    __RD_1_BLOCK
                    5882
F18D 280E           5883         JR      Z,NO_READ_ERRORS        ; READ WENT OK
                    5884
F18F FE9B           5885         CP      TIMEOUT                 ; WAS IT A TIMEOUT?
                    5886
F191 280A           5887         JR      Z,NO_READ_ERRORS          ; YUP,
                    5888
F193 E5             5889         PUSH    HL
F194 21FDD6         5890         LD      HL,RETRY_COUNT
F197 35             5891         DEC     [HL]                    ; DEC THE NO OF RETRY'S LEFT
F198 E1             5892         POP     HL
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F199 283F        5893         JR       Z,READ_ERROR            ; NO RETRY'S LEFT
    F19B 18EB        5894         JR       READ_LOOP               ; TRY AGAIN
                     5895
    F19D             5896 NO_READ_ERRORS:
    F19D 0602        5897         LD       B,ERROR_RETRY           ; NUMBER OF TIMES TO RETRY
                     5898
    F19F             5899 RETRY_FOR_ERRORS:
    F19F F1          5900         POP      AF                      ; RESTORE DEVCIE ID
    F1A0 F5          5901         PUSH     AF
                     5902
    F1A1 CDF473      5903         CALL     __REQUEST_STATUS        ; GET STATUS
                     5904
    F1A4 2808        5905         JR       Z,READ_COMMAND_COMPLETED;       READ ALL DONE
                     5906
    F1A6 FE9B        5907         CP       TIMEOUT                 ; WAS IT A TIMEOUT?
    F1A8 28F5        5908         JR       Z,RETRY_FOR_ERROR       ; NOT A REAL ERROR, DON'T NEED TO DEC COUNTER
                     5909
    F1AA 10F3        5910         DJNZ     RETRY_FOR_ERROR         ; IT'S A REAL ERROR, TRY AGAIN
                     5911
    F1AC 182C        5912         JR       READ_ERROR              ; CAN'T GET STATUS , GET OUT
                     5913
    F1AE             5914 READ_COMMAND_COMPLETE:
    F1AE F1          5915         POP      AF                      ; RESTORE DEVICE ID
    F1AF F5          5916         PUSH     AF
                     5917
    F1B0 CDF225      5918         CALL     GET_STATUS_FLAG         ; GET THE STATUS FOR THE PROPER
                     5919                                          ;  DEVICE IN THE A REGISTER
                     5920
    F1B3 FE00        5921         CP       0                       ; ANY ERRORS WITH DRIVE?
                     5922
    F1B5 2023        5923         JR       NZ,READ_ERROR           ; YUP
                     5924
    F1B7 3E02        5925         LD       A,ERROR_RETRY
    F1B9 32FDD6      5926         LD       [RETRY_COUNT],A
                     5927
    F1BC F1          5928         POP      AF
    F1BD FDE1        5929         POP      IY
    F1BF DDE1        5930         POP      IX
    F1C1 E1          5931         POP      HL
    F1C2 D1          5932         POP      DE
    F1C3 C1          5933         POP      BC
    F1C4 C5          5934         PUSH     BC
    F1C5             5935 _2ND_READ:
    F1C5 F5          5936         PUSH     AF
                     5937
    F1C6 CDFA9E      5938         CALL     __RD_1_BLOCK            ; SEND ANOTHER READ COMMAND
    F1C9 280C        5939         JR       Z,NO_ERRORS
    F1CB E5          5940         PUSH      HL
    F1CC 21FDD6      5941         LD        HL,RETRY_COUNT
    F1CF 35          5942         DEC       [HL]                   ; THE NO OF RETRY'S
    F1D0 E1          5943         POP       HL
    F1D1 2803        5944         JR        Z _2ND_READ_ERROR      ; NO RETRY'S LEFT
    F1D3 F1          5945         POP       AF
                     5946
    F1D4 [illegible] 5947         JR       _2ND_READ               ; GO TRY AGAIN
                     5948
    F1D6             5949 _2ND_READ_ERROR:
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F1D6 B7          5950         OR       A                        ; NZ CONDITION
    F1D7             5951 NO_ERRORS:
    F1D7 C1          5952         POP      BC
    F1D8 C1          5953         POP      BC
                     5954
    F1D9 C9          5955         RET
                     5956
    F1DA             5957 READ_ERROR:
    F1DA FDE1        5958         POP      IY                ; POP AF INTO IY TO PRESERVE FLAGS
    F1DC FDE1        5959         POP      IY
    F1DE DDE1        5960         POP      IX
    F1E0 E1          5961         POP      HL
    F1E1 D1          5962         POP      DE
    F1E2 C1          5963         POP      BC
                     5964
    F1E3 3E16        5965         LD       A,DEVICE_DEPD_ERR        ; INDICATE ERROR, LET APPL. HANDLE IT
                     5966
    F1E5 C9          5967         RET
                     5968
                     5969
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5971
                     5972 ********************************************************************************
                     5973 ********************************************************************************
                     5974 *
                     5975 *
                     5976 *       WRITE_BLOCK:
                     5977 *               1)  WRITES A BLOCK OUT TO TAPE
                     5978 *               2)  AND CHECKS STATUS AFTER WRITE IS COMPLETED
                     5979 *
                     5980 *       INPUT:  SAME AS FOR  __WR_1_BLOCK
                     5981 *
                     5982 *               A       DEVICE ID
                     5983 *                       LOW NIBBLE - DEVICE ADDRESS
                     5984 *                       HI NIBBLE  - SECONDARY DEVICE ID
                     5985 *
                     5986 *               HL      SOURCE ADDRESS OF BUFFER
                     5987 *
                     5988 *               REGISTER PAIR  BC DE
                     5989 *                       SECTOR NUMBER ON DEVICE
                     5990 *
                     5991 *
                     5992 *       OUTPUT: CONDITION FLAGS
                     5993 *                       Z:  NO ERRORS
                     5994 *                      NZ:  ERRORS OCCURED
                     5995 *                               A - ERROR CODE
                     5996 *
                     5997 *       ALL REGISTERS PRESERVED EXCEPT FOR AF
                     5998 *
                     5999 ********************************************************************************
                     6000 ********************************************************************************
                     6001
                     6002
                     6003
F1E6                 6004 __WRITE_BLOCK:
F1E6                 6005 WRITE_BLOCK:
F1E6 C5              6006         PUSH    BC
F1E7 FDE5            6007         PUSH    IY
F1E9 F5              6008         PUSH    AF
                     6009
F1EA 3E02            6010         LD      A,ERROR_RETRY
F1EC 32FDD6          6011         LD      [RETRY_COUNT],A
F1EF                 6012 TRY_WRITE_AGAIN:
F1EF F1              6013         POP     AF
F1F0 F5              6014         PUSH    AF
                     6015
F1F1 CDFAB2          6016         CALL    __WR_1_BLOCK
                     6017
F1F4 280E            6018         JR      Z,NO_WRITE_ERRORS       ; NO ERRORS
                     6019
F1F6 FE9B            6020         CP      TIMEOUT
                     6021
F1F8 280A            6022         JR      Z,NO_WRITE_ERRORS         ; ERROR OCCURED IN  TRANSMISSION
                     6023
F1FA E5              6024         PUSH    HL
F1FB 21FDD6          6025         LD      HL,RETRY_COUNT
F1FE 3'              6026         DEC     [HL]
F1FF E,              6027         POP     HL
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F200 281C        6028         JR       Z,WRITE_ERROR
                     6029
    F202 18EB        6030         JR       TRY_WRITE_AGAIN
                     6031
    F204             6032 NO_WRITE_ERRORS:
                     6033
    F204 0602        6034         LD       B,ERROR_RETRY          ; MAX NO OF RETRIES
                     6035
    F206             6036 RETRY_WRITE:
    F206 F1          6037         POP      AF                     ; RESTORE DEVICE ID
    F207 F5          6038         PUSH     AF
                     6039
    F208 CDF473      6040         CALL     __REQUEST_STATUS
                     6041
    F20B 2808        6042         JR       Z,WRITE_COMMAND_COMPLETE       ; STATUS BACK
                     6043
    F20D FE9B        6044         CP       TIMEOUT                ; CHECK FOR TIMEOUT
                     6045
    F20F 28F5        6046         JR       Z,RETRY_WRITE          ; TIMEOUT, NOT A 'REAL' ERROR
                     6047
    F211 10F3        6048         DJNZ     RETRY_WRITE            ; REAL ERROR, TRY AGAIN
                     6049
    F213 1809        6050         JR       WRITE_ERROR            ; DONE RETRYING, IT'S AN ERROR
                     6051
    F215             6052 WRITE_COMMAND_COMPLETE:
    F215 F1          6053         POP      AF                     ; RESTORE DEVICE ID
    F216 F5          6054         PUSH     AF
                     6055
    F217 CDF225      6056         CALL     GET_STATUS_FLAG        ; GET STATUS FOR SPECIFIED DEVICE
                     6057
    F21A FE00        6058         CP       0                      ; WRITE OK?
    F21C 2802        6059         JR       Z,WRITE_OK             ; YUP
                     6060
    F21E             6061 WRITE_ERROR:
    F21E 3E16        6062         LD       A,DEVICE_DEPD_ERR
                     6063
    F220             6064 WRITE_OK:
    F220 C1          6065         POP      BC                 ; REALLY AF , BUT PRESERVE CURRENT FLAGS
    F221 FDE1        6066         POP      IY
    F223 C1          6067         POP      BC
                     6068
                     6069
    F224 C9          6070         RET
                     6071
                     6072
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  F225                   GET_STATUS_FLAG:

  F225 CB3F                      SRL     A              ; GET SECONDARY DEVICE ID (HI NIBBLE)
  F227 CB3F                      SRL     A
  F229 CB3F                      SRL     A
  F22B CB3F                      SRL     A

  F22D FE00                      CP      0              ; WAS IT UNIT 0?

  F22F FD7E14                    LD      A,[IY+D_STATUS_FLAGS]   ; BY THE WAY, GET THE STATUS

  F232 2004                      JR      NZ,UNIT1       ; NOPE, UNIT 1

  F234 E60F                      AND     0FH            ; MASK OUT STATUS OF UNIT 1

  F236 1808                      JR      CHECK_STATUS

  F238                   UNIT1:
  F238 CB3F                      SRL     A
  F23A CB3F                      SRL     A
  F23C CB3F                      SRL     A
  F23E CB3F                      SRL     A

  F240                   CHECK_STATUS:                  ; RETURN WITH THE STATUS NIBBLE IN A

  F240 C9                        RET

```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    6103 ************************************************************************************
                    6104 ************************************************************************************
                    6105 *
                    6106 *
                    6107 *       TRIM_FILE RELEASES THE UNUSED BLOCKS IN THE SPECIFIED FILE
                    6108 *
                    6109 *  NOTE:  IF FILE BEING TRIMMED IS THE LAST FILE, THE UNUSED BLOCKS ARE
                    6110 *         ALLOCATED AS PART OF THE REMAINING AVAILABLE BLOCKS (RETURNED
                    6111 *         TO THE HOLE).
                    6112 *
                    6113 *       INPUT:  A       DEVICE ID
                    6114 *               DE      POINTER TO FILE NAME STRING
                    6115 *
                    6116 *       OUTPUT:
                    6117 *                CONDITION FLAGS
                    6118 *                Z:      NO ERRORS
                    6119 *               NZ:      ERRORS
                    6120 *                        A - ERROR CODE
                    6121 *
                    6122 ************************************************************************************
                    6123 ************************************************************************************
                    6124
                    6125        ;GLB     __TRIM_FILE
                    6126
                    6127        ;EXT     FCB_BUFFER
                    6128        ;EXT     NEW_HOLE_SIZE
                    6129        ;EXT     HOLE_FILE_NAME
                    6130
F241                6131 __TRIM_FILE:
                    6132
F241 E5             6133        PUSH     HL
F242 DDE5           6134        PUSH     IX
F244 FDE5           6135        PUSH     IY
F246 D5             6136        PUSH     DE
F247 F5             6137        PUSH     AF
                    6138
F248 21FDBA         6139        LD       HL,FCB_BUFFER
                    6140
F24B CDE61B         6141        CALL     __QUERY_FILE              ; FIND FILE IN DIRECTORY
                    6142
F24E C2F31A         6143        JP       NZ,ERROR                  ; HAD TROUBLE FINDING IT
                    6144
F251 DD21FDBA       6145        LD       IX,FCB_BUFFER             ; POINT TO THE BUFFER
                    6146
F255 DD5E13         6147        LD       E,[IX+FCB_USED_LENGTH]
F258 DD5614         6148        LD       D,[IX+FCB_USED_LENGTH+1]  ; GET THE NO OF USED BLOCKS
                    6149
F25B DD6E11         6150        LD       L,[IX+FCB_MAX_LENGTH]
F25E DD6612         6151        LD       H,[IX+FCB_MAX_LENGTH+1]   ; GET THE NO ALLOCATED
                    6152
F261 B7             6153        OR       A
F262 ED52           6154        SBC      HL,DE                     ; FIND OUT HOW MANY EMPTY
                    6155
F264 7C             6156        LD       A,H
F265 B5             6157        OR       L
                    6158
F266 CAF311         6159        JP       Z,NO_TRIM                 ; ALL USED
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6160
    F269 22FE1A      6161         LD       [NEW_HOLE_SIZE],HL      ; SAVE THE NUMBER OF EMPTY BLOCKS
    F26C DD7311      6162         LD       [IX+FCB_MAX_LENGTH],E   ; ALLOCATED = USED,   NOW
    F26F DD7212      6163         LD       [IX+FCB_MAX_LENGTH+1],D
                     6164
    F272 3AFDD4      6165         LD       A,[FILE_COUNT]          ; FILE ENTRY POSITION IN DIRECTORY
    F275 0627        6166         LD       B,ENT_PER_BLOCK
    F277             6167 TRY_AGAIN:
    F277 90          6168         SUB      B                       ; IS IT THE LAST ENTRY IN THIS BLOCK
    F278 2824        6169         JR       Z,LAST_ENTRY            ; YUP
    F27A 30FB        6170         JR       NC,TRY_AGAIN            ; NOPE, MUST BE IN NEXT BLOCK
                     6171
    F27C             6172 NOT_LAST_ENTRY:
    F27C DD2AFDFD    6173         LD       IX,[FCB_HEAD_ADDR]      ; GET START OF FCB'S
    F280 DD5E21      6174         LD       E,[IX+FCB_POINTER]      ; GET ADDRESS OF THE MODIFIED FCB
    F283 DD5622      6175         LD       D,[IX+FCB_POINTER+1]
    F286 D5          6176         PUSH     DE
                     6177
    F287 01001A      6178         LD       BC,DIR_ENT_LENGTH  ; OFFSET TO NEXT DIR ENTRY'S ATTR BYTE
    F28A C5          6179         PUSH     BC
                     6180
    F28B 21FDBA      6181         LD       HL,FCB_BUFFER
    F28E EDB0        6182         LDIR
                     6183
    F290 D1          6184         POP      DE
    F291 E1          6185         POP      HL
                     6186
    F292 19          6187         ADD      HL,DE                   ; POINT TO ATTR BYTE
                     6188
    F293 E5          6189         PUSH     HL
    F294 FDE1        6190         POP      IY
                     6191
    F296 FDCB0C46    6192         BIT      ATTR_HOLE_BIT,[IY+DIR_ATTR]      ; IS IT A HOLE?
    F29A 285E        6193         JR       Z,NOT_HOLE              ; NOPE
                     6194
    F29C 183C        6195         JR       ITS_HOLE
                     6196
                     6197
    F29E             6198 LAST_ENTRY:
    F29E 3AFDD4      6199         LD       A,[FILE_COUNT]
    F2A1 32FDD5      6200         LD       [MOD_FILE_COUNT],A      ; SAVE IT
                     6201
    F2A4 F1          6202         POP      AF
    F2A5 D1          6203         POP      DE
                     6204
    F2A6 D5          6205         PUSH     DE
    F2A7 F5          6206         PUSH     AF
                     6207
    F2A8 21FDBA      6208         LD       HL,FCB_BUFFER
                     6209
    F2AB CDE651      6210         CALL     __SET_FILE              ; PUT CHANGED DIRECTORY AWAY
                     6211
    F2AE 206A        6212         JR       NZ,ERROR
                     6213
    F2B0 F1          6214         POP      AF
    F2B1 D1          6215         POP      DE
                     6216
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F2B2 11F428      6217         LD       DE,HOLE_FILE_NAME        ; NAME OF HOLE
                     6218
    F2B5 D5          6219         PUSH     DE
    F2B6 F5          6220         PUSH     AF                       ; SAVE DEVICE ID
                     6221
    F2B7 21FDBA      6222         LD       HL,FCB_BUFFER
                     6223
    F2BA CDE61B      6224         CALL     __QUERY_FILE
    F2BD DD2AFDFD    6225         LD       IX,[FCB_HEAD_ADDR]
    F2C1 DD6E21      6226         LD       L,[IX+FCB_POINTER]       ; GET ADDRESS OF HOLE IN DIR BLOCK
    F2C4 DD6622      6227         LD       H,[IX+FCB_POINTER+1]
                     6228
    F2C7 E5          6229         PUSH     HL
    F2C8 FDE1        6230         POP      IY
                     6231
    F2CA FDCB0C46    6232         BIT      ATTR_HOLE_BIT,[IY+DIR_ATTR]   ; WAS IT A HOLE
    F2CE 284A        6233         JR       Z,ERROR
                     6234
    F2D0 21FDD4      6235         LD       HL,FILE_COUNT
    F2D3 3AFDD5      6236         LD       A,[MOD_FILE_COUNT]
                     6237
    F2D6 3C          6238         INC      A
                     6239
    F2D7 BE          6240         CP       [HL]
    F2D8 2037        6241         JR       NZ,NO_TRIM
                     6242
                     6243
    F2DA             6244 ITS_HOLE:
    F2DA FD6E11      6245         LD       L,[IY+DIR_MAX_LENGTH]    ; GET CURRENT HOLE SIZE
    F2DD FD6612      6246         LD       H,[IY+DIR_MAX_LENGTH+1]
                     6247
    F2E0 ED5BFE1A    6248         LD       DE,[NEW_HOLE_SIZE]       ; BLOCKS TO BE RETURNED TO HOLE
    F2E4 19          6249         ADD      HL,DE                    ; NEW HOLE SIZE
                     6250
    F2E5 FD7511      6251         LD       [IY+DIR_MAX_LENGTH],L
    F2E8 FD7412      6252         LD       [IY+DIR_MAX_LENGTH+1],H
                     6253
    F2EB FD6E0D      6254         LD       L,[IY+DIR_START_BLOCK]
    F2EE FD660E      6255         LD       H,[IY+DIR_START_BLOCK+1]
                     6256
    F2F1 B7          6257         OR       A
    F2F2 ED52        6258         SBC      HL,DE
                     6259
    F2F4 FD750D      6260         LD       [IY+DIR_START_BLOCK],L
    F2F7 FD740E      6261         LD       [IY+DIR_START_BLOCK+1],H
                     6262
    F2FA             6263 NOT_HOLE:
                     6264
    F2FA DD7E17      6265         LD       A,[IX+FCB_DEVICE]        ; DEVICE NUMBER
    F2FD 2AFDFF      6266         LD       HL,[FCB_DATA_ADDR]       ; GET ADDR OF MY BUFFER
    F300 DD5E19      6267         LD       E,[IX+FCB_BLOCK]         ; GET BLOCK ADDRESS IN BCDE
    F303 DD561A      6268         LD       D,[IX+FCB_BLOCK+1]
    F306 DD4E1B      6269         LD       C,[IX+FCB_BLOCK+2]
    F309 DD461C      6270         LD       B,[IX+FCB_BLOCK+3]
                     6271
    F30C CDF1E6      6272         CALL     WRITE_BLOCK              ; RE-WRITE THE DIRECTORY BLOCK
    F30F 2009        6273         JR       NZ,ERROR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6274
    F311             6275 NO_TRIM:
    F311 D1          6276         POP     DE              ; REALLY WAS AF
    F312 D1          6277         POP     DE
    F313 FDE1        6278         POP     IY
    F315 DDE1        6279         POP     IX
    F317 E1          6280         POP     HL
                     6281
    F318 AF          6282         XOR     A
                     6283
    F319 C9          6284         RET
                     6285
    F31A             6286 ERROR:
    F31A B7          6287         OR      A
    F31B D1          6288         POP     DE      ; REALLY AF
    F31C D1          6289         POP     DE
    F31D FDE1        6290         POP     IY
    F31F DDE1        6291         POP     IX
    F321 E1          6292         POP     HL
                     6293
    F322 C9          6294         RET
                     6295
                     6296
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6298 ************************************************************************
                     6299 *
                     6300 *       __INIT_TP_DIR INITIALIZES THE TAPE DIRECTORY
                     6301 *
                     6302 *       INPUT:  A       DEVICE ID
                     6303 *                       LOW NIBBLE - DEVICE ADDRESS
                     6304 *                       HI NIBBLE  _ SECONDARY DEVICE ID
                     6305 *
                     6306 *               C       NUMBER OF SECTORS IN DIRECTORY
                     6307 *               DE      NUMBER OF SECTORS ON DEVICE
                     6308 *
                     6309 *               HL      POINTER TO ASCII ETX TERMINATED STRING
                     6310 *                       ( VOLUME ID) (SHOULD BE BLANK-PADDED IF
                     6311 *                       LESS THAN 12 CHARACTERS)
                     6312 *       OUTPUT:
                     6313 *               CONDITION FLAGS
                     6314 *               Z       NO ERROR
                     6315 *              NZ       ERROR
                     6316 *                       A = ERROR CODE
                     6317 *
                     6318 ************************************************************************
                     6319
                     6320        ;GLB    __INIT_TAPE_DIR
                     6321        ;GLB    HOLE_FILE_NAME
                     6322        ;GLB    DIRECTORY_CHECK
                     6323
                     6324        ;EXT    SECTOR_NO
                     6325        ;EXT    DEVICE_ID
                     6326        ;EXT    FCB_DATA_ADDR
                     6327        ;EXT    SECTORS_TO_INIT
                     6328        ;EXT    WRITE_BLOCK
                     6329
F323                 6330 __INIT_TAPE_DIR:
                     6331
F323 FDE5            6332        PUSH    IY
F325 C5              6333        PUSH    BC
F326 D5              6334        PUSH    DE
F327 E5              6335        PUSH    HL
                     6336
F328 32FD72          6337        LD      [DEVICE_ID],A           ; SAVE THE DEVICE ID
                     6338
F32B 79              6339        LD      A,C
F32C 32FD86          6340        LD      [SECTORS_TO_INIT],A     ; NO OF SECTORS TO INITIALIZE
                     6341
F32F E5              6342        PUSH    HL
                     6343
F330 CDF3AF          6344        CALL    ZERO_OUT_BUFFER         ; ZERO OUT THE 1K BUFFER
                     6345
F333 21F3DA          6346        LD      HL,INITIAL_DIRECTORY     ; POINTER TO INITIAL SETUP
F336 ED5BFDFF        6347        LD      DE,[FCB_DATA_ADDR]      ; ADDRESS OF 1K BUFFER
F33A 010068          6348        LD      BC,DIR_ENT_LENGTH*3+VOL_DES_LENGTH  ; # BYTES TO MOVE INTO BUFFER
F33D EDB0            6349        LDIR
                     6350
F33F E1              6351        POP     HL                      ; GET ADDR OF ETX TERMINATED ASCII STRING
                     6352
F340 ED58FDFF        6353        LD      DE,[FCB_DATA_ADDR]      ; GET ADDRESS OF 1K BUFFER
F344 060C            6354        LD      B,12                    ; MAX OF 12 BYTES IN STRING
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6355
     F346            6356 LOAD_DIR:
     F346 7E         6357         LD      A,[HL]                  ; GET ASCII CHAR OF VOULME ID
     F347 FE03       6358         CP      ETX                     ; IS IT ETX?
                     6359
     F349 2808       6360         JR      Z,ITS_AN_ETX            ; YUP
                     6361
     F34B 12         6362         LD      [DE],A                  ; NOPE,PUT IN THE 1K BUFFER
                     6363
     F34C 23         6364         INC     HL
     F34D 13         6365         INC     DE
                     6366
     F34E 10F6       6367         DJNZ    LOAD_DIR
                     6368
     F350 3E03       6369         LD       A,ETX
     F352 1B         6370         DEC      DE
                     6371
     F353            6372 ITS_AN_ETX:
     F353 12         6373         LD      [DE],A                  ; ETX TERMINATED DIRECTORY NAME
     F354 FD2AFDFF   6374         LD      IY,[FCB_DATA_ADDR]      ; POINTER TO 1K BUFFER
                     6375
     F358 3AFD86     6376         LD      A,[SECTORS_TO_INIT]     ; GET THE NO OF SECTORS TO INTIALIZE
     F35B F680       6377         OR      080H                    ; SET HI BIT
     F35D FD770C     6378         LD      [IY+VOL_DIRSIZE],A      ; SAVE IN FIRST BYTE OF BUFFER
                     6379
     F360 E1         6380         POP     HL
     F361 D1         6381         POP     DE
     F362 C1         6382         POP     BC
                     6383
     F363 C5         6384         PUSH    BC
     F364 D5         6385         PUSH    DE
     F365 E5         6386         PUSH    HL
                     6387
     F366 FD7311     6388         LD      [IY+VOL_SIZE+0],E
     F369 FD7212     6389         LD      [IY+VOL_SIZE+1],D
                     6390
     F36C D5         6391         PUSH    DE                      ; GET TO FIRST DIR ENTRY
     F36D 11004E     6392         LD      DE,VOL_DES_LENGTH+DIR_ENT_LENGTH*2    ; POINT TO FIRST FILE ENTRY
     F370 FD19       6393         ADD     IY,DE
     F372 D1         6394         POP     DE
                     6395
     F373 FD71F7     6396         LD      [IY+DIR_MAX_LENGTH-DIR_ENT_LENGTH],C ; BLOCKS ALLOCATED FOR DIRECTORY
     F376 FD71F9     6397         LD      [IY+DIR_USED_LENGTH-DIR_ENT_LENGTH],C  ; MAKE USED BLOCKS AS ALRGE AS POSSIBLE
                     6398
     F379 0C         6399         INC     C                       ; FIRST "FILE" STARTS AFTER
     F37A FD710D     6400         LD      [IY+DIR_START_BLOCK],C  ;  LAST SECTOR OF DIR
                     6401
     F37D 0600       6402         LD      B,0                     ; WANT BC = # SECTORS USED
                     6403                                         ; (# DIR SECT + 1 TO COVER
                     6404                                         ;  SECT 0)
                     6405
     F37F EB         6406         EX      DE,HL                   ; HL = # SECTORS ON DEVICE
     F380 B7         6407         OR      A                       ; CLEAR CARRY
     F381 ED42       6408         SBC     HL,BC                   ; HL = # SECTORS LEFT FOR FILES ?????
                     6409
     F383 FD 11      6410         LD      [IY+DIR_MAX_LENGTH+0],L
     F386 FD 12      6411         LD      [IY+DIR_MAX_LENGTH+1],H
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6412
    F389 3E01        6413         LD       A,1                      ; FIRST SECTOR
    F38B 32FD87      6414         LD       [SECTOR_NO],A
                     6415
    F38E CDF3C5      6416         CALL     WRITE_1_BLOCK            ; WRITE BLOCK TO TAPE
                     6417
    F391 2016        6418         JR       NZ,WE_GOT_ERRORS         ; CAME BACK WITH ERROR
                     6419
    F393 3AFD86      6420         LD       A,[SECTORS_TO_INIT]      ; GET NO OF SECTORS TO INITIALIZE
                     6421
    F396 47          6422         LD       B,A
    F397 05          6423         DEC      B                        ; ONE LESS BECAUSE ONE WAS JUST INITIALIZED
                     6424
    F398 280E        6425         JR       Z,NO_SECTORS_TO_INIT     ; WAS ONLY ONE TO INIT
                     6426
    F39A CDF3AF      6427         CALL     ZERO_OUT_BUFFER          ; CLEAR 1K BUFFER
                     6428
    F39D             6429 SECT_INIT_LOOP:
    F39D 21FD87      6430         LD       HL,SECTOR_NO             ; SECTOR NO.
    F3A0 34          6431         INC      [HL]                     ; NEXT SECTOR
                     6432
    F3A1 CDF3C5      6433         CALL     WRITE_1_BLOCK            ; WRITE BLOCK OF ZEROS OUT
    F3A4 2003        6434         JR       NZ,WE_GOT_ERRORS         ; GOT A ERROR IN WRITE
                     6435
    F3A6 10F5        6436         DJNZ     SECT_INIT_LOOP
                     6437
    F3A8             6438 NO_SECTORS_TO_INIT:
                     6439
    F3A8 AF          6440         XOR      A                        ; ZERO CONDITION
                     6441
    F3A9             6442 WE_GOT_ERRORS:
                     6443
    F3A9 E1          6444         POP      HL
    F3AA D1          6445         POP      DE
    F3AB C1          6446         POP      BC
    F3AC FDE1        6447         POP      IY
                     6448
    F3AE C9          6449         RET
                     6450
                     6451
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6453
    F3AF             6454 ZERO_OUT_BUFFER:
                     6455
    F3AF C5          6456         PUSH    BC
    F3B0 D5          6457         PUSH    DE
    F3B1 E5          6458         PUSH    HL
                     6459
    F3B2 2AFDFF      6460         LD      HL,[FCB_DATA_ADDR]      ; POINTER TO 1K BUFFER
    F3B5 ED5BFDFF    6461         LD      DE,[FCB_DATA_ADDR]
    F3B9 13          6462         INC     DE
    F3BA 0103FF      6463         LD      BC,1023
                     6464
    F3BD 3600        6465         LD      [HL],0                  ; ZERO IT OUT
                     6466
    F3BF EDB0        6467         LDIR
                     6468
    F3C1 E1          6469         POP     HL
    F3C2 D1          6470         POP     DE
    F3C3 C1          6471         POP     BC
                     6472
    F3C4 C9          6473         RET
                     6474
                     6475
    F3C5             6476 WRITE_1_BLOCK:
    F3C5 C5          6477         PUSH    BC
                     6478
    F3C6 2AFDFF      6479         LD      HL,[FCB_DATA_ADDR]      ; ADDRESS OF 1K BUFFER
    F3C9 3AFD87      6480         LD      A,[SECTOR_NO]           ; GET SECTOR NUMBER
    F3CC 5F          6481         LD      E,A
    F3CD 1600        6482         LD      D,0
    F3CF 010000      6483         LD      BC,0
                     6484
    F3D2 3AFD72      6485         LD      A,[DEVICE_ID]
                     6486
    F3D5 CDF1E6      6487         CALL    WRITE_BLOCK             ; WRITE IT OUT
                     6488
    F3D8 C1          6489         POP     BC
                     6490
    F3D9 C9          6491         RET
                     6492
                     6493
    F3DA             6494 INITIAL_DIRECTORY:
    F3DA 2020202020  6495         DEFB    '
    F3DF 2020202020
    F3E4 2020
    F3E6 80          6496         DEFB    10000000B
    F3E7             6497 DIRECTORY_CHECK:
    F3E7 55AA00FF    6498         DEFB    055H,0AAH,000H,0FFH
    F3EB 0000        6499         DEFB    0,0
    F3ED 0000        6500         DEFB    0,0
    F3EF 0000        6501         DEFB    0,0
    F3F1 000000      6502         DEFB    0,0,0
                     6503
    F3F4 424F4F5403  6504         DEFB    'BOOT',ETX,'
    F3F9 2020202020
    F3FE 20
    F400 89          6505         DEFB    ATTR_PERMANENT.OR.ATTR_SYS.
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F401 00000000    6506         DEFB    0,0,0,0
    F405 0001        6507         DEFW    1
    F407 0001        6508         DEFW    1
    F409 0000        6509         DEFW    0
    F40B 000000      6510         DEFB    0,0,0
                     6511
    F40E 4449524543  6512         DEFB    'DIRECTORY',ETX,'  '
    F413 544F525903
    F418 2020
    F41A C8          6513         DEFB    ATTR_PERMANENT.OR.ATTR_WRITE_PROT.OR.ATTR_SYSTEM
    F41B 01000000    6514         DEFB    1,0,0,0
    F41F 0080        6515         DEFW    128
    F421 0001        6516         DEFW    1
    F423 0400        6517         DEFW    1024
    F425 000000      6518         DEFB    0,0,0
                     6519
    F428             6520 HOLE_FILE_NAME:
    F428 424C4F434B  6521         DEFB    'BLOCKS LEFT',ETX
    F42D 53204C4546
    F432 5403
    F434 01          6522         DEFB    ATTR_HOLE
    F435 00000000    6523         DEFB    0,0,0,0
    F439 0000        6524         DEFB    0,0
    F43B 0000        6525         DEFB    0,0
    F43D 0000        6526         DEFB    0,0
    F43F 570711      6527         DEFB    57H,07H,11H
                     6528
          <0068>     6529 INIT_INFO_SIZE EQU $-INITIAL_DIRECTORY
                     6530 ;
                     6531 ;*****************************************************************
                     6532 ;*****************************************************************
                     6533 ;*****************************************************************
                     6534 ;
                     6535
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    6537  NAME "REV 1 - RPD"
                    6538 *
                    6539 *   RETURN ERROR CODES UNTIL THESE ROUTINES ARE WRITTEN
                    6540 *
                    6541 *   10/12/83  VSB
                    6542 *   10/14/83  RPD  changed __EOS_4 to __CV_A
                    6543 *
                    6544
                    6545        ;GLB      __POSIT_FILE,__EOS_1,__EOS_2,__EOS_3,__CV_A
    F442            6546 __POSIT_FILE:
    F442            6547 __EOS_1:
    F442            6548 __EOS_2:
    F442            6549 __EOS_3:
    F442            6550 __CV_A:
    F442 3E17       6551          LD      A,PROG_NON_EXIST  ; SIGNIFIY PROGRAM DOES NOT EXIST
    F444 B7         6552          OR      A                 ; NZ CONDITION
    F445 C9         6553          RET
                    6554 ;
                    6555 ;*****************************************************************
                    6556 ;*****************************************************************
                    6557 ;*****************************************************************
                    6558 ;
                    6559
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6561 ;REV 0   (V/D 8/24/83)
                     6562
                     6563
                     6564 ********************************************************************************
                     6565 * THIS ROUTINE WILL FIND THE DCB FOR THE DEVICE DESIRED
                     6566 *
                     6567 *       INPUT:  A ==> POINTS TO THE DEVICE ID
                     6568 *       OUTPUT:
                     6569 *               CONDITION FLAG:
                     6570 *                       Z : NO ERRORS
                     6571 *                           IY ==> POINTS TO THE START OF THE DEVICE DCB
                     6572 *                      NZ : ERROR OCCURED
                     6573 *                           IY ==> IS DESTROYED
                     6574 *
                     6575 ********************************************************************************
                     6576
                     6577
                     6578        ;GLB      __GET_DCB_ADDR
                     6579         GLOBAL   __FIND_DCB
                     6580        ;EXT      CURRENT_PCB
                     6581
F446                 6582 __GET_DCB_ADDR:
F446                 6583 __FIND_DCB:
F446 C5              6584         PUSH     BC
F447 D5              6585         PUSH     DE
                     6586
F448 4F              6587         LD       C,A                    ; SAVE THE DEVICE ID
F449 FD2AFD70        6588         LD       IY,[CURRENT_PCB]       ; GET THE START ADDRESS OF PCB
F44D FD4603          6589         LD       B,[IY+P_NUM_DCBS]      ; GET THE NUMBER OF DCB'S
F450 AF              6590         XOR      A
F451 B8              6591         CP       B                      ; ARE THERE ZERO DCBS?
F452 2819            6592         JR       Z,DCB_NO_FIND          ; YES
                     6593                                         ; NO
F454 110004          6594         LD       DE,P_SIZE              ; GET THE SIZE OF THE PCB
F457 FD19            6595         ADD      IY,DE                  ; ADD TO IY TO GET TO THE START OF THE DCB
F459 110015          6596         LD       DE,D_SIZE             ; GET THE DCB SIZE
F45C 79              6597         LD       A,C                    ; RELOAD THE DEVICE ID
F45D E60F            6598         AND      0FH                    ; AND OF THE HIGH BITS TO GET ONLY
                     6599                                         ; THE DEVICE ID
                     6600
F45F                 6601 DCB_LOOP:
F45F FDBE10          6602         CP       [IY+D_DEV_ADDR]       ; IS THIS THE CORRECT DEVICE DCB?
F462 2806            6603         JR       Z,FOUND_IT             ; YES
                     6604                                         ; NO
F464 FD19            6605         ADD      IY,DE                  ; GET TO THE NEXT DCB AND CHECK IT
F466 10F7            6606         DJNZ     DCB_LOOP               ; GO CHECK THEM ALL UNTIL ONE IS FOUND
                     6607
F468 1803            6608         JR       DCB_NO_FIND           ;
                     6609
                     6610
F46A                 6611 FOUND_IT:
F46A 79              6612         LD       A,C                    ; RESTORE A TO ITS ORIGINAL STATE
F46B 1803            6613         JR       FINISHED
                     6614
F46D                 6615 DCB_NO_FIND:
F46D 3E01            6616         LD       A,DCB_NOT_FOUND
F46F B7              6617         OR       A
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6618
    F470             6619 FINISHED:
    F470 D1          6620         POP     DE
    F471 C1          6621         POP     BC
                     6622
    F472 C9          6623         RET
                     6624
                     6625
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6627
                     6628 ************************************************************************************************
                     6629 * THIS ROUTINE WILL INITIATE A STATUS REQUEST COMMAND, AND WILL RETURN
                     6630 * THE RESULT OF THE COMMAND.
                     6631 *       INPUT:  A==>    DEVICE ID
                     6632 *       OUTPUT: CONDITION BITS
                     6633 *               Z :     NO ERRORS
                     6634 *                       IY CONTAINS START ADDRESS OF DCB
                     6635 *               NZ:     ERROR OCCURED
                     6636 *                       A ==> ERROR CODE
                     6637 *                       IY IS DESTROYED (MAY BE ADDRESS OF DCB
                     6638 *                       UNDER CERTAIN ERROR CONDITIONS)
                     6639 *
                     6640 ************************************************************************************
                     6641
                     6642        ;GLB     __REQUEST_STATUS
                     6643
                     6644
F473                 6645 __REQUEST_STATUS:
F473 CDF446          6646        CALL     __FIND_DCB          ; GET THE CORRECT DEVICE DCB
                     6647
F476 200F            6648        JR       NZ,END_R_S
                     6649
F478 FD360001        6650        LD       [IY+D_COM_STAT],DCB_STATUS     ; SEND THE COMMAND TO REQUEST
                     6651                                    ; A STATUS
                     6652
F47C                 6653 REQUEST_LOOP:
F47C FDCB007E        6654        BIT      CMND_COMPLETE_BIT,[IY+D_COM_STAT]      ; HAS THE COMMAND COMPLETED?
F480 28FA            6655        JR       Z,REQUEST_LOOP             ; NO LOOP TILL IT DOES
                     6656
F482 FD7E00          6657        LD       A,[IY+D_COM_STAT]      ; CHECK THE STATUS
F485 FE80            6658        CP       CMND_FIN_STATUS          ; DID IT FINISH WITH NO ERRORS?
                     6659                                         ; IF ZERO EVERYTHING IS OK
                     6660                                         ; IF NZ THERE WAS AN ERROR
                     6661
F487                 6662 END_R_S:
F487 C9              6663        RET
                     6664
                     6665
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6667
                     6668       ;GLB     __RD_DEV_DEP_STAT
                     6669
    F488             6670 __RD_DEV_DEP_STAT:
    F488 FDE5        6671       PUSH     IY                  ; SAVE THIS REGISTER
    F48A CDF446      6672       CALL     __FIND_DCB          ; GET THE CORRECT DEVICE DCB
                     6673
    F48D 2004        6674       JR       NZ,END_R_D_D_S
                     6675
    F48F AF          6676       XOR      A                       ; NO ERROR
                     6677
    F490 FD7E14      6678       LD       A,[IY+D_STATUS_FLAGS] ; READ DEVICE-DEPENDENT STATUS FLAGS
                     6679
    F493             6680 END_R_D_D_S:
    F493 FDE1        6681       POP      IY                      ; RESTORE THE REGISTER
    F495 C9          6682       RET
                     6683
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6685 ********************************************************************************
                     6686 * THIS ROUTINE WILL CHECK AND SEE IF THE DEVICE DCB IS CURRENTLY ACTIVE.
                     6687 * IF IT IS THEN A NZ CONDITION WILL BE SENT BACK TO THE CALLING PROGRAM
                     6688 *      INPUT:  IY ===> POINTS TO START OF DEVICE DCB
                     6689 *      OUTPUT: CONDITION FLAGS
                     6690 *              Z :     DCB IS INACTIVE, IN OTHERWORDS IT IS AVAILABLE
                     6691 *                      A ===> IS PRESERVED
                     6692 *              NZ:     DCB IS ACTIVE
                     6693 *                      A ===>  ERROR CODE
                     6694 *
                     6695 ********************************************************************************
                     6696
                     6697 ; OK CHANGED TO G_OK TO PREVENT CONFLICT WITH IDENTICAL
                     6698 ; LABEL ELSEWHERE
                     6699
                     6700       ;GLB     CHK_IF_INACTIVE
                     6701
F496                 6702 CHK_IF_INACTIVE:
F496 C5              6703       PUSH     BC
                     6704
F497 4F              6705       LD       C,A                        ; PRESERVE A
F498 FD7E00          6706       LD       A,[IY+D_COM_STAT]          ; IS THE DCB AVAILABLE?
F49B FE00            6707       CP       DCB_IDLE                   ;
F49D 2806            6708       JR       Z,G_OK                     ; YES IT IS INACTIVE
                     6709                                           ; NO IT IS NOT
F49F FDCB007E        6710       BIT      CMND_COMPLETE_BIT,[IY+D_COM_STAT]      ; HAS THE COMMAND BEEN
                     6711                                           ; PROCESSED
F4A3 2804            6712       JR       Z,DEVICE_BUSY              ; NO, THE DEVICE IS STILL BUSY
                     6713                                           ; YES, THE DCB IS AVAILABLE
F4A5                 6714 G_OK:
F4A5 AF              6715       XOR      A                          ; INSURE ZERO CONDITION
F4A6 79              6716       LD       A,C                        ; RESTORE A
F4A7 1803            6717       JR       DONE                       ;
                     6718
F4A9                 6719 DEVICE_BUSY:
F4A9 3C              6720       INC      A                          ; INSURE A NON_ZERO CONDITION FOR FLAG
F4AA 3E02            6721       LD       A,DCB_BUSY                 ; SEND BACK THE ERROR CODE
                     6722
F4AC                 6723 DONE:
F4AC C1              6724       POP      BC
F4AD C9              6725       RET
                     6726
                     6727
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6729
                     6730 ********************************************************************************
                     6731 *
                     6732 *       CHK_FOR_IDLE CHECKS IF THE REQUESTED DEVICE IS IDLE
                     6733 *       SENDING BACK AN ERROR CODE
                     6734 *
                     6735 *       INPUT:  IY      POINTER TO THE DCB
                     6736 *
                     6737 *       OUTPUT: CONDITION FLAGS
                     6738 *               Z       DEVICE NOT IDLE
                     6739 *              NZ       DEVICE IDLE
                     6740 *
                     6741 ********************************************************************************
                     6742
                     6743
                     6744        :GLB     CHK_IF_IDLE
                     6745
F4AE                 6746 CHK_IF_IDLE:
                     6747
F4AE FD7E00          6748        LD       A,[IY+D_COM_STAT]       ; GET STATUS BYTE
                     6749
F4B1 B7              6750        OR       A                       ; IS IT IDLE(ZERO)?
F4B2 2004            6751        JR       NZ,NOT_IDLE             ; NOPE
                     6752
F4B4 3E03            6753        LD       A,DCB_IDLE_ERR          ; ERROR CODE FOR DEVICE IDLE
                     6754
F4B6 B7              6755        OR       A                       ; CREATE NZ CONDITION
                     6756
F4B7 C9              6757        RET
                     6758
F4B8                 6759 NOT_IDLE:
F4B8 AF              6760        XOR      A                       ; CLEAR ZERO FLAG
                     6761
F4B9 C9              6762        RET
                     6763 ;
                     6764 ;***************************************************************
                     6765 ;***************************************************************
                     6766 ;***************************************************************
                     6767 ;
                     6768
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6770 ;REV 0          (V/D  8-24-83)
                     6771
                     6772 ********************************************************************************
                     6773 * THIS ROUTINE WILL READ A CHARACTER FROM THE KEYBOARD
                     6774 *
                     6775 *       INPUT:  NOTHING
                     6776 *       OUTPUT: CONDITION FLAGS
                     6777 *               Z :     NO ERRORS
                     6778 *                       A ===>  CONTAINS THE KEY READ
                     6779 *               NZ:     ERROR OCCURED
                     6780 *                       A ===>  ERROR CODE
                     6781 *
                     6782 *
                     6783 ********************************************************************************
                     6784
                     6785        GLOBAL  __RD_KBD
                     6786       ;EXT     KEYBOARD_BUFFER
                     6787       ;EXT     __RD_CH_DEV
                     6788
F4BA                 6789 __RD_KBD:
F4BA C5              6790        PUSH    BC
F4BB D5              6791        PUSH    DE
                     6792
F4BC                 6793 RD_KBD_LOOP:
                     6794
F4BC CDF4DO          6795        CALL    __START_RD_KBD
                     6796
F4BF 2007            6797        JR      NZ,ERROR_OCCURED
                     6798
F4C1 4F              6799        LD      C,A                     ; PRESERVE THE DEVICE ID
                     6800
F4C2                 6801 WAITING:
F4C2 79              6802        LD      A,C                     ; RESTOR THE DEVICE ID
                     6803                                        ; BC IS SAVED IN ALL CALLED MODULES
F4C3 CDF4EO          6804        CALL    __END_RD_KBD
                     6805
F4C6 30FA            6806        JR      NC,WAITING              ; THE OPERATION HAS NOT COMPLETED YET
                     6807                                        ; AT THIS POINT WE EITHER HAVE THE
                     6808                                        ; KEY OR AN ERROR CODE IN A
F4C8                 6809 ERROR_OCCURED:
F4C8 D1              6810        POP     DE
F4C9 C1              6811        POP     BC
                     6812
F4CA C9              6813        RET
                     6814
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6816 ********************************************************************************
                     6817 * THIS ROUTINE WILL REQUEST A STATUS ON THE KEYBOARD
                     6818 *       INPUT:  NOTHING
                     6819 *       OUTPUT: CONDITION FLAG
                     6820 *               Z :     NO ERRORS
                     6821 *               NZ:     ERROR OCCURED
                     6822 *                       A ===>  ERROR CODE
                     6823 *
                     6824 ********************************************************************************
                     6825
                     6826         GLOBAL  __REQ_KBD_STAT
                     6827
                     6828        ;EXT     __REQUEST_STATUS
                     6829
F4CB                 6830 __REQ_KBD_STAT:
F4CB 3E01            6831         LD      A,KEYBOARD_ID          ; WANT STATUS ON THE KEYBOARD
F4CD C3F473          6832         JP      __REQUEST_STATUS       ;
                     6833
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6835 ********************************************************************************
                     6836 * THIS ROUTINE WILL START A READ DATA COMMNAD ON THE KEYBOARD
                     6837 *       INPUT:  NOTHING
                     6838 *       OUTPUT: CONDITION FLAGS
                     6839 *               Z :     NO ERROR
                     6840 *               NZ:     ERROR OCCURED
                     6841 *                       A ===>  ERROR CODE
                     6842 ********************************************************************************
                     6843
                     6844        ;GLB      __START_RD_KBD
                     6845
                     6846        ;EXT      __START_RD_CH_DEV
                     6847
F4D0                 6848 __START_RD_KBD:
F4D0 C5              6849        PUSH    BC
F4D1 D5              6850        PUSH    DE
                     6851
F4D2                 6852 STRT_RD_KBD_LOOP:
F4D2 3E01            6853        LD      A,KEYBOARD_ID                   ;
F4D4 11FD75          6854        LD      DE,KEYBOARD_BUFFER              ; DESTINATION OF WHERE TO PLACE KEY
F4D7 010001          6855        LD      BC,1                            ; GET 1 CHARACTER
F4DA CDFB86          6856        CALL    __START_RD_CH_DEV               ; GO INITIATE THE COMMAND
                     6857
F4DD D1              6858        POP     DE
F4DE C1              6859        POP     BC
                     6860
F4DF C9              6861        RET
                     6862
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6864
                     6865 ********************************************************************************
                     6866 * THIS ROUTINE WILL CHECK THE RESULT OF A COMMAND THAT WAS SENT.  IF THE
                     6867 * RESULT IS A NAK THEN THE ROUTINE WILL ISSUE A NEW READ COMMAND.
                     6868 *       INPUT:  NOTHING
                     6869 *       OUTPUT: CONDITION FLAG
                     6870 *               C :     COMMAND HAS FINISHED
                     6871 *               NC:     COMMAND HAS NOT BEEN FINISHED
                     6872 *               Z :     NO ERROR OCCURED
                     6873 *               NZ:     ERROR OCCURED
                     6874 *                       A ===>  ERROR CODE
                     6875 ********************************************************************************
                     6876
                     6877         ;GLB    __END_RD_KBD
                     6878         ;EXT    __END_RD_CH_DEV
                     6879
F4E0                 6880 __END_RD_KBD:
F4E0 3E01            6881         LD      A,KEYBOARD_ID          ; SEND THIS PARM TO __EN_RD_CH_DEV
F4E2 CDFBA5          6882         CALL    __END_RD_CH_DEV        ; CHECK THE RESULT
                     6883
F4E5 3014            6884         JR      NC,ROUTINE_DONE        ; CARRY FLAG IS CLEARED, CMND IS NOT COMPLETE
                     6885                                        ; THE COMMAND IS COMPLETE
F4E7 280C            6886         JR      Z,HAVE_THE_KEY         ; ROUTINE COMPLETED WITH NO ERRORS,
                     6887                                        ; CARRY FLAG IS SET, ZERO CONDITION
                     6888                                        ; THERE WAS AN ERROR
F4E9 FE8C            6889         CP      KBD_NAK                ; WAS THE ERROR A NAK?
F4EB 2005            6890         JR      NZ,NO_NAK              ; NO, GO SET C FLAG TO SHOW COMMAND COMPLETED
                     6891                                        ; YES, TRY THE COMMAND AGAIN
F4ED CDF4D0          6892         CALL    __START_RD_KBD         ;
F4F0 2808            6893         JR      Z,SHOW_COMPLETE        ;
                     6894                                        ; THERE WAS AN ERROR IN THE COMMAND RETRY
                     6895                                        ; SET THE CARRY FLAG TO SHOW THAT THE
                     6896                                        ; OPERATION COMPLETED.  THE FLAG
                     6897                                        ; CONDITION WILL BE NON-ZERO TO
                     6898                                        ; INDICATE AN ERROR OCCURED.  A CONTAINS
                     6899                                        ; THE ERROR CODE
F4F2                 6900 NO_NAK:
F4F2 37              6901         SCF
F4F3 1806            6902         JR      ROUTINE_DONE
                     6903
F4F5                 6904 HAVE_THE_KEY:
F4F5 3AFD75          6905         LD      A,[KEYBOARD_BUFFER]    ; PUT THE KEY IN THE KEYBOARD BUFFER
F4F8 1801            6906         JR      ROUTINE_DONE           ;
                     6907
F4FA                 6908 SHOW_COMPLETE:
F4FA B7              6909         OR      A                      ; CLEAR CARRY FLAG BECAUSE THE RETRY WAS STARTED
                     6910                                        ; WITH NO PROBLEMS, THE COMMAND IS NOT COMPLETE
F4FB                 6911 ROUTINE_DONE:
F4FB C9              6912         RET
                     6913 ;
                     6914 ;===============================================================
                     6915 ;===============================================================
                     6916 ;===============================================================
                     6917 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6920
                     6921 ;REV 1          (V/D   8-24-83)
                     6922 ***************************************************************************
                     6923 * PRINT_CHAR ALLOWS THE CALLER TO PRINT ONE CHARACTER
                     6924 *       INPUT:  A ===> CHARACTER TO BE PRINTED
                     6925 *       OUTPUT: CONDITION FLAGS
                     6926 *               Z :     NO ERROR
                     6927 *               NZ:     ERROR OCCURED
                     6928 *                       A ===> ERROR CODE
                     6929 *
                     6930 ***************************************************************************
                     6931
                     6932       ;GLOBAL  __PR_CH
                     6933       ;EXT     PRINT_BUFFER
                     6934       ;EXT     CHK_IF_INACTIVE
                     6935
F4FC                 6936 __PR_CH:
F4FC C5              6937       PUSH     BC
F4FD D5              6938       PUSH     DE
F4FE E5              6939       PUSH     HL
F4FF FDE5            6940       PUSH     IY
                     6941
F501 32FD76          6942       LD       [PRINT_BUFFER],A
F504 3E03            6943       LD       A,ETX                    ; TERMINATE STRIN W/ETX
F506 32FD77          6944       LD       [PRINT_BUFFER+1],A       ;
F509 21FD76          6945       LD       HL,PRINT_BUFFER          ; ADDRESS OF PRINT BUFFER
                     6946
F50C CDF515          6947       CALL     __PR_BUFF
                     6948
F50F FDE1            6949       POP      IY
F511 E1              6950       POP      HL
F512 D1              6951       POP      DE
F513 C1              6952       POP      BC
                     6953
F514 C9              6954       RET
                     6955
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE
                              *
                     6957 ********************************************************************************
                     6958 * PRINT_BUFFER ALLOWS THE CALLER TO PRINT AN ASCII ETX-TERMINATED STRING
                     6959 *       INPUT:  HL ==>  ADDRESS OF ETX-TERMINATED STRING
                     6960 *       OUTPUT: CONDITION FLAGS
                     6961 *               Z :     NO ERROR
                     6962 *               NZ:     ERROR OCCURED
                     6963 *                       A ===>  ERROR CODE
                     6964 *
                     6965 ********************************************************************************
                     6966
                     6967          GLOBAL __PR_BUFF
                     6968
                     6969        ;EXT     __FIND_DCB,__WR_CH_DEV
                     6970
F515                 6971 __PR_BUFF.
F515 C5              6972          PUSH    BC
F516 D5              6973          PUSH    DE
F517 E5              6974          PUSH    HL
F518 FDE5            6975          PUSH    IY
                     6976
F51A                 6977 PR_BUFF2:
F51A 3E02            6978          LD      A,PRINTER_ID            ; WANT THE PRINTERS DCB
F51C CDF446          6979          CALL    __FIND_DCB              ; GET THE DEVICE'S DCB
                     6980
F51F 2037            6981          JR      NZ,END_PRINT_ROUTINE    ; THERE WAS A PROBLEM FINDING THE DCB
                     6982
F521 CDF496          6983          CALL    CHK_IF_INACTIVE
F524 2032            6984          JR      NZ,PRTR_DCB_ACTIVE
                     6985
F526                 6986 CHECK_FOR_ETX:
F526 010000          6987          LD      BC,0                    ; CHARACTER COUNT
F529 5D              6988          LD      E,L                     ; SAVE THE STRING
F52A 54              6989          LD      D,H                     ; DE=HL= PTR TO UNPRINTED STRING
                     6990
F52B                 6991 GET_NEXT_ASCII:
F52B 3E03            6992          LD      A,ETX                   ;
F52D BE              6993          CP      [HL]                    ; IS IT AN ETX?
F52E 2818            6994          JR      Z,ITS_ETX               ; YES
                     6995
                     6996                                          ; NO
F530 23              6997          INC     HL                      ; POINT TO NEXT CHARACTER
F531 0C              6998          INC     C                       ; INC COUNT
F532 79              6999          LD      A,C                     ; HAVE WE LOOKED AT 16 BYTES?
F533 FE10            7000          CP      16                      ;
F535 20F4            7001          JR      NZ,GET_NEXT_ASCII       ; YES
                     7002 ;                                          DE = PTR TO UNPRINTED STRNG, DE=PTR TO UNPRNTED+16
F537 EB              7003          EX      DE,HL                   ; HL = PTR TO UNPRINTED STRNG, DE=PTR TO UNPRNTED+16
                     7004
F538                 7005 WRITE_COMMAND_AGAIN:
                     7006
F538 3E02            7007          LD      A,PRINTER_ID
F53A CDFB75          7008          CALL    __WR_CH_DEV             ;
                     7009
F53D 2806            7010          JR      Z,WRITE_DONE            ; WRITE COMPLETED WITH NO ERRORS
                     7011
F53F F               7012          CP      PR_NAK                  ;    IT A NAK?
F541 ?,,             7013          JR      NZ,PRINT_CMND_FAILED    ;    \ AN ERROR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7014
    F543 18F3        7015         JR       WRITE_COMMAND_AGAIN      ; SEND ANOTHER WRITE COMMAND
                     7016
    F545             7017 WRITE_DONE:
    F545 EB          7018         EX       DE,HL                    ; DE = PTR TO UNPRINTED STRNG, HL=PTR TO UNPRNTED+16
    F546 18DE        7019         JR       CHECK_FOR_ETX            ; GO CHECK THE NEXT CHARACTER
                     7020
                     7021
    F548             7022 ITS_ETX:
    F548 EB          7023         EX       DE,HL                    ;
    F549 AF          7024         XOR      A                        ; ZERO OUT A
    F54A B9          7025         CP       C                        ; WAS THE FIRST CHARACTER CHECKED
                     7026                                           ; AN ETX?
    F54B 280B        7027         JR       Z,END_PRINT_ROUTINE      ; YUP, DON'T BOTHER PRINTING IT
                     7028
    F54D             7029 SEND_ANOTHER_WRITE:
    F54D 3E02        7030         LD       A,PRINTER_ID             ; WILL WRITE TO THE PRINTER
    F54F CDFB75      7031         CALL     __WR_CH_DEV              ; WRITE CHARACTERS
                     7032
    F552 2804        7033         JR       Z,COMMAND_FINISHED_SUCCESSFULLY  ; ALL DONE
                     7034
    F554 FE86        7035         CP       PR_NAK                   ; DO WE HAVE A NAK?
    F556 28F5        7036         JR       Z,SEND_ANOTHER_WRITE     ; YUP
                     7037
                     7038                                          ; NOPE, MUST BE AN ERROR
                     7039
                     7040
    F558             7041 COMMAND_FINISHED_SUCCESSFULLY:
    F558             7042 PRINT_CMND_FAILED:
    F558             7043 PRNT_CMND_FAILED:
    F558             7044 PRTR_DCB_ACTIVE:
    F558             7045 END_PRINT_ROUTINE:
    F558 FDE1        7046         POP      IY
    F55A E1          7047         POP      HL
    F55B D1          7048         POP      DE
    F55C C1          7049         POP      BC                       ; ON ERROR, ZERO FLAG SET AND A
                     7050                                          ; CONTAINS THE ERROR CODE
    F55D C9          7051         RET
                     7052
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7054 ********************************************************************************
                     7055 *
                     7056 * __SETUP_PR_BUFF SETS UP PRINT_BUFFER AND THE HL REGISTER AND
                     7057 * CALLS __START_PR_BUFF
                     7058 *       INPUT:  A = ASCII CHARACTER
                     7059 *       OUTPUT:  CONDITION FLAGS
                     7060 *                  Z:    NO ERROR
                     7061 *                 NZ:    ERROR OCCURED
                     7062 *                        A = ERROR CODE
                     7063 *
                     7064 ********************************************************************************
                     7065
                     7066        ;GLB     __SETUP_PR_BUFF
                     7067
   F55E              7068 __SETUP_PR_BUFF:
   F55E 32FD76       7069        LD       [PRINT_BUFFER],A
   F561 3E03         7070        LD       A,ETX
   F563 32FD77       7071        LD       [PRINT_BUFFER+1],A
                     7072
   F566 21FD76       7073        LD       HL,PRINT_BUFFER
                     7074
   F569 CDF580       7075        CALL     __START_PR_BUFF
                     7076
   F56C C9           7077        RET
                     7078
                     7079
                     7080
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7082 ********************************************************************************
                     7083 *
                     7084 * __START_PR_CH SETS UP PRINT_BUFFER AND THE HL REGISTER AND
                     7085 * CALLS __START_PR_BUFF
                     7086 *       INPUT:  A = ASCII CHARACTER
                     7087 *       OUTPUT:  CONDITION FLAGS
                     7088 *                  Z:    NO ERROR
                     7089 *                 NZ:    ERROR OCCURED
                     7090 *                        A = ERROR CODE
                     7091 *
                     7092 ********************************************************************************
                     7093
                     7094        ;GLB      __START_PR_CH
                     7095
F56D                 7096 __START_PR_CH:
F56D 32FD76          7097        LD        [PRINT_BUFFER],A
F570 3E03            7098        LD        A,ETX
F572 32FD77          7099        LD        [PRINT_BUFFER+1],A
                     7100
F575 21FD76          7101        LD        HL,PRINT_BUFFER
                     7102
F578 CDF580          7103        CALL      __START_PR_BUFF
                     7104
F57B C9              7105        RET
                     7106
                     7107
                     7108
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7110 ********************************************************************************
                     7111 *
                     7112 *       __END_PR_CH   CALLS _END_PR_BUFF TO CHECK THE DCB STATUS BYTE AND
                     7113 *       RETURNS THE RESULT TO THE CALLER
                     7114 *
                     7115 *       INPUT:  NONE
                     7116 *
                     7117 *       OUTPUT: CONDITION FLAGS
                     7118 *               C:      COMMAND COMPLETED
                     7119 *              NC:      COMMAND NOT COMPLETED
                     7120 *               Z:      NO ERRORS OCCURED
                     7121 *              NZ:      ERRORS
                     7122 *                       A = ERROR CODE
                     7123 *
                     7124 *       ALL     REGISTERS ARE PRESERVED
                     7125 *
                     7126 ********************************************************************************
                     7127
                     7128        ;GLB     __END_PR_CH
F57C                 7129 __END_PR_CH:
F57C CDF5B8          7130        CALL     __END_PR_BUFF
                     7131
F57F C9              7132        RET
                     7133
                     7134
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7136 ********************************************************************************
                     7137 * __START_PR_BUFF ALLOWS THE CALLER TO PRINT AN ASCII ETX-TERMINATED STRING
                     7138 *      INPUT:  HL ==>  ADDRESS OF ETX-TERMINATED STRING
                     7139 *      OUTPUT: CONDITION FLAGS
                     7140 *              Z :     NO ERROR
                     7141 *              NZ:     ERROR OCCURED
                     7142 *                      A ===>  ERROR CODE
                     7143 *
                     7144 ********************************************************************************
                     7145
                     7146       ;GLB     __START_PR_BUFF
                     7147
                     7148       ;EXT     __START_WR_CH_DEV
                     7149
F580                 7150 __START_PR_BUFF:
F580 C5              7151       PUSH     BC
F581 D5              7152       PUSH     DE
F582 E5              7153       PUSH     HL
F583 FDE5            7154       PUSH     IY
                     7155
F585 3E02            7156       LD       A,PRINTER_ID            ; WANT THE PRINTERS DCB
F587 CDF446          7157       CALL     __FIND_DCB              ; GET THE DEVICE'S DCB
                     7158
F58A 2026            7159       JR       NZ,NO_DCB_FOUND1        ; THERE WAS A PROBLEM FINDING THE DCB
                     7160
F58C CDF496          7161       CALL     CHK_IF_INACTIVE
F58F 20C7            7162       JR       NZ,PRTR_DCB_ACTIVE1
                     7163
F591 010000          7164       LD       BC,0                    ; CHARACTER COUNT
F594 5D              7165       LD       E,L                     ; SAVE THE STRING
F595 54              7166       LD       D,H                     ; DE=HL= PTR TO UNPRINTED STRING
                     7167
F596                 7168 GET_NEXT_ASCII2:
F596 3E03            7169       LD       A,ETX                   ;
F598 BE              7170       CP       [HL]                    ; IS IT AN ETX?
F599 280F            7171       JR       Z,ITS_ETX2               ; YES
                     7172
                     7173                                        ; NO
F59B 23              7174       INC      HL                      ; POINT TO NEXT CHARACTER
F59C 0C              7175       INC      C                       ; INC COUNT
F59D 79              7176       LD       A,C                     ; HAVE WE LOOKED AT 16 BYTES?
F59E FE10            7177       CP       16                      ;
F5A0 20F4            7178       JR       NZ,GET_NEXT_ASCII2      ; YES
                     7179 ;                                        DE = PTR TO UNPRINTED STRNG, DE=PTR TO UNPRNTED+16
F5A2 EB              7180       EX       DE,HL                   , HL = PTR TO UNPRINTED STRNG, DE=PTR TO UNPRNTED+16
F5A3 3E02            7181       LD       A,PRINTER_ID
F5A5 CDFBC2          7182       CALL     __START_WR_CH_DEV            ;
                     7183
F5A8 1808            7184       JR       WRITE_CMND_SENT
                     7185
F5AA                 7186 ITS_ETX2:
F5AA EB              7187       EX       DE,HL                   ;
F5AB AF              7188       XOR      A                       ; ZERO OUT A
F5AC B9              7189       CP       C                       ; WAS THE FIRST CHARACTER CHECKED
                     7190                                        ; AN ETX?
                     7191
F5AD 3E02            7192       LD       A,PRINTER_ID            ; WILL WRITE TO THE PRINTER
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F5AF C4FBC2      7193         CALL    NZ,__START_WR_CH_DEV          ; NOPE, WRITE CHARACTER
                     7194                                             ; IS PASSED BACK TO CALLER
    F5B2             7195 NO_DCB_FOUND1:
    F5B2             7196 WRITE_CMND_SENT:
                     7197
    F5B2 FDE1        7198         POP     IY
    F5B4 E1          7199         POP     HL
    F5B5 D1          7200         POP     DE
    F5B6 C1          7201         POP     BC
                     7202
    F5B7 C9          7203         RET
                     7204
                     7205
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7207
                     7208 ********************************************************************************
                     7209 *
                     7210 *       __END_PR_BUFF CHECKS THE DCB COMM/STATUS BYTE AND RETURNS THE RESULT
                     7211 *       OF THE CHECK
                     7212 *
                     7213 *       INPUT:  NONE
                     7214 *
                     7215 *       OUTPUT: CONDITION FLAGS
                     7216 *               C:      COMMAND COMPLETED
                     7217 *              NC:      COMMAND NOT COMPLETED
                     7218 *               Z:      NO ERRORS OCCURED
                     7219 *              NZ:      ERRORS
                     7220 *                       A = ERROR CODE
                     7221 *
                     7222 *       ALL     REGISTERS ARE PRESERVED
                     7223 *
                     7224 ********************************************************************************
                     7225
                     7226        ;GLB     __END_PR_BUFF
                     7227        ;EXT     __END_WR_CH_DEV
                     7228
F5B8                 7229 __END_PR_BUFF:
                     7230
F5B8 FDE5            7231        PUSH     IY
                     7232
F5BA 3E02            7233        LD       A,PRINTER_ID
F5BC CDF446          7234        CALL     __FIND_DCB
F5BF CDFBE1          7235        CALL     __END_WR_CH_DEV          ; CHECK STATUS BYTE
                     7236
F5C2 300B            7237        JR       NC,CMND_NOT_PROCESSED    ; COMMAND NOT COMPLETED
                     7238
F5C4 2809            7239        JR       Z,FINISHED_WO_ERROR      ; COMMAND COMPLETED, NO ERROR
                     7240
F5C6 FE86            7241        CP       PR_NAK                   ; WAS IT A NAK?
                     7242
F5C8 2005            7243        JR       NZ,NOT_PR_NAK            ; NOPE, MUST BE AN ERROR
                     7244
F5CA FD360003        7245        LD       [IY+D_COM_STAT],DCB_WR   ; SEND ANOTHER WRITE COMMAND
                     7246
F5CE B7              7247        OR       A                        ; CLEAR CARRY TO INDICATE COMMAND
                     7248                                          ; HAS NOT BEEN FINISHED
                     7249
F5CF                 7250 NOT_PR_NAK:
F5CF                 7251 CMND_NOT_PROCESSED:
F5CF                 7252 FINISHED_WO_ERROR:
                     7253
F5CF FDE1            7254        POP      IY
                     7255
F5D1 C9              7256        RET
                     7257
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    7259 ************************************************************************************
                    7260 * THIS ROUTINE REQUESTS THE STATUS ON THE PRINTER
                    7261 *       INPUT:  NONE
                    7262 *       OUTPUT:
                    7263 *               CONDITION FLAGS
                    7264 *                       Z :     NO ERROR
                    7265 *                       NZ:     ERROR OCCURED
                    7266 *                               A ===>  ERROR CODE
                    7267 *
                    7268 ************************************************************************************
                    7269         GLOBAL  __REQ_PR_STAT
                    7270
                    7271        ;EXT     __REQUEST_STATUS
                    7272
F5D2                7273 __REQ_PR_STAT:
F5D2 3E02           7274         LD      A,PRINTER_ID
F5D4 C3F473         7275         JP      __REQUEST_STATUS
                    7276 ;
                    7277 ;*******************************************************************
                    7278 ;*******************************************************************
                    7279 ;*******************************************************************
                    7280 ;
                    7281
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7283
                     7284 ;REV 0           (V/D   8-24-83)
                     7285
                     7286        GLOBAL  __REQ_TAPE_STAT
                     7287
                     7288       ;EXT     __REQUEST_STATUS
                     7289
                     7290 ************************************************************************
                     7291 * THIS ROUTINE REQUESTS THE STATUS ON THE TAPE DRIVE
                     7292 *       INPUT:  NONE
                     7293 *       OUTPUT:
                     7294 *               CONDITION FLAGS
                     7295 *                       Z :     NO ERROR
                     7296 *                       NZ:     ERROR OCCURED
                     7297 *                               A ===> ERROR CODE
                     7298
    F5D7             7299 __REQ_TAPE_STAT:
    F5D7 3E08        7300        LD      A,TAPE_ID
    F5D9 C3F473      7301        JP      __REQUEST_STATUS
                     7302 ;
                     7303 ;************************************************************************
                     7304 ;************************************************************************
                     7305 ;************************************************************************
                     7306 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7308  NAME   "Rev 18  - RPD"
                     7309 ;
                     7310 ;               Rev History
                     7311 ;               Rev    Date        Name      Change
                     7312 ;               18  30sept400p    RPD       extracted RAM areas and put into EOS_COMN
                     7313 ;               17  21Sept83_1440 JKL       take out stuff that EOS does not need, put CR back to normal
                     7314 ;                                           take out flashing cursor
                     7315 ;               16  19Sept83_1730 JKL       CR now does a clear to EOL and LF
                     7316 ;               15  18Sept83_1800 JKL       CR now does a CLEAR to EOL
                     7317 ;               14  15Sept83_1600 JKL       added routine to flash cursor
                     7318 ;               13     15Sept83   JKL       XOR cursor with char that is under, take out  print screen
                     7319 ;               12     14Sept83   JKL       include print screen
                     7320 ;               11     14Sept83   JKL       return cursor position in DE
                     7321 ;               10     06Sept83   JKL       added routine to return cursor position in window
                     7322 ;                9     05Sept83   JKL       fix for SHIFT LOCK back space, to be removed later
                     7323 ;                8     04Sept83   JKL       added delete char and shift left
                     7324 ;                7     04Sept83   JKL       added insert blank, made LINEBUFFER_ 33 chars long
                     7325 ;                6     03Sept83   JKL       INIT will set cursor pos
                     7326 ;                5     03Sept83   JKL       Added return cursor pos routine
                     7327 ;                4     01Sept83   JKL       Fixes and speedups, Major rewrite
                     7328 ;                3     27Aug83    RPD       new case implementation (jump table)
                     7329 ;                2     27Aug83    RPD       added scroll
                     7330 ;                1     26Aug83    RPD       moved to EOS, contains window and XY logic
                     7331 ;                0     26Aug83    RPD       copied from old EOS, uses WRITE and READ VRAM
                     7332 ;               - - - - - - - - - - - - - - -
                     7333
                     7334 ;
                     7335 ;External Routines
                     7336 ;
                     7337                ;EXT     WRITE_VRAM      ;OS-7
                     7338                ;EXT     READ_VRAM       ;OS-7
                     7339 ;
                     7340 ;External data structures used
                     7341 ;
                     7342                ;EXT     OLDCHAR_
                     7343                ;EXT     X_MIN
                     7344                ;EXT     X_MAX
                     7345                ;EXT     Y_MIN
                     7346                ;EXT     Y_MAX
                     7347                ;EXT     LINEBUFFER_
                     7348                ;EXT     NUM_LINES
                     7349                ;EXT     NUM_COLUMNS
                     7350                ;EXT     UPPER_LEFT
                     7351                ;EXT     PTRN_NAME_TBL
                     7352                ;EXT     CURSOR
                     7353
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7355 ;
                     7356 ;Local Equates
                     7357 ;
              <000A> 7358 LF              EQU     0AH
              <000C> 7359 FF              EQU     0CH
              <0008> 7360 BS              EQU     08H
              <000D> 7361 CR              EQU     0DH
              <0016> 7362 CLR_TO_EOL      EQU     16H
              <0018> 7363 CLR_TO_EOS      EQU     18H
              <001C> 7364 GOTO_XY_CHAR    EQU     1CH
              <0080> 7365 HOME            EQU     80H
              <00A0> 7366 UP              EQU     0A0H
              <00A1> 7367 RIGHT           EQU     0A1H
              <00A2> 7368 DOWN            EQU     0A2H
              <00A3> 7369 LEFT            EQU     0A3H
              <005F> 7370 CURSOR_CHAR     EQU     "_"             ;Char to emit as cursor
              <0020> 7371 BLANKING_CHAR   EQU     " "             ;Char used when scrolling to fill screen
              <FEA6> 7372 CURSOR_X        EQU     CURSOR+1
                     7373
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7375 ;
                     7376 ;__CONS_INIT
                     7377 ;
                     7378 ;       Sets up the window sizes for the CONSOLE_OUT routine
                     7379 ;
                     7380 ;Input Parameters:
                     7381 ;  B - number of columns in the window - 1, 0..31
                     7382 ;  C - number of lines in the window - 1, 0..23
                     7383 ;  D - Absolute (0 based) X coord of upper left corner of window, 0..31
                     7384 ;  E - Absolute (0 based) Y coord of upper left corner of window, 0..23
                     7385 ; HL - Pattern Name Table (PNT) starting address
                     7386 ;
                     7387 ;NOTE- User must initialize the VDP
                     7388 ;
                     7389                ;GLB     __CONS_INIT
F5DC                 7390 __CONS_INIT:
F5DC 04              7391                INC     B               ;Adjust for old calling convention
F5DD 0C              7392                INC     C
F5DE ED43FE9F        7393                LD      [NUM_LINES],BC  ;Set num lines, cols in window
F5E2 ED53FEA1        7394                LD      [UPPER_LEFT],DE ;Set upper left corner of window
F5E6 ED53FEA5        7395                LD      [CURSOR],DE
F5EA 22FEA3          7396                LD      [PTRN_NAME_TBL],HL ;Save address of Pattern name table
                     7397
F5ED 7A              7398                LD      A,D             ;Calc min/max coords of window
F5EE 32FE7A          7399                LD      [X_MIN],A
F5F1 80              7400                ADD     A,B
F5F2 3D              7401                DEC     A
F5F3 32FE7B          7402                LD      [X_MAX],A
F5F6 7B              7403                LD      A,E
F5F7 32FE7C          7404                LD      [Y_MIN],A
F5FA 81              7405                ADD     A,C
F5FB 3D              7406                DEC     A
F5FC 32FE7D          7407                LD      [Y_MAX],A
                     7408
F5FF 3E20            7409                LD      A,BLANKING_CHAR
F601 32FE79          7410                LD      [OLDCHAR_],A
F604 3E5F            7411                LD      A,CURSOR_CHAR
F606 CDF7DA          7412                CALL    PUT_CHAR_ON_SCREEN
                     7413
F609 C9              7414                RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7416 ;
                     7417 ; __CONS_OUT    -  This entry point will check special chars such as
                     7418 ;                  Carriage Return and performs their normal functions
                     7419 ;                  all other, printable, chars are put onto the screen
                     7420 ;
                     7421 ; __CONS_DISP   -  This entry point will put all chars onto the screen
                     7422 ;                  no matter what, no checking is done for control chars
                     7423 ;
                     7424 ;
                     7425 ;Input Parameters:
                     7426 ;   A - character to emit
                     7427 ;
                     7428 ;  optionally
                     7429 ;   E - Y location to goto (must be in window)
                     7430 ;   D - X location to goto (must be in window)
                     7431 ;
                     7432 ;
                     7433 ;   Saves all registers
                     7434 ;
                     7435                ;GLB     __CONS_OUT
                     7436                ;GLB     __CONS_DISP
F60A                 7437 __CONS_OUT:
F60A F5              7438                 PUSH    AF
F60B C5              7439                 PUSH    BC
F60C E5              7440                 PUSH    HL
F60D DDE5            7441                 PUSH    IX
F60F FDE5            7442                 PUSH    IY
F611 D5              7443                 PUSH    DE              ;This may contain the X-Y coord of goto xy
                     7444
F612 21F80E          7445                 LD      HL,SPECIAL_CHARS
F615 01000C          7446                 LD      BC,NUM_SPCL_CHARS
F618 EDB1            7447                 CPIR
F61A 2013            7448                 JR      NZ,NOT_SPECIAL
                     7449
F61C 21F81A          7450                 LD      HL,SPCL_VECTOR_TBL
F61F 09              7451                 ADD     HL,BC
F620 09              7452                 ADD     HL,BC
                     7453
F621 47              7454                 LD      B,A             ;Save char in B
                     7455
F622 7E              7456                 LD      A,[HL]
F623 23              7457                 INC     HL
F624 66              7458                 LD      H,[HL]
F625 6F              7459                 LD      L,A
F626 E9              7460                 JP      [HL]            ;Vector to special char handler
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F627             7462 __CONS_DISP:
    F627 F5          7463                 PUSH    AF
    F628 C5          7464                 PUSH    BC
    F629 E5          7465                 PUSH    HL
    F62A DDE5        7466                 PUSH    IX
    F62C FDE5        7467                 PUSH    IY
    F62E D5          7468                 PUSH    DE
    F62F             7469 NOT_SPECIAL:
    F62F CDF7DA      7470                 CALL    PUT_CHAR_ON_SCREEN ;Display char where cursor is at
    F632 2AFEA5      7471                 LD      HL,[CURSOR]
    F635 24          7472                 INC     H                  ;Move cursor to right (+1)
    F636 3AFE7B      7473                 LD      A,[X_MAX]          ;  but check for going out of
    F639 BC          7474                 CP      H                  ;    the window
    F63A 3014        7475                 JR      NC,SAVE_CURSOR     ;If taken then stayed in window
    F63C 3AFE7A      7476                 LD      A,[X_MIN]          ;Went out of window so set X
    F63F 67          7477                 LD      H,A                ;   to left edge
    F640 2C          7478                 INC     L                  ; and move down one line
    F641 3AFE7D      7479                 LD      A,[Y_MAX]          ;   but check to see if still
    F644 BD          7480                 CP      L                  ;      in the window
    F645 3009        7481                 JR      NC,SAVE_CURSOR
    F647 2D          7482                 DEC     L                  ; Put back into window
    F648 E5          7483                 PUSH    HL
    F649 CDF7ED      7484                 CALL    GET_CHAR_UNDER_CURSOR
    F64C CDF794      7485                 CALL    SCROLL_UP
    F64F E1          7486                 POP     HL
                     7487
    F650 22FEA5      7488 SAVE_CURSOR:    LD      [CURSOR],HL
    F653 CDF7F0      7489 NEW_CURSOR:     CALL    GET_CHAR_UNDER
    F656             7490 PUT_CURSOR_ON:
    F656 3E5F        7491                 LD      A,CURSOR_CHAR
    F658 CDF7DA      7492                 CALL    PUT_CHAR_ON_SCREEN
                     7493
    F65B D1          7494 EXIT:           POP     DE
    F65C FDE1        7495                 POP     IY
    F65E DDE1        7496                 POP     IX
    F660 E1          7497                 POP     HL
    F661 C1          7498                 POP     BC
    F662 F1          7499                 POP     AF
    F663 C9          7500                 RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7502 ;
                     7503 ;Special Character Handlers
                     7504 ;
                     7505 ;
                     7506 ; RETURN_WITH_ERASE -  moves cursor to right edge of window on current line
                     7507 ;            after clearing to end of line
                     7508 ;
                     7509 ;RETURN_WITH_ERASE:
                     7510 ;                  CALL    ERASE_THIS_LINE
    F664 CDF7D7      7511 RETURN:          CALL    PUT_OLD_ON_SCREEN          ;Display char cursor was hiding
    F667 3AFE7A      7512                  LD      A,[X_MIN]
    F66A 32FEA6      7513                  LD      [CURSOR_X],A
    F66D 2AFEA5      7514                  LD      HL,[CURSOR]
    F670 C3F653      7515                  JP      NEW_CURSOR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7517 ;
                     7518 ; MOVE_UP  -  Moves the cursor up if not at top of window
                     7519 ;
    F673             7520 MOVE_UP:
    F673 2AFEA5      7521                 LD      HL,[CURSOR]
    F676 3AFE7C      7522                 LD      A,[Y_MIN]
    F679 BD          7523                 CP      L
    F67A 28DF        7524                 JR      Z,EXIT          ;Forget request
    F67C 2D          7525                 DEC     L               ;Move up
                     7526
    F67D CDF7D7      7527 RESTORE_OLD:    CALL    PUT_OLD_ON_SCREEN ;Display char cursor was hiding
    F680 C3F650      7528                 JP      SAVE_CURSOR     ;Display cursor in new spot
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7530 ;
                     7531 ; MOVE_DOWN  -  Moves the cursor down if will stay in window, except
                     7532 ;               if used by linefeed when the screen is scrolled
                     7533 ;
     F683            7534 MOVE_DOWN:
                     7535
     F683 2AFEA5     7536                 LD      HL,[CURSOR]
     F686 3AFE7D     7537                 LD      A,[Y_MAX]       ;See if at bottom of window
     F689 BD         7538                 CP      L
     F68A 2804       7539                 JR      Z,SCROLL_POSIBLE
     F68C 2C         7540                 INC     L               ;Move down
     F68D C3F67D     7541                 JP      RESTORE_OLD
                     7542 ;
     F690 3E0A       7543 SCROLL_POSIBLE: LD      A,LF            ;See if linefeed, and if it is
     F692 B8         7544                 CP      B               ;   then scroll up, else leave
     F693 20C6       7545                 JR      NZ,EXIT         ;     cursor and screen as is
     F695 CDF794     7546                 CALL    SCROLL_UP
     F698 2AFEA5     7547                 LD      HL,[CURSOR]
     F69B C3F653     7548                 JP      NEW_CURSOR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7550 ;
                     7551 ; MOVE_RIGHT  -  Moves the cursor to the right, and if at window's
                     7552 ;                edge will move down a line to the left edge, unless
                     7553 ;                at the lower right corner of the window in which case
                     7554 ;                nothing is down
                     7555 ;
   F69E              7556 MOVE_RIGHT:
   F69E 2AFEA5       7557                  LD      HL,[CURSOR]
   F6A1 3AFE7B       7558                  LD      A,[X_MAX]
   F6A4 BC           7559                  CP      H
   F6A5 2804         7560                  JR      Z,TRY_MOVE_DOWN
   F6A7 24           7561                  INC     H               ;Move to right
   F6A8 C3F67D       7562                  JP      RESTORE_OLD
                     7563 ;
   F6AB 3AFE7D       7564 TRY_MOVE_DOWN:   LD      A,[Y_MAX]
   F6AE BD           7565                  CP      L
   F6AF 28AA         7566                  JR      Z,EXIT          ;At bottom of window so abort
   F6B1 2C           7567                  INC     L
   F6B2 3AFE7A       7568                  LD      A,[X_MIN]
   F6B5 67           7569                  LD      H,A
   F6B6 C3F67D       7570                  JP      RESTORE_OLD
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7572 ;
                     7573 ; MOVE_LEFT  -  Moves the cursor to the left and if at left edge of
                     7574 ;               window will move up a line an to right edge if not
                     7575 ;               at upper left corner of window
                     7576 ;
    F6B9 2AFEA5      7577 MOVE_LEFT:      LD      HL,[CURSOR]
    F6BC 3AFE7A      7578                 LD      A,[X_MIN]
    F6BF BC          7579                 CP      H
    F6C0 2804        7580                 JR      Z,TRY_SCROLL_UP
    F6C2 25          7581                 DEC     H               ;Move to left
    F6C3 C3F67D      7582                 JP      RESTORE_OLD
                     7583 ;
    F6C6 3AFE7C      7584 TRY_SCROLL_UP:  LD      A,[Y_MIN]
    F6C9 BD          7585                 CP      L
    F6CA 288F        7586                 JR      Z,EXIT          ;Abort if at upper left corner
    F6CC 2D          7587                 DEC     L
    F6CD 3AFE7B      7588                 LD      A,[X_MAX]
    F6D0 67          7589                 LD      H,A
    F6D1 C3F67D      7590                 JP      RESTORE_OLD
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7592 ;
                     7593 ; MOVE_HOME  -  Moves cursor to the upper left corner of the window
                     7594 ;
                     7595 ;
                     7596 ; FORM_FEED  -  Clears the screen and puts the cursor at the upper left
                     7597 ;               corner of the window
    F6D4             7598 FORM_FEED:
    F6D4 3AFE9F      7599                 LD      A,[NUM_LINES]
    F6D7 47          7600                 LD      B,A
    F6D8 2AFEA1      7601                 LD      HL,[UPPER_LEFT]
                     7602
    F6DB CDF74C      7603                 CALL    BLANK_SCREEN
                     7604
    F6DE 3E20        7605                 LD      A,BLANKING_CHAR
    F6E0 32FE79      7606                 LD      [OLDCHAR_],A
                     7607
    F6E3 CDF7D7      7608 MOVE_HOME:      CALL    PUT_OLD_ON_SCREEN  ;Needed only for MOVE_HOME not FORM_FEED
    F6E6 2AFEA1      7609                 LD      HL,[UPPER_LEFT]
    F6E9 C3F650      7610                 JP      SAVE_CURSOR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7612 ;
                     7613 ; ERASE_TO_EOL  -  From cursor position fills rest of line
                     7614 ;                  with BLANKING_CHAR, to end of window only
                     7615 ;                  Will blank char under cursor always.
                     7616 ;
    F6EC             7617 ERASE_TO_EOL:
    F6EC CDF766      7618                 CALL    ERASE_THIS_LINE
    F6EF C3F656      7619                 JP      PUT_CURSOR_ON
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7621 ;
                     7622 ; ERASE_TO_EOS  -  Will erase the screen (window) from the current
                     7623 ;                  cursor postion to the end of the window. The
                     7624 ;                  cussor position is also blanked.
                     7625 ;
                     7626 ;
    F6F2             7627 ERASE_TO_EOS:
    F6F2 CDF766      7628                CALL    ERASE_THIS_LINE   ;Erase to end of current line
    F6F5 2AFEA5      7629                LD      HL,[CURSOR]
    F6F8 2C          7630                INC     L                 ;Move down
    F6F9 3AFE7C      7631                LD      A,[Y_MIN]
    F6FC 4F          7632                LD      C,A
    F6FD 3AFE9F      7633                LD      A,[NUM_LINES]
    F700 81          7634                ADD     A,C
    F701 95          7635                SUB     L                 ;Number of lines left to blank
    F702 CAF656      7636                JP      Z,PUT_CURSOR_ON
    F705 47          7637                LD      B,A
    F706 3AFE7A      7638                LD      A,[X_MIN]
    F709 67          7639                LD      H,A
    F70A CDF74C      7640                CALL    BLANK_SCREEN
    F70D C3F656      7641                JP      PUT_CURSOR_ON
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7643 ;
                     7644 ; GOTO_XY  -  Puts cursor to desired X-Y coord if in window
                     7645 ;
                     7646 ;Input Parameters:
                     7647 ;  D - X location
                     7648 ;  E - Y location
                     7649 ;
    F710             7650 GOTO_XY:
    F710 3AFE7A      7651                  LD      A,[X_MIN]
    F713 BA          7652                  CP      D
    F714 2803        7653                  JR      Z,CHK_X_WITH_MAX
    F716 D2F65B      7654                  JP      NC,EXIT
    F719 3AFE7B      7655 CHK_X_WITH_MAX:  LD      A,[X_MAX]
    F71C BA          7656                  CP      D
    F71D 2803        7657                  JR      Z,CHK_Y_COORD
    F71F DAF65B      7658                  JP      C,EXIT
    F722 3AFE7C      7659 CHK_Y_COORD:     LD      A,[Y_MIN]
    F725 BB          7660                  CP      E
    F726 2803        7661                  JR      Z,CHK_Y_WITH_MAX
    F728 D2F65B      7662                  JP      NC,EXIT
    F72B 3AFE7D      7663 CHK_Y_WITH_MAX:  LD      A,[Y_MAX]
    F72E BB          7664                  CP      E
    F72F 2803        7665                  JR      Z,SET_CURSOR
    F731 DAF65B      7666                  JP      C,EXIT
    F734 EB          7667 SET_CURSOR:      EX      DE,HL
    F735 C3F67D      7668                  JP      RESTORE_OLD
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7670 ;
                     7671 ;Internal Subroutines
                     7672 ;
                     7673 ;
                     7674 ; Subroutine
                     7675 ; BLANK_PORTION  -  Fills LINEBUFFER_ with number of chars in B
                     7676 ;                 from where HL points
                     7677 ;
                     7678 ; Input Parameters:
                     7679 ;   HL - Pointer into LINEBUFFER_
                     7680 ;   B  - Number of the BLANKING_CHAR's to stuff in
                     7681 ;
                     7682 ;   A is destroyed
                     7683 ;
   F738              7684 BLANK_PORTION:
   F738 E5           7685                 PUSH    HL
   F739 C5           7686                 PUSH    BC
   F73A 1807         7687                 JR      BLANK_REST
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7689 ;
                     7690 ; Subroutine
                     7691 ; BLANKS       -  Fills LINEBUFFER_ with 32 BLANKING_CHAR's
                     7692 ;
                     7693 ;  A is destroyed
                     7694 ;
    F73C             7695 BLANKS:
    F73C E5          7696                 PUSH    HL
    F73D C5          7697                 PUSH    BC
    F73E 21FE7E      7698                 LD      HL,LINEBUFFER_
    F741 0620        7699                 LD      B,32
    F743 3E20        7700 BLANK_REST:     LD      A,BLANKING_CHAR
    F745 77          7701 BLANK_FILL:     LD      (HL),A
    F746 23          7702                 INC     HL
    F747 10FC        7703                 DJNZ    BLANK_FILL
    F749 C1          7704                 POP     BC
    F74A E1          7705                 POP     HL
    F74B C9          7706                 RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7708 ;
                     7709 ; Subroutine
                     7710 ; BLANK_SCREEN  -  Will blank a given number of lines on the screen
                     7711 ;
                     7712 ; Input Parameters:
                     7713 ;   B  - Number of lines to blank
                     7714 ;   HL - Absolute X-Y coords of start position of 1st line to blank
                     7715 ;
                     7716 ;   Destroys C, A, DE
                     7717 ;   User must reset cursor
                     7718 ;
   F74C              7719 BLANK_SCREEN:
   F74C CDF73C       7720                 CALL    BLANKS          ;Fill buffer with BLANKING_CHAR
   F74F 3AFEA0       7721                 LD      A,[NUM_COLUMNS]
   F752 4F           7722                 LD      C,A
                     7723
   F753 C5           7724 BLNK_SCRN_LOOP: PUSH    BC
   F754 E5           7725                 PUSH    HL              ;Save X-Y coord of line
   F755 CDF7FC       7726                 CALL    CONVERT_XY
   F758 0600         7727                 LD      B,00H           ;C has number of bytes to read
   F75A 21FE7E       7728                 LD      HL,LINEBUFFER_
   F75D CDFD1A       7729                 CALL    WRITE_VRAM
   F760 E1           7730                 POP     HL
   F761 2C           7731                 INC     L               ;To next line below
   F762 C1           7732                 POP     BC
   F763 10EE         7733                 DJNZ    BLNK_SCRN_LOOP
   F765 C9           7734                 RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7736 ;
                     7737 ; Subroutine
                     7738 ; ERASE_THIS_LINE  -  This routine will erase the current line
                     7739 ;                     from the cursor position to the right edge
                     7740 ;                     of the window, the char under the cursor is
                     7741 ;                     also blanked.
                     7742 ;
                     7743 ;Input Parameters:
                     7744 ;   None
                     7745 ;
                     7746 ;  Destroys all registers
                     7747 ;  Sets OLDCHAR_ to BLANKING_CHAR
                     7748 ;
F766                 7749 ERASE_THIS_LINE:
F766 CDF779          7750                 CALL    READ_FROM_CURSOR_TO_EOL
F769 E5              7751                 PUSH    HL
F76A C5              7752                 PUSH    BC
F76B 41              7753                 LD      B,C
F76C CDF738          7754                 CALL    BLANK_PORTION
F76F C1              7755                 POP     BC
F770 E1              7756                 POP     HL
                     7757
F771 3E20            7758                 LD      A,BLANKING_CHAR
F773 32FE79          7759                 LD      (OLDCHAR_),A
                     7760
F776 C3FD1A          7761                 JP      WRITE_VRAM      ;Write line with blanks, return thru WRITE_VRAM
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7763 ;
                     7764 ;Subroutine
                     7765 ;READ_FROM_CURSOR_TO_EOL  -  reads the line on screen from the
                     7766 ;                            cursor position to the right edge of the
                     7767 ;                            screen
                     7768 ;
                     7769 ;Input Parameters:
                     7770 ;  None
                     7771 ;
                     7772 ;Returns:
                     7773 ;   BC - number of chars read
                     7774 ;   HL - address of LINEBUFFER_
                     7775 ;   DE - VRAM address of place cursor is
                     7776 ;
   F779              7777 READ_FROM_CURSOR_TO_EOL:
   F779 2AFEA5       7778                 LD      HL,[CURSOR]
   F77C 3AFE7B       7779                 LD      A,[X_MAX]
   F77F 3C           7780                 INC     A               ;Calc # chars to read
   F780 94           7781                 SUB     H
   F781 4F           7782                 LD      C,A
   F782 CDF7FC       7783                 CALL    CONVERT_XY
   F785 21FE7E       7784                 LD      HL,LINEBUFFER_
   F788 0600         7785                 LD      B,00H
   F78A C5           7786                 PUSH    BC
   F78B E5           7787                 PUSH    HL
   F78C D5           7788                 PUSH    DE
   F78D CDFD1D       7789                 CALL    READ_VRAM
   F790 D1           7790                 POP     DE
   F791 E1           7791                 POP     HL
   F792 C1           7792                 POP     BC
   F793 C9           7793                 RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7795 ;
                     7796 ;Subroutine
                     7797 ; SCROLL_UP  -  Scrolls the entire window up 1 line and puts a line
                     7798 ;               of BLANKING_CHAR's on the last line
                     7799 ;
                     7800 ;   Caller must restore cursor to screen
                     7801 ;
    F794             7802 SCROLL_UP:
    F794 CDF7D7      7803                 CALL    PUT_OLD_ON_SCREEN
    F797 2AFEA1      7804                 LD      HL,[UPPER_LEFT]
    F79A E5          7805                 PUSH    HL
    F79B CDF7FC      7806                 CALL    CONVERT_XY
    F79E E1          7807                 POP     HL
    F79F 3AFE9F      7808                 LD      A,[NUM_LINES]
    F7A2 47          7809                 LD      B,A
    F7A3 05          7810                 DEC     B
                     7811
    F7A4 C5          7812 SCROLL_LOOP:    PUSH    BC              ;Save number of remaining lines
    F7A5 D5          7813                 PUSH    DE              ;Save next write location
    F7A6 2C          7814                 INC     L               ;Point to next line below
    F7A7 E5          7815                 PUSH    HL
    F7A8 CDF7FC      7816                 CALL    CONVERT_XY
    F7AB D5          7817                 PUSH    DE              ;Save address of this read
    F7AC 3AFEA0      7818                 LD      A,[NUM_COLUMNS]
    F7AF 4F          7819                 LD      C,A
    F7B0 0600        7820                 LD      B,00H
    F7B2 C5          7821                 PUSH    BC              ;Save number of chars in line
    F7B3 21FE7E      7822                 LD      HL,LINEBUFFER_
    F7B6 CDFD1D      7823                 CALL    READ_VRAM
    F7B9 C1          7824                 POP     BC              ;Get number of bytes that were read
    F7BA D1          7825                 POP     DE              ;Get address of last read
    F7BB E1          7826                 POP     HL              ;Get X-Y coords
    F7BC E3          7827                 EX      [SP],HL         ;Swap prev read address with X-Y coords
    F7BD EB          7828                 EX      DE,HL           ;Swap last read with previous address
    F7BE E5          7829                 PUSH    HL              ;  and save previous
    F7BF 21FE7E      7830                 LD      HL,LINEBUFFER_
    F7C2 CDFD1A      7831                 CALL    WRITE_VRAM      ;Write line one line up
    F7C5 D1          7832                 POP     DE              ;Get address of next write
    F7C6 E1          7833                 POP     HL              ;Get starting X-Y of last read
    F7C7 C1          7834                 POP     BC
    F7C8 10DA        7835                 DJNZ    SCROLL_LOOP
                     7836
    F7CA CDF73C      7837                 CALL    BLANKS
    F7CD 21FE7E      7838                 LD      HL,LINEBUFFER_
    F7D0 3AFEA0      7839                 LD      A,[NUM_COLUMNS]
    F7D3 4F          7840                 LD      C,A             ;B is 0 from DJNZ
    F7D4 C3FD1A      7841                 JP      WRITE_VRAM      ;Return thru WRITE_VRAM
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7843 ;
                     7844 ; Subroutines
                     7845 ; PUT_OLD_ON_SCREEN  - This entry point puts the char that the
                     7846 ;                      cursor was on back onto the screen
                     7847 ;
                     7848 ; PUT_CHAR_ON_SCREEN _ This entry point will put the char in A
                     7849 ;                      onto the screen where the cursor currently is
                     7850 ;
                     7851 ;
F7D7                 7852 PUT_OLD_ON_SCREEN:
F7D7 3AFE79          7853                 LD      A,[OLDCHAR_]
                     7854
F7DA                 7855 PUT_CHAR_ON_SCREEN:
F7DA E5              7856                 PUSH    HL
F7DB 2AFEA5          7857                 LD      HL,[CURSOR]     ;Get location to put char
F7DE CDF7FC          7858                 CALL    CONVERT_XY      ;Convert to a VRAM address
F7E1 21FE7E          7859                 LD      HL,LINEBUFFER_
F7E4 77              7860                 LD      [HL],A
F7E5 010001          7861                 LD      BC,0001H
F7E8 CDFD1A          7862                 CALL    WRITE_VRAM
F7EB E1              7863                 POP     HL
F7EC C9              7864                 RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7866 ;
                     7867 ; Subroutine
                     7868 ; GET_CHAR_UNDER_CURSOR  -  This entry loads HL from CURSOR before
                     7869 ;                           falling into GET_CHAR_UNDER
                     7870 ;
                     7871 ; GET_CHAR_UNDER  -  Reads the char where the cursor will go
                     7872 ;                    and saves it in the variable OLDCHAR_
                     7873 ;
                     7874 ;Input Parameters:
                     7875 ;  HL - Cursor position
                     7876 ;
                     7877 ; Destroys - HL, BC, DE, A
                     7878 ;
     F7ED            7879 GET_CHAR_UNDER_CURSOR:
     F7ED 2AFEA5     7880                 LD      HL,[CURSOR]
                     7881
     F7F0            7882 GET_CHAR_UNDER:
     F7F0 CDF7FC     7883                 CALL    CONVERT_XY
     F7F3 010001     7884                 LD      BC,0001H
     F7F6 21FE79     7885                 LD      HL,OLDCHAR_
     F7F9 C3FD1D     7886                 JP      READ_VRAM       ;Return thru READ_VRAM
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7888 ;
                     7889 ; Subroutine
                     7890 ;
                     7891 ; CONVERT_XY   - Converts X-Y coordinates on the screen to a
                     7892 ;                VRAM address for that char position
                     7893 ;                Calculates (( Y * 32) + X ) + base of pattern name table
                     7894 ;                to yield the absolute address in VRAM of the char
                     7895 ;
                     7896 ;Input Parameters:
                     7897 ;  H - X coord
                     7898 ;  L - Y coord
                     7899 ;
                     7900 ;Returns:
                     7901 ;  DE - VRAM address
                     7902 ;  HL - Address of Pattern Name Table (side effect)
                     7903 ;  No Others are altered
                     7904 ;
    F7FC             7905 CONVERT_XY:
    F7FC 5C          7906                 LD      E,H             ;Save X coord
    F7FD 2600        7907                 LD      H,00H
    F7FF 29          7908                 ADD     HL,HL           ;Calc Y coord * 32
    F800 29          7909                 ADD     HL,HL
    F801 29          7910                 ADD     HL,HL
    F802 29          7911                 ADD     HL,HL
    F803 29          7912                 ADD     HL,HL
    F804 1600        7913                 LD      D,00H
    F806 19          7914                 ADD     HL,DE           ;Add in X coord
    F807 ED5BFEA3    7915                 LD      DE,[PTRN_NAME_TBL]
    F808 19          7916                 ADD     HL,DE
    F80C EB          7917                 EX      DE,HL
    F80D C9          7918                 RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7920 ;
                     7921 ; This is a table of chars that perform some function rather
                     7922 ; than being printed out to the screen as is.  There is a
                     7923 ; jump table that is asscociated with this table, but NOTE that
                     7924 ; it has its entrys in reverse order of this table.  It would
                     7925 ; be best to add to the end of this table and to the beginning
                     7926 ; of the jump table.
                     7927 ;
    F80E             7928 SPECIAL_CHARS:
    F80E 08          7929                  DEFB    BS
    F80F 0D          7930                  DEFB    CR
    F810 0A          7931                  DEFB    LF
    F811 0C          7932                  DEFB    FF
    F812 80          7933                  DEFB    HOME
    F813 16          7934                  DEFB    CLR_TO_EOL
    F814 18          7935                  DEFB    CLR_TO_EOS
    F815 1C          7936                  DEFB    GOTO_XY_CHAR
    F816 A0          7937                  DEFB    UP
    F817 A2          7938                  DEFB    DOWN
    F818 A3          7939                  DEFB    LEFT
    F819 A1          7940                  DEFB    RIGHT
                     7941
            <000C>   7942 NUM_SPCL_CHARS   EQU     $-SPECIAL_CHARS
                     7943
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7945 ;
                     7946 ;NOTE - This table is in the reverse order of the chars so
                     7947 ;       any additions to the end of the chars goes at the top
                     7948 ;       of this table
                     7949 ;
    F81A             7950 SPCL_VECTOR_TBL:
    F81A F69E        7951                 DEFW    MOVE_RIGHT
    F81C F689        7952                 DEFW    MOVE_LEFT
    F81E F683        7953                 DEFW    MOVE_DOWN
    F820 F673        7954                 DEFW    MOVE_UP
    F822 F710        7955                 DEFW    GOTO_XY
    F824 F6F2        7956                 DEFW    ERASE_TO_EOS
    F826 F6EC        7957                 DEFW    ERASE_TO_EOL
    F828 F6E3        7958                 DEFW    MOVE_HOME
    F82A F6D4        7959                 DEFW    FORM_FEED
    F82C F683        7960                 DEFW    MOVE_DOWN
    F82E F664        7961                 DEFW    RETURN
    F830 F689        7962                 DEFW    MOVE_LEFT
                     7963
                     7964 ;
                     7965 ;   CODE FROM HERE TO END OF CONS_OUT WAS INSERTED AFTER THE
                     7966 ;   REV. 06 ROM WAS BURNED AND HAS BEEN COMMENTED OUT TO MAINTAIN
                     7967 ;   COMPATIBILITY WITH THAT ROM.
                     7968 ;
                     7969 ;                END
                     7970 ;  SKIP
                     7971 ;;
                     7972 ;; RETURN_CURSOR  -  Will return the current cursor position
                     7973 ;;                   to the caller.  It is in absolute screen
                     7974 ;;                   coordinates (0 based).
                     7975 ;;
                     7976 ;;Input Parameters:
                     7977 ;;  None
                     7978 ;;
                     7979 ;;Returns:
                     7980 ;;  D - X coord of cursor, absolute based on upper left of screen not window
                     7981 ;;  E - Y coord of cursor
                     7982 ;;
                     7983 ;               ;GLB     RETURN_CURSOR
                     7984 ;RETURN_CURSOR:
                     7985 ;                LD      DE,[CURSOR]
                     7986 ;                RET
                     7987 ;  SKIP
                     7988 ;;
                     7989 ;; RETURN_RELATIVE_CURSOR  - Returns the current cursor position relative
                     7990 ;;                           to the window.  0,0 will be returned for the
                     7991 ;;                           upper left corner of the window.
                     7992 ;;
                     7993 ;;Input Parameters:
                     7994 ;;  NONE
                     7995 ;;
                     7996 ;;Returns:
                     7997 ;;   D  -  X cursor position relative to window
                     7998 ;;   E  -  Y cussor position relative to window
                     7999 ;;
                     8000 ;;   Destroys A
                     8001 ;;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8002 ;
                     8003 ;                ;GLB     RETURN_RELATIVE_CURSOR
                     8004 ;
                     8005 ;RETURN_RELATIVE_CURSOR:
                     8006 ;                LD      DE,[CURSOR]     ;get absolute position
                     8007 ;                LD      A,[X_MIN]
                     8008 ;                NEG
                     8009 ;                ADD     A,D
                     8010 ;                LD      D,A
                     8011 ;                LD      A,[Y_MIN]
                     8012 ;                NEG
                     8013 ;                ADD     A,E
                     8014 ;                LD      E,A
                     8015 ;                RET
                     8016 ;;
                     8017 ;;*****************************************************************
                     8018 ;;*****************************************************************
                     8019 ;;*****************************************************************
                     8020 ;;
                     8021
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8023  NAME ^Rev 07 - rfj^
                     8024
                     8025 De_EOS_START MACRO              ;Header Rev. 4
                     8026                 .GOTO Ede_EOS_START
                     8027
                     8028  Project:      EOS, 83-101
                     8029
                     8030  |&k1S                                  |&kOS
                     8031  |&k1S                                  |&kOS
                     8032  |&k1S   EOS_START           DLS       |&kOS
                     8033  |&k1S                                  |&kOS
                     8034  |&k1S                                  |&kOS
                     8035
                     8036        Rev History
                     8037        Rev.  Date          Name      Change
                     8038         7    24oct1951     rfj       Changed value for REV_NUM to 06
                     8039         6    13oct1045a    RPD       added init of REV_NUM
                     8040         5    1oct1005a     RPD       took out memory bank switch logic for GOTO_WP
                     8041         4    25sept1000a   RPD       reworked for new boot from tape logic
                     8042         3    24sept1110a   RPD       moved clear ram logic
                     8043         2    23sept1255p   RPD       moved FCB's and added calls A_uOS routines
                     8044         1    03SEP1445     DLS       MORE CODE
                     8045         0    02SEP0630     DLS       CODE
                     8046
                     8047 Ede_EOS_START  MEND
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    8049 ************************************************************
                    8050 *                                                          *
                    8051 *  MODULE NAME:                                            *
                    8052 *                                                          *
                    8053 *    EOS_START                                             *
                    8054 *                                                          *
                    8055 *  INPUTS:                                                 *
                    8056 *                                                          *
                    8057 *    NONE                                                  *
                    8058 *                                                          *
                    8059 *  FUNCTION(S):                                            *
                    8060 *     0. SET STACK.                                        *
                    8061 *     1. TO BANK SELECT OS7 AND 24K RAM.                   *
                    8062 *     2. CALL INIT_EOS TO ALLOW THE 1) THE Z80/6801 SYNCH_UP,  *
                    8063 *        POLL FOR NET DEVICES AND ESTABLISH DCB'S.         *
                    8064 *     3. CHECK FOR THE PRESENCE OF A TAPE. IF PRESENT THEN *
                    8065 *        LOAD IN A COLD_START_LOADER AND EXECUTE FROM THERE.   *
                    8066 *     4. IF NO TAPE, THEN BRING UP THE ROM RESIDENT WORD   *
                    8067 *        PROCESSOR BY JUMPING TO OS7.                      *
                    8068 *                                                          *
                    8069 *  OUTPUTS:                                                *
                    8070 *                                                          *
                    8071 *     DEVICE ADDR _ REG B                                  *
                    8072 *                                                          *
                    8073 *  CALLS:                                                  *
                    8074 *                                                          *
                    8075 *    EOS_INIT                                              *
                    8076 *    EOS_QUERY_DEVICE                                      *
                    8077 *    EOS_READ_BLOCK                                        *
                    8078 *                                                          *
                    8079 *  CALLED BY:                                              *
                    8080 *                                                          *
                    8081 *    INVOKED BY EOS_BOOT                                   *
                    8082 *                                                          *
                    8083 *  NOTES:                                                  *
                    8084 *                                                          *
                    8085 *    NONE.                                                 *
                    8086 *                                                          *
                    8087 ************************************************************
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8090 ;
                     8091 ; external routines used
                     8092 ;
                     8093                ;EXT    __FIND_DCB
                     8094                ;EXT    __REQUEST_STATUS
                     8095                ;EXT    __HARD_INIT
                     8096                ;EXT    __RD_1_BLOCK
                     8097                ;EXT    __FMGR_INIT
                     8098                ;EXT    _GOTO_WP
                     8099                ;EXT    FILL_VRAM
                     8100                ;EXT    TURN_OFF_SOUND
                     8101                ;EXT    SWITCH_MEM
                     8102                ;EXT    PORT_COLLECTION
                     8103 ;
                     8104 ; external data structures used
                     8105 ;
                     8106                ;EXT    REV_NUM
                     8107                ;EXT    CLEAR_RAM_START
                     8108                ;EXT    CLEAR_RAM_SIZE
                     8109                ;EXT    EOS_STACK
                     8110                ;EXT    DEFAULT_BT_DEV
                     8111                ;EXT    MEM_CNFGO1
                     8112                ;EXT    MEM_CNFGO3
                     8113 ;
                     8114 ; local equates
                     8115 ;
<0005>               8116 EOS_REV        EQU     005H        ;current EOS revision number
                     8117                                    ;---NOTE---   this is actually rev.
                     8118                                    ;06 but to match the production ROM
                     8119                                    ;which was labeled rev. 05 we fudge
                     8120                                    ;just a bit (two bits actually)
                     8121
<C800>               8122 COLD_START_ADDR EQU    0C800H
<0004>               8123 DSK4           EQU     4
<0005>               8124 DSK5           EQU     5
<0001>               8125 ONE_BLOCK      EQU     1
<0000>               8126 BLOCK_ZERO     EQU     0
                     8127
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8129               ;GLB     __EOS_START
    F832             8130 __EOS_START:
    F832 31FE58      8131               LD       SP,EOS_STACK            ; SET THE STACK
                     8132
    F835 010147      8133               LD       BC,CLEAR_RAM_SIZE       ; clear EOS ram
    F838 11FD61      8134               LD       DE,CLEAR_RAM_START+1
    F83B 21FD60      8135               LD       HL,CLEAR_RAM_START
    F83E AF          8136               XOR      A
    F83F 77          8137               LD       (HL),A
    F840 EDB0        8138               LDIR
                     8139
    F842 3E05        8140               LD       A,EOS_REV               ; init REV_NUM
    F844 32FD60      8141               LD       (REV_NUM),A
                     8142
    F847 CDFD11      8143               CALL     PORT_COLLECTION         ; collect the port addresses from OS7
                     8144
    F84A CDFD53      8145               CALL     TURN_OFF_SOUND          ; KILL SOUNDS
                     8146
    F84D 3E00        8147               LD       A,0                     ; CLEAR VRAM
    F84F 210000      8148               LD       HL,0
    F852 114000      8149               LD       DE,16*1024
    F855 CDFD26      8150               CALL     FILL_VRAM
                     8151
    F858 3AFC18      8152               LD       A,(MEM_CNFG01)          ;get the data for configuration 1 (64K RAM)
    F858 CDFD14      8153               CALL     SWITCH_MEM              ;select this configuration
                     8154 ;
                     8155 ; EOS_INIT WILL PERFORM THE 6801/Z80 SYNCH_UP, PERFORM A ROLL_CALL
                     8156 ; POLL ON THE NET, AND ESTABLISH THE DCB'S
                     8157 ;
    F85E CDF8F6      8158               CALL     __HARD_INIT
                     8159 ;
                     8160 ; setup the FCB's
                     8161 ;
    F861 11D400      8162               LD       DE,THREE1K_BLKS         ; START ADDRESS OF THREE 1K BLOCKS
    F864 21D390      8163               LD       HL,FCB_S                ; START ADDRESS OF 3 FCB BLOCKS
    F867 CDEEEA      8164               CALL     __FMGR_INIT             ; TELL THE FILE MANAGER
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8166 ;
                     8167 ;    CHECK FOR BOOT LOADER IN ANY MASS STORAGE DEVICE
                     8168 ;
    F86A 3E08        8169                LD      A,TAPE_ID               ;set the default boot device to tape
    F86C 32FD6F      8170                LD      [DEFAULT_BT_DEV],A
                     8171
    F86F 3E04        8172                LD      A,DSK4                  ;check for DISK 4 device active
    F871 CDF473      8173                CALL    __REQUEST_STATUS
    F874 2010        8174                JR      NZ,CHK_D5
                     8175
    F876 3E04        8176                LD      A,DSK4
    F878 CDF446      8177                CALL    __FIND_DCB
    F87B FD7E14      8178                LD      A,[IY+D_STATUS_FLAGS]
    F87E E60F        8179                AND     0FH
    F880 FE03        8180                CP      3
    F882 3E04        8181                LD      A,DSK4
    F884 3842        8182                JR      C,DEV1_OK
                     8183
    F886 3E05        8184 CHK_D5:        LD      A,DSK5                  ;check for DISK 5 device active
    F888 CDF473      8185                CALL    __REQUEST_STATUS
    F88B 2010        8186                JR      NZ,CHK_TAPE
                     8187
    F88D 3E05        8188                LD      A,DSK5
    F88F CDF446      8189                CALL    __FIND_DCB
    F892 FD7E14      8190                LD      A,[IY+D_STATUS_FLAGS]
    F895 E60F        8191                AND     0FH
    F897 FE03        8192                CP      3
    F899 3E05        8193                LD      A,DSK5
    F89B 3828        8194                JR      C,DEV1_OK
    F89D             8195 CHK_TAPE:
    F89D 3E08        8196                LD      A,TAPE_ID               ;check if TAPE device active
    F89F CDF473      8197                CALL    __REQUEST_STATUS
    F8A2 201E        8198                JR      NZ,CALL_WP
                     8199
    F8A4 3E08        8200                LD      A,TAPE_ID
    F8A6 CDF446      8201                CALL    __FIND_DCB
    F8A9 FD7E14      8202                LD      A,[IY+D_STATUS_FLAGS]
    F8AC F5          8203                PUSH    AF
    F8AD E60F        8204                AND     0FH
    F8AF FE03        8205                CP      3
    F8B1 3812        8206                JR      C,DEVO_OK
                     8207
    F8B3 F1          8208                POP     AF
    F8B4 CB3F        8209                SRL     A
    F8B6 CB3F        8210                SRL     A
    F8B8 CB3F        8211                SRL     A
    F8BA CB3F        8212                SRL     A
    F8BC FE03        8213                CP      3
    F8BE 3E18        8214                LD      A,TAPE_ID+10H
    F8C0 3806        8215                JR      C,DEV1_OK
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F8C2             8217 CALL_WP:                                  ;devices disk 4, 5 and tape are not active
                     8218 ;              LD      A,[MEM_CNFG03]     ;get the data for configuration 3 (8K OS7 and 56K RAM)
                     8219 ;              CALL    SWITCH_MEM         ;select this configuration
    F8C2 C3FCE7      8220                JP      _GOTO_WP           ;default to ALPHA
    F8C5             8221 DEVO_OK:
    F8C5 F1          8222                POP     AF
    F8C6 3E08        8223                LD      A,TAPE_ID
    F8C8             8224 DEV1_OK:
    F8C8 32FD6F      8225                LD      [DEFAULT_BT_DEV],A
    F8CB             8226 READ_AGAIN:
    F8CB 21C800      8227                LD      HL,COLD_START_ADDR
    F8CE 010000      8228                LD      BC,0
    F8D1 3AFD6F      8229                LD      A,[DEFAULT_BT_DEV]
    F8D4 110000      8230                LD      DE,BLOCK_ZERO
    F8D7 CDFA9E      8231                CALL    __RD_1_BLOCK
    F8DA CAF8EF      8232                JP      Z,GO_TO_TAPECODE
    F8DD 4F          8233                LD      C,A
    F8DE 3AFD6F      8234                LD      A,[DEFAULT_BT_DEV]
    F8E1 E60F        8235                AND     0FH
    F8E3 FE08        8236                CP      TAPE_ID
    F8E5 20E4        8237                JR      NZ,READ_AGAIN
    F8E7 79          8238                LD      A,C
    F8E8 FE98        8239                CP      TIMEOUT
    F8EA 28DF        8240                JR      Z,READ_AGAIN
    F8EC C3F8C2      8241                JP      CALL_WP
    F8EF             8242 GO_TO_TAPECODE:
                     8243
    F8EF 3AFD6F      8244                LD      A,[DEFAULT_BT_DEV]
    F8F2 47          8245                LD      B,A
    F8F3 C3C800      8246                JP      COLD_START_ADDR
                     8247 ;
                     8248 ;*****************************************************************
                     8249 ;*****************************************************************
                     8250 ;*****************************************************************
                     8251 ;
                     8252
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8254 ;
                     8255 ;       Rev History
                     8256 ;       Rev.  Date         Name      Change
                     8257 ;        3    5oct1135a    RPD       WP equate changed to 100H
                     8258 ;        2    2oct155p     RPD       added initialization of reserved byte at OFFFFH to 0
                     8260 ;        0    aug/sept     VB        original
                     8261 ;
                     8262
                     8263 ; ENTRY:        __HARD_INIT
                     8264 ; NEEDS:        NOTHING
                     8265 ; RETURNS:      DEFAULT PCB ADDRESS SET UP FOR USE ([CURRENT_PCB] = PCB)
                     8266 ;               HARD RESET APPLIED TO NETWORK
                     8267 ;               NETWORK MASTER NODE IN SYNC WITH Z80
                     8268 ;               1 DCB ALLOTTED FOR EACH DEVICE ANSWERING A STATUS REQUEST
                     8269 ;                CORRECTLY
                     8270 ;               MASTER NODE SCANNING EACH OF THESE DCB'S FOR WORK TO DO
                     8271
                     8272       ;GLB     __HARD_INIT
                     8273
                     8274       ;EXT     CURRENT_PCB
                     8275       ;EXT     PCB
                     8276
F8F6                 8277 __HARD_INIT
F8F6 C5              8278       PUSH     BC                      ; SAVE BC
F8F7 D5              8279       PUSH     DE                      ;      DE
F8F8 E5              8280       PUSH     HL                      ;      HL
F8F9 FDE5            8281       PUSH     IY                      ;      IY
                     8282
F8FB 21FECO          8283       LD       HL,PCB                  ; INIT OUR IDEA OF CURRENT_PCB
F8FE 22FD70          8284       LD       [CURRENT_PCB],HL        ;
                     8285
F901 CDF94B          8286       CALL     __HARD_RESET_NET        ; HARD RESET NETWORK
                     8287
F904 CDF95F          8288       CALL     __DLY_AFT_HRD_RES       ; GOOD DELAY AFTER HARD
                     8289                                        ;  NET RESET
                     8290
F907 21FECO          8291       LD       HL,PCB                  ; ZERO OUT PCB/DCB'S
F90A 11FEC1          8292       LD       DE,PCB+1                ;
                     8293 ;     LD       BC,P_SIZE+(15*D_SIZE)-1 ;
F90D 01013F          8294       LD       BC,P_SIZE+(15*D_SIZE)   ;this byte count will zero out PCB, DCB's and the reserved byte
F910 3600            8295       LD       [HL],0                  ;
F912 EDB0            8296       LDIR                             ;
                     8297
F914                 8298 REPEAT_SYNC1:
F914 CDF970          8299       CALL     __SYNC
F917 20FB            8300       JR       NZ,REPEAT_SYNC1
                     8301
F919 CDF9CB          8302       CALL     __SCAN_ACTIVE             FIND ALL THE ACTIVE DEVICES ON THE NET
                     8303
F91C FDE1            8304       POP      IY                      ; RESTORE IY
F91E E1              8305       POP      HL                      ;         HL
F91F D1              8306       POP      DE                      ;         DE
F920 C1              8307       POP      BC                      ;         BC
                     8308
F921 C9              8309       RET
                     8310
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8312
                     8313 ; ENTRY:        __SOFT_INIT
                     8314 ; NEEDS:        HL = PCB ADDRESS SET UP FOR USE
                     8315 ; RETURNS:      [CURRENT_PCB] = HL
                     8316 ;               HARD RESET APPLIED TO NETWORK
                     8317 ;               NETWORK MASTER NODE IN SYNC WITH Z80
                     8318 ;               1 DCB ALLOTTED FOR EACH DEVICE ANSWERING A STATUS REQUEST
                     8319 ;                CORRECTLY
                     8320 ;               MASTER NODE SCANNING EACH OF THESE DCB'S FOR WORK TO DO
                     8321
                     8322         ;GLB    __SOFT_INIT
                     8323
F922                 8324 __SOFT_INIT
F922 C5              8325         PUSH    BC                      ; SAVE BC
F923 D5              8326         PUSH    DE                      ;      DE
F924 E5              8327         PUSH    HL                      ;      HL
F925 FDE5            8328         PUSH    IY                      ;      IY
                     8329
F927 22FD70          8330         LD      [CURRENT_PCB],HL        ; ESTABLISH CURRENT_PCB ADDR
                     8331
F92A CDF94B          8332         CALL    __HARD_RESET_NET        ; HARD RESET NETWORK
                     8333
F92D CDF95F          8334         CALL    __DLY_AFT_HRD_RES       ;
                     8335
F930 2AFD70          8336         LD      HL,[CURRENT_PCB]        ; ZERO OUT PCB/DCB'S
F933 5D              8337         LD      E,L
F934 54              8338         LD      D,H
F935 13              8339         INC     DE
F936 01013E          8340         LD      BC,P_SIZE+(15*D_SIZE)-1 ;
F939 3600            8341         LD      [HL],0                  ;
F93B EDB0            8342         LDIR                            ;
                     8343
F93D                 8344 REPEAT_SYNC2:
F93D CDF970          8345         CALL    __SYNC
F940 20FB            8346         JR      NZ,REPEAT_SYNC2
                     8347
F942 CDF9CB          8348         CALL    __SCAN_ACTIVE             FIND ALL THE ACTIVE DEVICES ON THE NET
                     8349
F945 FDE1            8350         POP     IY                      ; RESTORE IY
F947 E1              8351         POP     HL                      ;         HL
F948 D1              8352         POP     DE                      ;         DE
F949 C1              8353         POP     BC                      ;         BC
                     8354
F94A C9              8355         RET
                     8356
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8358
                     8359 ; ENTRY:        __HARD_RESET_NET
                     8360 ; NEEDS:        NOTHING
                     8361 ; RETURNS:      HARD RESET APPLIED TO NETWORK
                     8362
            <0000>   8363 NET_DELAY_COUNT EQU     0
                     8364
                     8365        ;GLB     __HARD_RESET_NET
                     8366
    F94B             8367 __HARD_RESET_NET:
                     8368        ;EXT     NET_RESET_PORT
    F94B 3AFC28      8369         LD      A,[NET_RESET_PORT]
    F94E 4F          8370         LD      C,A
    F94F 3E0F        8371         LD      A,0FH                    ; RESET NETWORK
    F951 ED79        8372         OUT     [C],A
                     8373
    F953 3E00        8374         LD      A,NET_DELAY_COUNT        ; DELAY (***** PROBABLY TOO LONG)
                     8375
    F955             8376 NET_RES_DELAY:
    F955 00          8377         NOP
    F956 00          8378         NOP
    F957 00          8379         NOP
    F958 3D          8380         DEC     A
    F959 20FA        8381         JR      NZ,NET_RES_DELAY
                     8382
    F95B AF          8383         XOR      A                       ; HELLLLLLLOOOOOO NETWORK .....
    F95C ED79        8384         OUT     [C],A
                     8385
    F95E C9          8386         RET
                     8387
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8389
                     8390       ;GLB     __DLY_AFT_HRD_RES
                     8391
   F95F              8392 __DLY_AFT_HRD_RES:
   F95F C5           8393       PUSH     BC
   F960 D5           8394       PUSH     DE
                     8395
   F961 0601         8396       LD       B,1
                     8397
   F963              8398 LOOP8:
   F963 110001       8399       LD       DE,0001H
                     8400
   F966              8401 LOOP7:
   F966 1B           8402       DEC      DE
   F967 7A           8403       LD       A,D
   F968 B3           8404       OR       E
   F969 20FB         8405       JR       NZ,LOOP7
                     8406
   F96B 10F6         8407       DJNZ     LOOP8
                     8408
   F96D D1           8409       POP      DE
   F96E C1           8410       POP      BC
                     8411
   F96F C9           8412       RET
                     8413
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8415
                     8416 ; ENTRY:        __SYNC
                     8417 ; NEEDS:        [CURRENT_PCB] SET TO ADDRESS OF CURRENTLY OPERATING PCB
                     8418 ; RETURNS:      NETWORK MASTER NODE IN SYNC WITH Z80
                     8419 ;               NUMBER OF DCB'S = 0 (SO MASTER NODE IS NOT SCANNING
                     8420 ;                ANY DCB'S - SO A __SCAN_ACTIVE NEEDS TO BE DONE
                     8421 ;                TO ESTABLISH DCB'S)
                     8422
           <0000>    8423 SYNC_DELAY_DE EQU 0
           <0002>    8424 SYNC_DELAY_B  EQU 2
                     8425
                     8426         ;GLB     __SYNC
                     8427
F970                 8428 __SYNC
F970 FDE5            8429         PUSH    IY                      ; SAVE IY
F972 E5              8430         PUSH    HL                      ;      HL
F973 C5              8431         PUSH    BC
F974 D5              8432         PUSH    DE
                     8433
F975 FD2AFD70        8434         LD      IY,[CURRENT_PCB]        ; CURRENT PCB
                     8435
F979 FD360300        8436         LD      [IY+P_NUM_DCBS],0       ; 0 DCB'S
                     8437
F97D FD360001        8438         LD      [IY+P_COM_STAT],PCB_SYNC1       ; START SYNC WITH FIRST
                     8439                                                  ;  BYTE
                     8440
F981 110000          8441         LD      DE,SYNC_DELAY_DE
F984 0602            8442         LD      B,SYNC_DELAY_B
                     8443
F986                 8444 LOOP1:
F986 1B              8445         DEC     DE
F987 7A              8446         LD      A,D
F988 B3              8447         OR      E
F989 200A            8448         JR      NZ,NOT_TIME1
                     8449
F98B 110000          8450         LD      DE,SYNC_DELAY_DE
                     8451
F98E 1005            8452         DJNZ    NOT_TIME1
                     8453
F990 3E12            8454         LD      A,CANT_SYNC1
F992 B7              8455         OR      A                       ; ERROR
                     8456
F993 1830            8457         JR      END_SYNC
                     8458
F995                 8459 NOT_TIME1:
F995 FD7E00          8460         LD      A,[IY+P_COM_STAT]       ; DID MASTER NODE ACKNOWLEDGE ?
F998 FE81            8461         CP      PCB_SYNC1_ACK
F99A 20EA            8462         JR      NZ,LOOP1                ; ***** NEED PROTECTION HERE
                     8463
F99C FD360002        8464         LD      [IY+P_COM_STAT],PCB_SYNC2       ; CONTINUE SYNC WITH
                     8465                                                  ;  SECOND BYTE
                     8466
F9A0 110000          8467         LD      DE,SYNC_DELAY_DE
F9A3 0602            8468         LD      B,SYNC_DELAY_B
                     8469
F9A5                 8470 LOOP2:
F9A5 1B              8471         DEC     DE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    F9A6 7A          8472         LD      A,D
    F9A7 B3          8473         OR      E
    F9A8 200A        8474         JR      NZ,NOT_TIME2
                     8475
    F9AA 110000      8476         LD      DE,SYNC_DELAY_DE
                     8477
    F9AD 1005        8478         DJNZ    NOT_TIME2
                     8479
    F9AF 3E13        8480         LD      A,CANT_SYNC2
    F9B1 B7          8481         OR      A                       ; ERROR
                     8482
    F9B2 1811        8483         JR      END_SYNC
                     8484
    F9B4             8485 NOT_TIME2:
    F9B4 FD7E00      8486         LD      A,[IY+P_COM_STAT]         ; DID MASTER NODE ACKNOWLEDGE ?
    F9B7 FE82        8487         CP      PCB_SYNC2_ACK
    F9B9 20EA        8488         JR      NZ,LOOP2                ; ***** NEED PROTECTION HERE
                     8489
    F9BB FDE5        8490         PUSH    IY                      ; GET PCB ADDR INTO HL
    F9BD E1          8491         POP     HL
                     8492
    F9BE FD7501      8493         LD      [IY+P_REL_ADDR_LO],L    ; IN CASE SOMEONE ACCIDENTALLY DOES
    F9C1 FD7402      8494         LD      [IY+P_REL_ADDR_HI],H    ;  A PCB RELOCATION COMMAND -
                     8495                                         ;  IT WILL RELOCATE TO THIS SPOT
                     8496
    F9C4 AF          8497         XOR     A                       ; NO ERROR
                     8498
    F9C5             8499 END_SYNC:
    F9C5 D1          8500         POP     DE
    F9C6 C1          8501         POP     BC
    F9C7 E1          8502         POP     HL                      ; RESTORE HL
    F9C8 FDE1        8503         POP     IY                      ;         IY
                     8504
    F9CA C9          8505         RET
                     8506
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8508
                     8509 ; ENTRY:        __SCAN_ACTIVE
                     8510 ; NEEDS:        [CURRENT_PCB] SET TO CURRENTLY OPERATING PCB
                     8511 ; RETURNS:      1 DCB ALLOTTED FOR EACH DEVICE ANSWERING A STATUS REQUEST
                     8512 ;                CORRECTLY
                     8513 ;               MASTER NODE SCANNING EACH OF THESE DCB'S FOR WORK TO DO
                     8514
                     8515         ;GLB    __SCAN_ACTIVE
                     8516
F9CB                 8517 __SCAN_ACTIVE
F9CB C5              8518         PUSH    BC                      ; SAVE BC
F9CC D5              8519         PUSH    DE                      ;      DE
F9CD E5              8520         PUSH    HL                      ;      HL
F9CE FDE5            8521         PUSH    IY                      ;      IY
F9D0 DDE5            8522         PUSH    IX                      ;      IX
                     8523
F9D2 2AFD70          8524         LD      HL,[CURRENT_PCB]        ; ZERO OUT DCB'S
F9D5 110004          8525         LD      DE,P_SIZE               ; POINT TO DCB'S
F9D8 19              8526         ADD     HL,DE                   ;
F9D9 5D              8527         LD      E,L                     ; WANT DE = [CURRENT_PCB]+P_SIZE+1
F9DA 54              8528         LD      D,H                     ;
F9DB 13              8529         INC     DE                      ;
F9DC 01013A          8530         LD      BC,(15*D_SIZE)-1        ;
F9DF 3600            8531         LD      [HL],0                  ;
F9E1 EDB0            8532         LDIR                            ;
                     8533
F9E3 FD2AFD70        8534         LD      IY,[CURRENT_PCB]        ; FIND START ADDR OF DCB'S
F9E7 110004          8535         LD      DE,P_SIZE               ;
F9EA FD19            8536         ADD     IY,DE                   ;
                     8537
F9EC DD2AFD70        8538         LD      IX,[CURRENT_PCB]        ; IX = [CURRENT_PCB]
F9F0 DD360301        8539         LD      [IX+P_NUM_DCBS],1       ; NEED AT LEAST 1 ACTIVE DCB
                     8540                                         ;  TO START CHECKING
                     8541
F9F4 3E01            8542         LD      A,1                     ; START WITH DEV ADDR 1
                     8543
F9F6                 8544 LOOP3:
F9F6 F5              8545         PUSH    AF                      ; SAVE CURRENT DEVICE ADDR
                     8546
F9F7 FD7710          8547         LD      [IY+D_DEV_ADDR],A       ; SELECT A NEW ADDRESS
                     8548
F9FA FD360001        8549         LD      [IY+D_COM_STAT],DCB_STATUS      ; ISSUE A REQUEST FOR STATUS
                     8550
F9FE                 8551 LOOP4:
F9FE FDCB007E        8552         BIT     CMND_COMPLETE_BIT,[IY+D_COM_STAT]       ; IO DONE ?
FA02 28FA            8553         JR      Z,LOOP4                                 ; ***** NEED PROTECTION HERE
                     8554
FA04 FD7E00          8555         LD      A,[IY+D_COM_STAT]       ; CHECK IF VALID RESPONSE
FA07 FE80            8556         CP      CMND_FIN_STATUS         ;
FA09 2808            8557         JR      Z,RECVD_VALID_STAT      ; YES INDEED ...
                     8558
FA0B F1              8559         POP     AF                      ; TRY NEXT DEVICE ADDR
                     8560
FA0C 3C              8561         INC     A                       ;
FA0D FE10            8562         CP      MAX_DEV_ADDR+1          ; PAST LAST VALID ADDR ?
FA0F 20E5            8563         JR      NZ,LOOP3                ; NOPE - NEW ADDR
                     8564
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    FA11 DD3503      8565         DEC     [IX+P_NUM_DCBS]         ; DON'T NEED LAST DCB WE USED
                     8566
    FA14 1811        8567         JR      END_S_A                 ; ALL DONE
                     8568
    FA16             8569 RECVD_VALID_STAT:
    FA16 DD3403      8570         INC     [IX+P_NUM_DCBS]         ;
                     8571
    FA19 110015      8572         LD      DE,D_SIZE               ; ADVANCE TO NEXT DCB
    FA1C FD19        8573         ADD     IY,DE                   ;
                     8574
    FA1E F1          8575         POP     AF                      ; ADVANCE TO NEXT NET ADDR
                     8576
    FA1F 3C          8577         INC     A                       ;
    FA20 FE10        8578         CP      MAX_DEV_ADDR+1          ; NO MORE ?
    FA22 2002        8579         JR      NZ,LOOP3                ; MORE TO CHECK
                     8580
    FA24 DD3503      8581         DEC     [IX+P_NUM_DCBS]         ; DON'T NEED LAST DCB WE USED
                     8582
    FA27             8583 END_S_A:
    FA27 DDE1        8584         POP     IX                      ; RESTORE IX
    FA29 FDE1        8585         POP     IY                      ;         IY
    FA2B E1          8586         POP     HL                      ;         HL
    FA2C D1          8587         POP     DE                      ;         DE
    FA2D C1          8588         POP     BC                      ;         BC
                     8589
    FA2E C9          8590         RET
                     8591
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8593
                     8594 ; ENTRY:        __RELOC_PCB
                     8595 ; NEEDS:        [CURRENT_PCB] SET TO CURRENTLY OPERATING PCB
                     8596 ;               HL = ADDRESS OF WHERE PCB SHOULD BE RELOCATED TO
                     8597 ; RETURNS:      PCB RELOCATED AS REQUESTED
                     8598 ;               [CURRENT_PCB] UPDATED TO REFLECT NEW ADDRESS
                     8599
                     8600       ;GLB      __RELOC_PCB
                     8601
FA2F                 8602 __RELOC_PCB
FA2F FDE5            8603       PUSH      IY
                     8604
FA31 FD2AFD70        8605       LD        IY,[CURRENT_PCB]          ; IY = ADDRESS OF CURRENT PCB
                     8606
FA35 FD7501          8607       LD        [IY+P_REL_ADDR_LO],L      ; PLACE ADDR OF NEW PCB IN PLACE
FA38 FD7402          8608       LD        [IY+P_REL_ADDR_HI],H      ;
                     8609
FA3B FD360003        8610       LD        [IY+P_COM_STAT],PCB_SNA   ; TELL MASTER TO SET NEW PCB ADDRESS
                     8611
FA3F                 8612 LOOP5:
FA3F FD7E00          8613       LD        A,[IY+P_COM_STAT]         ; CHECK FOR OPERATION COMPLETE
FA42 FE83            8614       CP        PCB_SNA_ACK               ; ***** NEED PROTECTION HERE
FA44 20F9            8615       JR        NZ,LOOP5                  ; NOPE
                     8616
FA46 22FD70          8617       LD        [CURRENT_PCB],HL          ; REMEMBER WHERE PCB IS (PLEASE...)
                     8618
FA49 FDE1            8619       POP       IY
                     8620
FA4B C9              8621       RET
                     8622
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8624
                     8625 ; ENTRY:        __GET_PCB_ADDR
                     8626
                     8627         ;GLB    __GET_PCB_ADDR
                     8628
    FA4C             8629 __GET_PCB_ADDR:
    FA4C FD2AFD70    8630         LD      IY,[CURRENT_PCB]        ; ADDRESS OF PCB
                     8631
    FA50 C9          8632         RET
                     8633
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8635
                     8636         ;GLB     __SOFT_RES_KBD,__SOFT_RES_PR,__SOFT_RES_TAPE
                     8637
                     8638         ;EXT     __FIND_DCB,CHK_IF_INACTIVE
                     8639
FA51                 8640 __SOFT_RES_KBD:
FA51 3E01            8641         LD       A,KEYBOARD_ID          ; RESET KBD
FA53 1808            8642         JR       __SOFT_RES_DEV
                     8643
FA55                 8644 __SOFT_RES_PR:
FA55 3E02            8645         LD       A,PRINTER_ID           ; RESET PRINTER
FA57 1804            8646         JR       __SOFT_RES_DEV         ;
                     8647
FA59                 8648 __SOFT_RES_TAPE:
FA59 3E08            8649         LD       A,TAPE_ID              ; RESET TAPE
FA5B 1800            8650         JR       __SOFT_RES_DEV         ;
                     8651
                     8652         ;GLB     __SOFT_RES_DEV
                     8653
FA5D                 8654 __SOFT_RES_DEV:
FA5D FDE5            8655         PUSH     IY                     ; SAVE IY
                     8656
FA5F CDF446          8657         CALL     __FIND_DCB             ; FIND APPROPRIATE DEVICE
FA62 2014            8658         JR       NZ,NO_DCB_FOUND        ; BAD DEVICE I.D.
                     8659
FA64 CDF496          8660         CALL     CHK_IF_INACTIVE        ; IS DEVICE IDLE?
FA67 200F            8661         JR       NZ,ACTIVE_DCB          ; NOPE ... OOPS ...
                     8662
FA69 FD360002        8663         LD       [IY+D_COM_STAT],DCB_RESET      ; SOFT RESET DEVICE
                     8664
FA6D                 8665 DCB_CMND_LOOP:
FA6D FDCB007E        8666         BIT      CMND_COMPLETE_BIT,[IY+D_COM_STAT] ; WAS COMMAND COMPLETE?
FA71 28FA            8667         JR       Z,DCB_CMND_LOOP                   ; NOPE, CHECK AGAIN
                     8668
FA73 FD7E00          8669         LD       A,[IY+D_COM_STAT]      ; YUP
FA76 FE80            8670         CP       CMND_FIN_STATUS        ;
                     8671
FA78                 8672 ACTIVE_DCB:
FA78                 8673 NO_DCB_FOUND:
FA78 FDE1            8674         POP      IY                     ;
                     8675
FA7A C9              8676         RET
                     8677
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8679
                     8680 ; READ DCB RETURN CODE (FIRST BYTE OF DCB)
                     8681
                     8682       ;GLB     __RD_KBD_RET_CODE,__RD_PR_RET_CODE,__RD_TAPE_RET_CODE
                     8683
FA7B                 8684 __RD_KBD_RET_CODE:
FA7B 3E01            8685       LD       A,KEYBOARD_ID          ; RESET KBD
FA7D 1808            8686       JR       __RD_RET_CODE
                     8687
FA7F                 8688 __RD_PR_RET_CODE:
FA7F 3E02            8689       LD       A,PRINTER_ID           ; RESET PRINTER
FA81 1804            8690       JR       __RD_RET_CODE          ;
                     8691
FA83                 8692 __RD_TAPE_RET_CODE:
FA83 3E08            8693       LD       A,TAPE_ID              ; RESET TAPE
FA85 1800            8694       JR       __RD_RET_CODE          ;
                     8695
                     8696       ;GLB     __RD_RET_CODE
                     8697
FA87                 8698 __RD_RET_CODE:
FA87 FDE5            8699       PUSH     IY                     ; SAVE IY
                     8700
FA89 CDF446          8701       CALL     __FIND_DCB             ; FIND APPROPRIATE DEVICE
FA8C 2003            8702       JR       NZ,NO_DCB_FOUND2       ; BAD DEVICE I.D.
                     8703
                     8704                                       ; Z CONDITION CODE PRESENT
                     8705
FA8E FD7E00          8706       LD       A,[IY+D_COM_STAT]      ; COMMAND/STATUS BYTE FROM DCB
                     8707
FA91                 8708 NO_DCB_FOUND2:
FA91 FDE1            8709       POP      IY
                     8710
FA93 C9              8711       RET
                     8712
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8714 ;WP              EQU 02000H          ; WHERE TO GO TO START WP
               <0100>8715 WP               EQU 00100H          ; WHERE TO GO TO START WP
                     8716                  ;GLB __GOTO_WP
                     8717                  ;EXT MEM_CNFGOO
                     8718                  ;EXT SWITCH_MEM
     FA94            8719 __GOTO_WP:
     FA94 3AFC17     8720        LD        A,[MEM_CNFGOO]      ;get configuration byte for 32K ROM and 32 RAM
     FA97 CDFD14     8721        CALL      SWITCH_MEM
                     8722
                     8723 ;WP_CODE_START   EQU 2000H
                     8724 ;WP_SIZE         EQU 3*8*1024
                     8725 ;WP_DEST         EQU 8000H
                     8726 ;       LD        A,0                 ; BANK SWITCH IN THE BOOT ROMS
                     8727 ;       OUT       [7FH],A             ;  (I HOPE)
                     8728 ;
                     8729 ;       LD        HL,WP_CODE_START    ; WHERE DOES WP SIT IN BOOT ROMS
                     8730 ;       LD        DE,WP_DEST          ; WHERE WP GOES
                     8731 ;       LD        BC,WP_SIZE          ; HOW BIG WP IS
                     8732 ;       LDIR                          ; MOVE WP INTO PLACE
                     8733 ;
                     8734 ;       LD        A,3                 ; BANK SWITCH IN OS_7
                     8735 ;       OUT       [7FH],A             ;  (I KNOW)
                     8736
     FA9A C30100     8737        JP        WP                  ; GOTO WP
                     8738
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    8740        ;GLB      __READ_EOS
                    8741
    FA9D            8742 __READ_EOS:
    FA9D C9         8743        RET
                    8744 ;
                    8745 ;      CODE FROM HERE TO END OF EOS_UTIL WAS INSERTED AFTER THE
                    8746 ;      REV. 06 ROM WAS BURNED.  IT HAS BEEN COMMENTED OUT TO MAINTAIN
                    8747 ;      COMPATIBILITY WITH THAT ROM
                    8748 ;
                    8749 ;       END
                    8750 ;
                    8751 ;
                    8752 ;EOS_CODE_START  EQU 00800H
                    8753 ;EOS_SIZE        EQU 02000H-800H
                    8754 ;EOS_DEST        EQU 0E000H+800H
                    8755 ;
                    8756 ;       LD       A,0                        ; BANK SWITCH IN THE BOOT ROMS
                    8757 ;       OUT      [7FH],A                    ;  (I HOPE)
                    8758 ;
                    8759 ;       LD       HL,EOS_CODE_START           ; WHERE DOES EOS SIT IN BOOT ROMS
                    8760 ;       LD       DE,EOS_DEST                 ; WHERE EOS GOES
                    8761 ;       LD       BC,EOS_SIZE                 ; HOW BIG EOS IS
                    8762 ;       LDIR                                ; MOVE EOS INTO PLACE
                    8763 ;
                    8764 ;       LD       A,3                        ; BANK SWITCH IN OS_7
                    8765 ;       OUT      [7FH],A                    ;  (I KNOW)
                    8766 ;
                    8767 ;       RET                                 ; EOS IS IN
                    8768 ;;
                    8769 ;;*****************************************************************
                    8770 ;;*****************************************************************
                    8771 ;;*****************************************************************
                    8772 ;;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8774 ; REV 0          8/24/83  V/D
                     8775 ********************************************************************************
                     8776 *
                     8777 *    __RD_1_BLOCK WILL READ ONE BLOCK OF DATA FROM A BLOCK DEVICE
                     8778 *    TO A LOCATION SPECIFIED BY REGISTER HL
                     8779 *
                     8780 *       INPUT:  A       DEVICE ID
                     8781 *                       LOW NIBBLE - DEVICE ADDRESS
                     8782 *                       HI NIBBLE  - SECONDARY DEVICE ID
                     8783 *
                     8784 *               HL      DESTINATION ADDRESS IN RAM
                     8785 *
                     8786 *               REGISTER PAIR.
                     8787 *                       BC DE       - SECTOR NUMBER ON DEVICE
                     8788 *
                     8789 *       THE TRANSFER BYTE COUNT IS DEFINED BY "DCB_MAX_MESS_LEN" WHICH
                     8790 *       IS FOUND IN THE DEVICE DCB.
                     8791 *
                     8792 *       OUTPUT:
                     8793 *               CONDITION BITS:
                     8794 *                       Z: NO ERRORS
                     8795 *                      NZ: ERRORS OCCURED
                     8796 *                               A - ERROR CODE
                     8797 *
                     8798 *       ALL REGISTERS ARE PRESERVED
                     8799 *
                     8800 ********************************************************************************
                     8801
                     8802        ;GLB     __RD_1_BLOCK
                     8803        ;EXT     __FIND_DCB
                     8804        ;EXT     CHK_IF_INACTIVE
                     8805
FA9E                 8806 __RD_1_BLOCK:
FA9E FDE5            8807        PUSH     IY
                     8808
FAA0 F5              8809        PUSH     AF                          ; SAVE DEVICE ID
                     8810
FAA1 CDFAC6          8811        CALL     __START_RD_1_BLOCK
                     8812
FAA4 2007            8813        JR       NZ,ERROR_WITH_DCB
                     8814
FAA6                 8815 RD_NOT_COMPLETED:
FAA6 F1              8816        POP      AF                 ; GET DEVICE ID
FAA7 F5              8817        PUSH     AF                 ; SAVE IT AGAIN
                     8818
FAA8 CDFAE2          8819        CALL     __END_RD_1_BLOCK           ; LOOK FOR STATUS
                     8820
FAAB 30F9            8821        JR       NC,RD_NOT_COMPLETED        ; COMMAND NOT COMPLETED, CHECK STATUS AGAIN
                     8822
FAAD                 8823 ERROR_WITH_DCB:
FAAD FDE1            8824        POP      IY                 ; REALLY IS AF BUT NEED TO SAVE IF AN ERROR
                     8825
FAAF FDE1            8826        POP      IY
                     8827
FAB1 C9              8828        RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8830 ************************************************************************
                     8831 *
                     8832 *       __WR_1_BLOCK WILL WRITE ONE BLOCK OF DATA TO A BLOCK DRIVER
                     8833 *       FORM A LOCATION SPECIFIED BY REGISTER HL
                     8834 *
                     8835 *       INPUT:
                     8836 *               A       DEVICE ID
                     8837 *                       LOW NIBBLE - DEVICE ADDRESS
                     8838 *                       HI NIBBLE  - SECONDARY DEVICE ID
                     8839 *
                     8840 *               HL      SOURCE ADDRESS IN RAM
                     8841 *
                     8842 *               REGISTER PAIR:
                     8843 *                       BC DE  -  SECTOR NUMBER ON DEVICE
                     8844 *
                     8845 *       THE TRANSFER BYTE COUNT IS DEFINED BY "DCB_MAX_MESS_LEN"
                     8846 *       WHICH IS FOUND IN THE DEVICE DCB
                     8847 *
                     8848 *       OUTPUT:
                     8849 *               CONDITION BITS:
                     8850 *                       Z:  NO ERRORS
                     8851 *                      NZ:  ERRORS OCCURED
                     8852 *                               A - ERROR CODE
                     8853 *
                     8854 *       ALL REGISTERS  ARE PRESERVED
                     8855 *
                     8856 ************************************************************************
                     8857
                     8858        ;GLB    __WR_1_BLOCK
                     8859
FAB2                 8860 __WR_1_BLOCK:
FAB2 FDE5            8861        PUSH    IY
                     8862
FAB4 F5              8863        PUSH    AF                      ; SAVE DEVICE ID
                     8864
FAB5 CDFAFF          8865        CALL    __START_WR_1_BLOCK
                     8866
FAB8 2007            8867        JR      NZ,ERROR_WITH_DCB2      ; DCB NOT FOUND OR NOT IDLE
                     8868
FABA                 8869 WR_NOT_COMPLETED:
FABA F1              8870        POP     AF                      ; GET DEVICE ID
                     8871
FABB F5              8872        PUSH    AF
                     8873
FABC CDFB1B          8874        CALL    __END_WR_1_BLOCK
                     8875
FABF 30F9            8876        JR      NC,WR_NOT_COMPLETED     ; COMMAND NOT COMPLETED,CHECK STATUS AGAIN
                     8877
FAC1                 8878 ERROR_WITH_DCB2:
FAC1 FDE1            8879        POP     IY              ; IS REALLY AF BUT NEED TO PRESERVE A REG IN CASE OF ERRORS
                     8880
FAC3 FDE1            8881        POP     IY
                     8882
FAC5 C9              8883        RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8885 ********************************************************************************
                     8886 *
                     8887 *       __START_RD_1_BLOCK WILL SEND A READ COMMAND TO READ A BLOCK
                     8888 *       OF DATA FROM A BLOCK DEVICE TO A BUFFER SPECIFIED BY REGISTER HL
                     8889 *
                     8890 *       INPUT:  A       DEVICE ID
                     8891 *                       LOW NIBBLE - DEVICE ADDRESS
                     8892 *                       HI NIBBLE  - SECONDARY DEVICE ID
                     8893 *
                     8894 *               HL      DESTINATION ADDRESS IN RAM
                     8895 *
                     8896 *               REGISTER PAIR:
                     8897 *                       BC DE       - SECTOR NUMBER ON DEVICE
                     8898 *
                     8899 *       THE TRANSFER BYTE COUNT IS DEFINED BY "DCB_MAX_MESS_LEN" WHICH
                     8900 *       IS FOUND IN THE DEVICE DCB.
                     8901 *
                     8902 *       OUTPUT:
                     8903 *               CONDITION BITS:
                     8904 *                       Z: NO ERRORS
                     8905 *                      NZ: ERRORS OCCURED
                     8906 *                               A - ERROR CODE
                     8907 *
                     8908 *       ALL REGISTERS ARE PRESERVED
                     8909 *
                     8910 ********************************************************************************
                     8911
                     8912 ;
                     8913 ;NO_DCB_FOUND1 CHANGED TO BLK_NO_CB_FOUND1 TO AVOID CONFLICT
                     8914 ;WITH IDENTICAL LABEL ELSEWHERE
                     8915         ;GLB     __START_RD_1_BLOCK
                     8916
FAC6                 8917 __START_RD_1_BLOCK:
FAC6 C5              8918         PUSH    BC
FAC7 D5              8919         PUSH    DE
FAC8 E5              8920         PUSH    HL
FAC9 FDE5            8921         PUSH    IY
                     8922
FACB CDF446          8923         CALL    __FIND_DCB
                     8924
FACE 200C            8925         JR      NZ,B_NO_DCB_FOUND1     ; COULD NOT FIND DCB FOR THIS DEVICE
                     8926
FAD0 CDF496          8927         CALL    CHK_IF_INACTIVE          ; IS DEVICE IDLE?
FAD3 2007            8928         JR      NZ,ACTIVE_DCB1            ; NOPE, GET OUT
                     8929
FAD5 CDFB38          8930         CALL    PREP_DCB                 ; SET UP THE DCB FOR READ
                     8931
FAD8 FD360004        8932         LD      [IY+D_COM_STAT],DCB_RD   ; SEND READ COMMAND
                     8933
FADC                 8934 ACTIVE_DCB1:
FADC                 8935 B_NO_DCB_FOUND1:
FADC FDE1            8936         POP     IY
FADE E1              8937         POP     HL
FADF D1              8938         POP     DE
FAE0 C1              8939         POP     BC
                     8940
FAE1 C9              8941         RET
```

LOCATION OBJECT CODE LINE SOURCE LINE

8942

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     8944 ********************************************************************************
                     8945 *
                     8946 *       __END_RD_1_BLOCK CHECKS THE STATUS AFTER A __START_RD_1_BLOCK
                     8947 *       IS EXECUTED
                     8948 *
                     8949 *       INPUT:  A       DEVICE ID
                     8950 *                       LOW NIBBLE - DEVICE ADDRESS
                     8951 *                       HI NIBBLE  - SECONDARY DEVICE ID
                     8952 *
                     8953 *       OUTPUT: CONDITION FLAGS
                     8954 *               C:  OPERATION COMPLETED
                     8955 *              NC:  OPERATION NOT COMPLETED
                     8956 *               Z:  NO ERRORS
                     8957 *              NZ:  ERRORS
                     8958 *                       A = ERROR CODE
                     8959 *
                     8960 *       ALL REGISTERS ARE SAVED
                     8961 *
                     8962 ********************************************************************************
                     8963
                     8964         ;GLB      __END_RD_1_BLOC1
                     8965         ;EXT      CHK_IF_IDLE
                     8966
FAE2                 8967 __END_RD_1_BLOCK:
FAE2 FDE5            8968         PUSH     IY
                     8969
FAE4 CDF446          8970         CALL     __FIND_DCB
                     8971
FAE7 37              8972         SCF                                 ; SET CARRY FLAG TO INDICATE OPERATION COMPLETED
                     8973
FAE8 2012            8974         JR       NZ,NO_DCB_FOUND5           ; NO DCB FOUND
                     8975
FAEA CDF4AE          8976         CALL     CHK_IF_IDLE                ; IS DEVICE IDLE?
                     8977
FAED 200D            8978         JR       NZ,DCB_IS_IDLE5
                     8979
FAEF B7              8980         OR       A                          ; CLEAR CARRY FLAG
FAF0 FDCB007E        8981         BIT      CMND_COMPLETE_BIT,[IY+D_COM_STAT] ; WAS COMMAND COMPLETE?
FAF4 2806            8982         JR       Z,COMMAND_NOT_COMPLETED    ; NOPE,
                     8983
                     8984
FAF6 FD7E00          8985         LD       A,[IY+D_COM_STAT]          ; GET STATUS OF LAST OPERATION
                     8986
FAF9 FE80            8987         CP       CMND_FIN_STATUS            ; ANY ERRORS?
FAFB 37              8988         SCF                                 ; INDICATE COMMAND COMPLETED
                     8989                                             ; CONDITON FLAGS: IF NO ERRORS:  C,Z
                     8990                                             ;                 IF ERRORS:  C,NZ  A=ERROR CODE
                     8991
FAFC                 8992 DCB_IS_IDLE5:
FAFC                 8993 NO_DCB_FOUND5:
FAFC                 8994 COMMAND_COMPLETED:
FAFC                 8995 COMMAND_NOT_COMPLETED:
FAFC FDE1            8996         POP      IY
                     8997
FAFE C9              8998         RET
                     8999
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     9001 ********************************************************************
                     9002 *
                     9003 *       __START_WR_1_BLOCK WILL SEND A WRITE COMMAND TO WRITE A BLOCK
                     9004 *       OF DATA TO A BLOCK DEVICE FROM A LOCATION SPECIFIED BY REGISTER HL
                     9005 *
                     9006 *       INPUT:
                     9007 *               A       DEVICE ID
                     9008 *                       LOW NIBBLE - DEVICE ADDRESS
                     9009 *                       HI NIBBLE  - SECONDARY DEVICE ID
                     9010 *
                     9011 *               HL      SOURCE ADDRESS IN RAM
                     9012 *
                     9013 *               REGISTER PAIR:
                     9014 *                       BC DE  -  SECTOR NUMBER ON DEVICE
                     9015 *
                     9016 *
                     9017 *       THE TRANSFER BYTE COUNT IS DEFINED BY "DCB_MAX_MESS_LEN"
                     9018 *       WHICH IS FOUND IN THE DEVICE DCB
                     9019 *
                     9020 *       OUTPUT:
                     9021 *               CONDITION BITS:
                     9022 *                       Z:  NO ERRORS
                     9023 *                      NZ:  ERRORS OCCURED
                     9024 *                               A - ERROR CODE
                     9025 *
                     9026 *       ALL REGISTERS ARE PRESERVED
                     9027 *
                     9028 ********************************************************************
                     9029
                     9030 ;NO_DCB_FOUND3 CHANGED TO B_NO_DCB_FOUND3 TO AVOID CONFLICT WITH
                     9031 ;IDENTICAL LABEL ELSEWHERE
                     9032
                     9033       ;GLB     __START_WR_1_BLOCK
                     9034
     FAFF            9035 __START_WR_1_BLOCK:
     FAFF C5         9036        PUSH    BC
     FB00 D5         9037        PUSH    DE
     FB01 E5         9038        PUSH    HL
     FB02 FDE5       9039        PUSH    IY
                     9040
     FB04 CDF446     9041        CALL    __FIND_DCB
                     9042
     FB07 200C       9043        JR      NZ,B_NO_DCB_FOUND3     ; NO DCB FOUND
                     9044
     FB09 CDF496     9045        CALL    CHK_IF_INACTIVE          ; IS DEVICE ACTIVE?
     FB0C 2007       9046        JR      NZ,ACTIVE_DCB3           ; NOPE, GET OUT
                     9047
     FB0E CDFB38     9048        CALL    PREP_DCB                 ; SET UP DCB FOR WRITE
                     9049
     FB11 FD360003   9050        LD      [IY+D_COM_STAT],DCB_WR   ; SEND WRITE COMMAND
                     9051
     FB15            9052 B_NO_DCB_FOUND3:
     FB15            9053 ACTIVE_DCB3:
     FB15 FDE1       9054        POP     IY
     FB17 E1         9055        POP     HL
     FB18 D1         9056        POP     DE
     FB19 C1         9057        POP     BC
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     9058
    FB1A C9          9059         RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     9061
                     9062 ********************************************************************************
                     9063 *
                     9064 *       __END_WR_1_BLOCK WILL CHECK THE STATUS OF THE LAST OPERATION
                     9065 *       EXECUTED
                     9066 *
                     9067 *       INPUT:
                     9068 *               A       DEVICE ID
                     9069 *                       LOW NIBBLE - DEVICE ADDRESS
                     9070 *                       HI NIBBLE  _ SECONDARY DEVICE ID
                     9071 *
                     9072 *       OUTPUT:
                     9073 *               CONDITON FLAGS
                     9074 *                       C       OPERATION COMPLETED
                     9075 *                      NC       OPERATION NOT COMPLETED
                     9076 *                       Z       NO ERRORS
                     9077 *                      NZ       ERRORS
                     9078 *                               A = ERROR CODE
                     9079 *
                     9080 *       ALL REGISTERS ARE PRESERVED
                     9081 *
                     9082 ********************************************************************************
                     9083
                     9084 ;NO_DCB_FOUND4           B_NO_DCB_FOUND4
                     9085 ;DCB_IS_IDLE4 CHANGED TO BLK_DCB_IS_IDLE4 TO AVOID CONFLICT WITH
                     9086 ;IDENTICAL LABEL ELSEWHERE
                     9087
                     9088        ;GLB     __END_WR_1_BLOCK
                     9089
FB1B                 9090 __END_WR_1_BLOCK:
FB1B FDE5            9091        PUSH     IY
                     9092
FB1D CDF446          9093        CALL     __FIND_DCB
                     9094
FB20 37              9095        SCF
FB21 2012            9096        JR       NZ,B_NO_DCB_FOUND4     ; CONDITION FLAGS: C,NZ  A=ERROR CODE
                     9097
FB23 CDF4AE          9098        CALL     CHK_IF_IDLE            ; IS DEVICE IDLE?
FB26 200D            9099        JR       NZ,BLK_DCB_IS_IDLE4    ; YUP
                     9100
FB28 B7              9101        OR       A                      ; RESET CARRY FLAG
                     9102
FB29 FDCB007E        9103        BIT      CMND_COMPLETE_BIT,[IY+D_COM_STAT] ; WAS COMMAND COMPLETE?
FB2D 2806            9104        JR       Z,CMND_NOT_COMPLETED   ; NOPE.
                     9105
FB2F FD7E00          9106        LD       A,[IY+D_COM_STAT]      ; YUP
                     9107
                     9108
FB32 FE80            9109        CP       CMND_FIN_STATUS        ; ANY ERRORS?
FB34 37              9110        SCF                             ; INDICATE COMMAND COMPLETED
                     9111                                        ; CONDITION FLAGS: IF NO ERRORS: C,Z
                     9112                                        ;                  IF ERRORS: C,NZ, A=ERROR CODE
                     9113
                     9114
FB35                 9115 BLK_DCB_IS_IDLE4:
FB35                 9116 B_NO_DCB_FOUND4:
FB35                 9117 CMND_COMPLETED:
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    FB35             9118 CMND_NOT_COMPLETED:
    FB35 FDE1        9119         POP     IY
    FB37 C9          9120         RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     9122 ********************************************************************************
                     9123 *
                     9124 *       PREP_DCB LOADS THE DCB BUFFER WITH THE PROPER VALUES FOR
                     9125 *       A READ OR WRITE COMMAND
                     9126 *
                     9127 *       INPUT:
                     9128 *               IY      START OF SPECIFIC DCB
                     9129 *               HL      ADDRESS OF RAM DESTINATION(READ)/SOURCE(WRITE)
                     9130 *
                     9131 *               BC DE   DEVICE SECTOR NUMBER
                     9132 *
                     9133 *       OUTPUT:    NONE
                     9134 *
                     9135 ********************************************************************************
                     9136
FB38                 9137 PREP_DCB:
FB38 F5              9138         PUSH    AF
                     9139
FB39 CB3F            9140         SRL     A                              ; EXTRACT SECONDARY DEVICE ID
FB3B CB3F            9141         SRL     A
FB3D CB3F            9142         SRL     A
FB3F CB3F            9143         SRL     A
                     9144
FB41 FD7709          9145         LD      [IY+D_SEC_DEV_ID],A              ; ... AND PLACE IN DCB
FB44 FD7E11          9146         LD      A,[IY+D_MAX_MSG_LN_LO]         ; GET BLOCK LENGTH AND PLACE ....
FB47 FD7703          9147         LD      [IY+D_BUF_LEN_LO],A             ; .... IT IN BUFFER LENGTH ....
FB4A FD7E12          9148         LD      A,[IY+D_MAX_MSG_LN_HI]         ; .... FOR READ
FB4D FD7704          9149         LD      [IY+D_BUF_LEN_HI],A
                     9150
FB50 FD7305          9151         LD      [IY+D_SECT_NUM],E                ; LOAD UP A 4 BYTE SECTOR....
FB53 FD7206          9152         LD      [IY+D_SECT_NUM+1],D              ; .... NUMBER FROM REGISTERS....
FB56 FD7107          9153         LD      [IY+D_SECT_NUM+2],C              ; .... PAIR BC DE
FB59 FD7008          9154         LD      [IY+D_SECT_NUM+3],B
                     9155
FB5C FD7501          9156         LD      [IY+D_BUF_ADR_LO],L            ; ADDRESS OF SOURCE/DESTIATION
FB5F FD7402          9157         LD      [IY+D_BUF_ADR_HI],H            ; .... IN RAM
                     9158
FB62 F1              9159         POP     AF
FB63 C9              9160         RET
                     9161 ;
                     9162 ;***************************************************************
                     9163 ;***************************************************************
                     9164 ;***************************************************************
                     9165 ;
                     9166
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     9168
                     9169 ;REV 0          (V/D  8-24-83)
                     9170
                     9171       ;EXT     DEVICE_ID
                     9172
                     9173 ********************************************************************************
                     9174 * THIS ROUTINE WILL INITIATE A READ COMMAND ON A CHARACTER DEVICE.  THE
                     9175 * ROUTINE WILL WAIT FOR THE COMMAND TO BE COMPLETED.  IF THE COMMAND
                     9176 * COMPLETED WITH NO ERRORS, THEN THE FLAG WILL BE ZERO.  IF THERE
                     9177 * WERE ERRORS THEN THE FLAG WILL BE NON-ZERO.
                     9178 *      INPUT:  A ==>   DEVICE ID
                     9179 *              DE =>   DESTINATION ADDRESS
                     9180 *              BC =>   NUMBER OF BYTES TO READ
                     9181 *      OUTPUT:
                     9182 *              CONDITION FLAGS
                     9183 *                      Z :     NO ERROR
                     9184 *                              A ===> KEY READ
                     9185 *                      NZ:     ERROR OCCURED
                     9186 *                              A ===> ERROR CODE
                     9187 ********************************************************************************
                     9188
                     9189       GLOBAL   __RD_CH_DEV
                     9190       ;EXT     __FIND_DCB
                     9191
FB64                 9192 __RD_CH_DEV:
FB64 32FD72          9193       LD       [DEVICE_ID],A          ; SAVE DEVICE ID
                     9194
FB67 CDFB86          9195       CALL     __START_RD_CH_DEV
FB6A 2008            9196       JR       NZ,GOOD_BYE
                     9197
FB6C                 9198 RD_DEV_LOOP:
FB6C 3AFD72          9199       LD       A,[DEVICE_ID]                  ; RESTOR THE DEVICE ID
FB6F CDFBA5          9200       CALL     __END_RD_CH_DEV
FB72 30F8            9201       JR       NC,RD_DEV_LOOP
                     9202
FB74                 9203 GOOD_BYE:
FB74 C9              9204       RET
                     9205
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     9207 ********************************************************************************
                     9208 * WRITE CHAR DEV ALLOWS THE CALLER TO SEND A PRINT COMMAND BY SPECIFYING
                     9209 * THE BUFFER OF THE STRING AND THE NUMBER OF BYTES TO PRINT.
                     9210 *       INPUT:  A ==>   DEVICE ID
                     9211 *               BC =>   NUMBER OF BYTES TO WRITE
                     9212 *               HL =>   SOURCE OF DATA
                     9213 *       OUTPUT:
                     9214 *               CONDITION FLAGS
                     9215 *                       Z :     NO ERRORS
                     9216 *                       NZ:     ERROR OCCURED
                     9217 *                               A ===> ERROR CODE
                     9218 ********************************************************************************
                     9219
                     9220        GLOBAL  __WR_CH_DEV
                     9221
FB75                 9222 __WR_CH_DEV:
FB75 32FD72          9223        LD      [DEVICE_ID],A
                     9224
FB78 CDFBC2          9225        CALL    __START_WR_CH_DEV
FB7B 2008            9226        JR      NZ,SO_LONG
                     9227
FB7D                 9228 DCB_WRITE_LOOP:
FB7D 3AFD72          9229        LD      A,[DEVICE_ID]           ; RESTORE THE ID FOR USE IN THE CALL
FB80 CDFBE1          9230        CALL    __END_WR_CH_DEV
FB83 30F8            9231        JR      NC,DCB_WRITE_LOOP
                     9232
FB85                 9233 SO_LONG:
FB85 C9              9234        RET
                     9235
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     9237
                     9238 ********************************************************************************
                     9239 * THIS ROUTINE IS RESPONSIBLE FOR SETTING UP A DEVICE DCB TO DO A READ.
                     9240 * THE LAST THING DONE IN THIS ROUTINE IS TO ISSUE THE COMMAND TO READ
                     9241 * A KEY FROM THE KEYBOARD.
                     9242 *       INPUT:  A  ===> CONTAINS THE DEVICE ID
                     9243 *               DE ===> DESTINATION ADDRESS
                     9244 *               BC ===> NUMBER OF BYTES TO READ
                     9245 *       OUTPUT:  CONDITION FLAGS
                     9246 *               Z:      THE READ DATA FROM KEYBOARD COMMAND HAS BEEN SENT
                     9247 *               NZ:     ERROR OCCURED
                     9248 *                       A ===>  ERROR CODE
                     9249 ********************************************************************************
                     9250
                     9251        ;GLB     __START_RD_CH_DEV
                     9252        ;EXT     CHK_IF_INACTIVE
                     9253
FB86                 9254 __START_RD_CH_DEV:
FB86 FDE5            9255        PUSH     IY
FB88 CDF446          9256        CALL     __FIND_DCB              ; FIND THE DCB
                     9257
FB8B 2015            9258        JR       NZ,NO_DCB_DEFINED
                     9259
FB8D CDF496          9260        CALL     CHK_IF_INACTIVE         ; SEE IF THE KEY BOARD IS AVAILABLE
                     9261
FB90 2010            9262        JR       NZ,DCB_IS_ACTIVE        ; DCB IS CURRENTLY ACTIVE
                     9263
FB92 FD7301          9264        LD       [IY+D_BUF_ADR_LO],E ; LOAD THE DESTINATION ADDRESS INTO DCB
FB95 FD7202          9265        LD       [IY+D_BUF_ADR_HI],D
                     9266
FB98 FD7103          9267        LD       [IY+D_BUF_LEN_LO],C      ; LOAD THE BYTE COUNT
FB9B FD7004          9268        LD       [IY+D_BUF_LEN_HI],B
                     9269
FB9E FD360004        9270        LD       [IY+D_COM_STAT],DCB_RD        ; INITIATE THE COMMAND TO READ
                     9271                                           ; ZERO FLAG WAS SET ABOVE
                     9272
FBA2                 9273 DCB_IS_ACTIVE:                          ; NZ IS ALREADY THE CONDITION OF THE FLAG
FBA2                 9274 NO_DCB_DEFINED:                         ; A CONTAINS THE ERROR CODE
FBA2 FDE1            9275        POP      IY
                     9276
FBA4 C9              9277        RET
                     9278
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     9280 ********************************************************************************
                     9281 * THIS ROUTINE WILL CHECK THE DEVICES DCB COM/STATUS BYTE AND RETURN
                     9282 * THE RESULT OF THE CHECK.
                     9283 *       INPUT:  A  ===> DEVICE ID
                     9284 *       OUTPUT: CONDITION FLAGS
                     9285 *               C :     COMMAND HAS FINISHED
                     9286 *               NC:     COMMAND HAS NOT BEEN PROCESSED
                     9287 *               Z :     NO ERROR OCCURED
                     9288 *               NZ:     ERROR OCCURED
                     9289 *                       A ===>  ERROR CODE
                     9290
                     9291 ********************************************************************************
                     9292
                     9293        ;GLB     __END_RD_CH_DEV
                     9294        ;EXT     CHK_IF_IDLE
                     9295
FBA5                 9296 __END_RD_CH_DEV:
FBA5 FDE5            9297        PUSH     IY
FBA7 CDF446          9298        CALL     __FIND_DCB
FBAA 37              9299        SCF
                     9300
FBAB 2012            9301        JR       NZ,NO_DCB                ; THERE WAS NO KEYBOARD DCB
                     9302
FBAD CDF4AE          9303        CALL     CHK_IF_IDLE              ; IS DEVICE IDL?
FBB0 2000            9304        JR       NZ,DCB_IS_IDL            ; YUP
                     9305
FBB2 B7              9306        OR       A                        ; CLEAR THE CARRY FLAG
FBB3 FDCB007E        9307        BIT      CMND_COMPLETE_BIT,[IY+D_COM_STAT]        ; WAS THE COMMAND PROCESSED?
FBB7 2806            9308        JR       Z,ROUTINE_FINISHED       ; NO, RETURN WITH THE CARRY FLAG CLEARED
                     9309                                         ; YES, THE COMMAND WAS PROCESSED
FBB9 FD7E00          9310        LD       A,[IY+D_COM_STAT]        ; DID ANY ERRORS OCCUR?
FBBC FE80            9311        CP       CMND_FIN_STATUS          ;
FBBE 37              9312        SCF
                     9313                                         ; NO ERROR, SET THE CARRY FLAG, ZERO FLAG CONDITION
FBBF                 9314 DCB_IS_IDL:
FBBF                 9315 NO_DCB:
FBBF                 9316 ROUTINE_FINISHED:
FBBF FDE1            9317        POP      IY
                     9318
FBC1 C9              9319        RET
                     9320
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     9322 ********************************************************************************
                     9323 * __START_WR_CH_DEV ALLOWS THE CALLER TO SEND A PRINT COMMAND BY SPECIFYING
                     9324 * THE BUFFER OF THE STRING AND THE NUMBER OF BYTES TO PRINT.
                     9325 *       INPUT:  A ==>   DEVICE ID
                     9326 *               BC =>   NUMBER OF BYTES TO WRITE
                     9327 *               HL =>   SOURCE OF DATA
                     9328 *       OUTPUT:
                     9329 *               CONDITION FLAGS
                     9330 *                       Z :     NO ERRORS
                     9331 *                       NZ:     ERROR OCCURED
                     9332 *                               A ===> ERROR CODE
                     9333 ********************************************************************************
                     9334
                     9335 ;NO_DCB_FND3 CHANGED TO C_NO_DCB_FND3 TO AVOID CONFLICT WITH
                     9336 ;IDENTICAL LABEL ELSEWHERE
                     9337
                     9338        ;GLOBAL  __START_WR_CH_DEV
                     9339
FBC2                 9340 __START_WR_CH_DEV:
FBC2 FDE5            9341        PUSH    IY
FBC4 CDF446          9342        CALL    __FIND_DCB      ; FIND THE CORRECT DEVICE DCB
                     9343
FBC7 2015            9344        JR      NZ,C_NO_DCB_FND3
                     9345
FBC9 CDF496          9346        CALL    CHK_IF_INACTIVE             ; IS DCB ACTIVE?
FBCC 2010            9347        JR      NZ,DCB_IS_ACTIVE3           ; YUP
                     9348
FBCE FD7103          9349        LD      [IY+D_BUF_LEN_LO],C          ; LENGTH OF THE BUFFER
FBD1 FD7004          9350        LD      [IY+D_BUF_LEN_HI],B          ;
                     9351
FBD4 FD7501          9352        LD      [IY+D_BUF_ADR_LO],L         ; ADDRESS OF THE STRING
FBD7 FD7402          9353        LD      [IY+D_BUF_ADR_HI],H         ;
                     9354
FBDA FD360003        9355        LD      [IY+D_COM_STAT],DCB_WR       ; INITIATE THE WRITE DATA COMMAND
                     9356
FBDE                 9357 C_NO_DCB_FND3:
FBDE                 9358 DCB_IS_ACTIVE3:
FBDE FDE1            9359        POP     IY
FBE0 C9              9360        RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     9362 ********************************************************************************
                     9363 * __END_WR_CH_DEV WILL CHECK THE DEVICES DCB COM/STATUS BYTE AND RETURN
                     9364 * THE RESULT OF THE CHECK.
                     9365 *       INPUT:  A  ===> DEVICE ID
                     9366 *       OUTPUT: CONDITION FLAGS
                     9367 *               C :     COMMAND HAS FINISHED
                     9368 *               NC:     COMMAND HAS NOT BEEN PROCESSED
                     9369 *               Z :     NO ERROR OCCURED
                     9370 *               NZ:     ERROR OCCURED
                     9371 *                       A ===>  ERROR CODE
                     9372
                     9373 ********************************************************************************
                     9374
                     9375 ;NO_DCB_FND4            C_NO_DCB_FND4
                     9376 ;DCB_IS_IDL4 CHANGED TO CHAR_DCB_IS_IDL4 TO AVOID CONFLICT WITH
                     9377 ;IDENTICAL LABEL ELSEWHERE
                     9378
                     9379        ;GLB      __END_WR_CH_DEV
                     9380
    FBE1             9381 __END_WR_CH_DEV:
    FBE1 FDE5        9382         PUSH     IY
                     9383
    FBE3 CDF446      9384         CALL     __FIND_DCB
    FBE6 37          9385         SCF
                     9386
    FBE7 2012        9387         JR       NZ,C_NO_DCB_FND4          ; THERE WAS NO PRINTER DCB
                     9388
    FBE9 CDF4AE      9389         CALL     CHK_IF_IDLE               ; IS DEVICE IDL?
    FBEC 200D        9390         JR       NZ,CHAR_DCB_IS_IDL4
                     9391
    FBEE B7          9392         OR       A                         ; CLEAR THE CARRY FLAG
    FBEF FDCB007E    9393         BIT      CMND_COMPLETE_BIT,[IY+D_COM_STAT]       ; WAS THE COMMAND PROCESSED?
    FBF3 2806        9394         JR       Z,NOT_COMPLETED           ; NO, RETURN WITH THE CARRY FLAG CLEARED
                     9395                                            ; YES, THE COMMAND WAS PROCESSED
    FBF5 FD7E00      9396         LD       A,[IY+D_COM_STAT]         ; DID ANY ERRORS OCCUR?
    FBF8 FE80        9397         CP       CMND_FIN_STATUS           ;
    FBFA 37          9398         SCF
                     9399                                            ; NO ERROR, SET THE CARRY FLAG, ZERO FLAG CONDITION
    FBFB             9400 CHAR_DCB_IS_IDL4:
    FBFB             9401 C_NO_DCB_FND4:
    FBFB             9402 NOT_COMPLETED:
    FBFB FDE1        9403         POP      IY
    FBFD C9          9404         RET
                     9405 ;
                     9406 ;***************************************************************
                     9407 ;***************************************************************
                     9408 ;***************************************************************
                     9409 ;

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
8672	ACTIVE_DCB	A	8661
8934	ACTIVE_DCB1	A	8928
9053	ACTIVE_DCB3	A	9046
3819	ACTIVE_FILE	A	3792
322	ADD816	A	
1727	ADD_TO_BASE	A	1691,1696
5695	ADJUST	A	5693
2499	APSV	A	2600
2838	AREA_SONG_IS	A	3200
2160	ARM_MASK	A	2240
5529	AROUND_POP	A	5525
2497	ASTEP	A	3280,3361,3383
2490	ATN	A	3277
2588	ATN_SWEEP	A	3217
5716	ATTRIB_TBL	A	5690
1052	ATTR_DELETED	A	5812
1053	ATTR_DEL_BIT	A	3791,3921,5367
1054	ATTR_EXECUTE	A	5720
1056	ATTR_HOLE	A	4086,6522
1055	ATTR_HOLE_BIT	A	3785,5365,6192,6232
1047	ATTR_PERMANENT	A	5719,6505,6513
1049	ATTR_READ_PROT	A	5717
1051	ATTR_SYSTEM	A	6505,6513
1050	ATTR_USER	A	4190
1048	ATTR_WRITE_PROT	A	5718,6513
2902	B1	A	2905
1074	BAD_FNUM_ERR	A	4478,4591,4849,5079
5494	BASECMP	A	3823,5375
1578	BASE_FACTORS	A	1560
7371	BLANKING_CHAR	A	7409,7605,7700,7758
7695	BLANKS	A	7720,7837
7701	BLANK_FILL	A	7703
7684	BLANK_PORTION	A	7754
7700	BLANK_REST	A	7687
7719	BLANK_SCREEN	A	7603,7640
9115	BLK_DCB_IS_IDLE	A	9099
553	BLK_STRT_PTR	A	
7724	BLNK_SCRN_LOOP	A	7733
580	BLOCKS_REQ	A	3710,3714,3799,3927,3936,3978,3982,3991,4193
8126	BLOCK_ZERO	A	8230
7360	BS	A	7929
579	BUF_END	A	3736,4662,4675,4930,4949
578	BUF_START	A	3733,3828,3889,4032,4044,4097,4291,4341,4374,4660,4707,4722,4791,4928,5011,5022,5045
575	BYTES_REQ	A	4641,4819,4852,4905
576	BYTES_TO_GO	A	4643,4685,4693,4696,4767,4803,4808,4811,4820,4853,4907,4956,4969,4972,5060
8935	B_NO_DCB_FOUND1	A	8925
9052	B_NO_DCB_FOUND3	A	9043
9116	B_NO_DCB_FOUND4	A	9096
304	CALC_OFFSET	A	
8217	CALL_WP	A	8198,8241
1084	CANT_SYNC1	A	8454
1085	CANT_SYNC2	A	8480
1553	CASE_OF_CLR10	A	1550
1546	CASE_OF_COLOR	A	1527
1533	CASE_OF_GEN	A	1525
1540	CASE_OF_GEN10	A	1537
9400	CHAR_! IS_IDL	A	9390

LINE#	SYMBOL	TYPE	REFERENCES
2401	CHECK	A	2394,2396,2398
3777	CHECK_ENT	A	3863
6986	CHECK_FOR_ETX	A	7019
3981	CHECK_HOLE_SIZE	A	3972
5448	CHECK_IF_DIRECT	A	3764,5300
4985	CHECK_LAST_BLOC	A	4967
6098	CHECK_STATUS	A	6090
8184	CHK_D5	A	8174
6746	CHK_IF_IDLE	A	8976,9098,9303,9389
6702	CHK_IF_INACTIVE	A	6983,7161,8660,8927,9045,9260,9346
1408	CHK_REG_1	A	1403
8195	CHK_TAPE	A	8186
7655	CHK_X_WITH_MAX	A	7653
7659	CHK_Y_COORD	A	7657
7663	CHK_Y_WITH_MAX	A	7661
5020	CLEAR_DIRTY_BIT	A	
663	CLEAR_RAM_SIZE	A	8133
434	CLEAR_RAM_START	A	663,8134,8135
4432	CLOSE2	A	4437
4465	CLOSE3	A	4444
4457	CLOSE4	A	
7362	CLR_TO_EOL	A	7934
7363	CLR_TO_EOS	A	7935
9117	CMND_COMPLETED	A	
1201	CMND_COMPLETE_B	A	6654,6710,8552,8666,8981,9103,9307,9393
1203	CMND_FIN_STATUS	A	6658,8556,8670,8987,9109,9311,9397
9118	CMND_NOT_COMPLE	A	9104
7251	CMND_NOT_PROCES	A	7237
8122	COLD_START_ADDR	A	8227,8246
463	COLORTABLE	A	
8994	COMMAND_COMPLET	A	
7041	COMMAND_FINISHE	A	7033
8995	COMMAND_NOT_COM	A	8982
2316	CONTROLLER_0_MA	A	2323
853	CONTROLLER_0_PO	A	2215,2428
2317	CONTROLLER_1_BI	A	2327,2349
856	CONTROLLER_1_PO	A	
7905	CONVERT_XY	A	7726,7783,7806,7816,7858,7883
7361	CR	A	7930
2491	CTRL	A	3040
478	CURRENT_DEV	A	
483	CURRENT_PCB	A	6588,8284,8330,8336,8434,8524,8534,8538,8605,8617,8630
659	CURSOR	A	7372,7395,7471,7488,7514,7521,7536,7547,7557,7577,7629,7778,7857,7880
7370	CURSOR_CHAR	A	7411,7491
7372	CURSOR_X	A	7513
468	CUR_BANK	A	1897,1957,1966
4473	C_ERROR	A	4425,4427,4441
4481	C_ERROR2	A	4455
9357	C_NO_DCB_FND3	A	9344
9401	C_NO_DCB_FND4	A	9387
679	DCBS	A	
1066	DCB_BUSY	A	6721
8665	DCB_CMND_LOOP	A	8667
1185	DCB_IDLE	A	6707
1067	DCB_IDLE_ERR	A	6753
516	DCB_IMAGE	A	
9273	DCB_IS_ACTIVE	A	9262

LINE#	SYMBOL	TYPE	REFERENCES
1139	D_MAX_MSG_LN_HI	A	9148
1138	D_MAX_MSG_LN_LO	A	9146
1129	D_RET_COUNT	A	1131,1132
1132	D_RET_COUNT_HI	A	
1131	D_RET_COUNT_LO	A	
1125	D_SECT_NUM	A	9151,9152,9153,9154
1127	D_SEC_DEV_ID	A	9145
1145	D_SIZE	A	680,6596,8294,8340,8530,8572
1143	D_STATUS_FLAGS	A	6084,6678,8178,8190,8202
332	EFFECT_OVER	A	
3221	EFXOVER	A	3188
1766	ELSE_11	A	1763
1781	ELSE_12	A	1778
3297	ENDNOREP	A	3289
2482	ENDSDATA	A	3086
5619	END_FCB	A	5597
1768	END_IF_11	A	1765
1783	END_IF_12	A	1780
7045	END_PRINT_ROUTI	A	6981,7027
6680	END_R_D_D_S	A	6674
6662	END_R_S	A	6648
8499	END_SYNC	A	8457,8483
8583	END_S_A	A	8567
4752	ENOUGH	A	4688
1004	ENT_PER_BLOCK	A	3767,3862,5313,5353,6166
1075	EOF_ERR	A	4827,4998
5527	EOS	A	5507
97	EOS_CODE	A	1263
562	EOS_DAY	A	4208,5130,5158
100	EOS_GLB_RAM	A	428
98	EOS_GLB_TBL	A	694
99	EOS_JMP_TBL	A	118
561	EOS_MONTH	A	4206,5132,5160
101	EOS_PCB_DCB	A	670
8116	EOS_REV	A	8140
596	EOS_STACK	A	8131
560	EOS_YEAR	A	4204,5134,5162
7749	ERASE_THIS_LINE	A	7618,7628
7617	ERASE_TO_EOL	A	7957
7627	ERASE_TO_EOS	A	7956
6286	ERROR	A	6143,6212,6233,6273
5379	ERROR_CHECK	A	5376
6809	ERROR_OCCURED	A	6797
1155	ERROR_RETRY	A	5875,5897,5925,6010,6034
8823	ERROR_WITH_DCB	A	8813
8878	ERROR_WITH_DCB2	A	8867
4161	ERR_GEN	A	4170
1209	ETX	A	5506,5523,5532,5609,5634,6358,6369,6504,6512,6521,6943,6992,7070,7098,7169
7494	EXIT	A	7524,7545,7566,7586,7654,7658,7662,7666
88	FALSE	A	91
1011	FCB_ATTR	A	5809,5813
1023	FCB_BLOCK	A	3621,3622,3623,3624,3719,3720,3721,3722,3744,3745,3746,3747,3809,3810,3831,3837,3840,3843,3846,3854,3855 3856,3857,3882,3883,3895,3896,3948,3951,3954,3957,4033,4034,4035,4036,4048,4054,4055,4056,4057,4098,4099 4100,4101,4346,4348,4350,4352,4375,4376,4377,4378,4449,4450,4451,4452,4530,4531,4532,4533,4543,4545,4547 4549,4569,4570,4571,4572,4711,4713,4715,4717,4723,4724,4725,4726,4734,4737,4740,4743,4754,4757,4760,4763 5012,5013,5014,5015,5029,5031,5033,5035,5046,5047,5048,5049,5264,5265,5266,5267,5279,5280,5281,5282,5317 5324,5327,5330,5333,5339,5340,5341,5342,6267,6268,6269,6270

LINE#	SYMBOL	TYPE	REFERENCES
524	FCB_BUFFER	A	6139,6145,6181,6208,6222
569	FCB_DATA_ADDR	A	3620,3729,3753,4259,4431,4513,4652,4923,5210,5269,5278,5338,5349,6266,6347,6353,6374,6460,6461,6479
1021	FCB_DEVICE	A	3619,3671,3741,3851,3899,4031,4051,4096,4296,4305,4373,4448,4458,4529,4568,4721,5010,5044,5262,5277,5337 6265
1013	FCB_FIRST_BLOCK	A	4313,4315,4318,4321,4345,4347,4349,4351,4542,4544,4546,4548
568	FCB_HEAD_ADDR	A	3555,3605,3668,4255,4430,4512,4651,4915,5211,5219,5259,5581,6173,6225
1024	FCB_LAST_BLOCK	A	3759,3760,3761,3762,3836,3839,3842,3845,3949,3952,3955,3958,4314,4317,4320,4323,4325,4327,4329,4331,4333 4335,4337,4339,4733,4736,4739,4742,4753,4756,4759,4762,5295,5296,5297,5298,5323,5326,5329,5332
1016	FCB_LAST_COUNT	A	1018,4559,4560,4792,4793,5026,5027,5061,5062,5064,5065
1026	FCB_LENGTH	A	4256,4269,4433,4515,4654,4916,5218,5587
1014	FCB_MAX_LENGTH	A	4941,4944,6150,6151,6162,6163
1022	FCB_MODE	A	3669,3681,3788,3971,3979,4265,4294,4354,4369,4396,4439,4466,4521,4539,4541,4554,4672,4745,4788,4947,4983 4986,5021,5038,5073,5222,5593,5641,5680
1010	FCB_NAME	A	
1025	FCB_POINTER	A	3565,3566,3613,3614,3730,3731,3742,3743,3770,3771,3774,3775,3778,3779,3820,3821,3829,3830,3852,3853,3890 3891,3907,3908,3914,3915,4017,4020,4023,4024,4045,4046,4052,4053,4063,4064,4342,4343,4551,4552,4676,4677 4698,4699,4708,4709,4770,4771,4775,4776,4795,4796,4813,4814,4950,4951,4974,4975,5023,5024,5066,5067,5071 5072,5270,5271,5303,5305,5306,5308,5350,5351,5359,5360,5383,5385,5386,5388,6174,6175,6226,6227
1195	FCB_S	A	8163
1012	FCB_START_BLOCK	A	1013,3724,3725,3726,3727
1018	FCB_STORED_BYTE	A	
1015	FCB_USED_LENGTH	A	4312,4316,4362,4365,4562,4563,4940,4943,5006,5008,6147,6148
7359	FF	A	7932
542	FILENAME_CMPS	A	3549,3576,5372
528	FILE_COUNT	A	5274,5310,5356,6165,6199,6235
5382	FILE_DELETED	A	5368
4123	FILE_EXISTS	A	3824
1071	FILE_EXISTS_ERR	A	4124
492	FILE_NAME_ADDR	A	4446,4461
1079	FILE_NM_ERR	A	4162
538	FILE_NUMBR	A	5585,5620,5646
1469	FILL	A	1475
296	FILL_VRAM	A	8150
6619	FINISHED	A	6613
7252	FINISHED_WO_ERR	A	7239
2159	FIRE_MASK	A	2255
565	FMGR_DIR_ENT	A	
574	FNUM	A	4642,4906
7598	FORM_FEED	A	7959
549	FOUND_AVAIL_ENT	A	3717,3804,3866,3875
3659	FOUND_ENTRY	A	3805,3808,3867,3876,3880
3874	FOUND_HOLE	A	3786
6611	FOUND_IT	A	6603
2494	FPSV	A	2549
2489	FREQ	A	2688,2692,2695
2529	FREQ_SWEEP	A	3218
2495	FSTEP	A	2533,2568,3279,3360
3865	FULL_DIR	A	3838,3841,3844,3847
1077	FULL_DIR_ERR	A	4128
4127	FULL_DIR_EXIT	A	3868
5377	FULL_FILENAME	A	5374
1078	FULL_TAPE_ERR	A	4132
5286	GET_BUFF_ADDR	A	
5346	GET_BUFF_ADDR1	A	
7882	GET_CHAR_UNDER	A	7489
7879	GET_CHAR_UNDER_	A	7484
2441	GET_CO' 'LLER_	A	2432

LINE#	SYMBOL	TYPE	REFERENCES
5171	GET_ERROR	A	5167
4148	GET_FILE_NM_LEN	A	4185
6991	GET_NEXT_ASCII	A	7001
7168	GET_NEXT_ASCII2	A	7178
5651	GET_OUT_ERR	A	5630
5468	GET_OUT_HERE	A	5477
6075	GET_STATUS_FLAG	A	5918,6056
302	GET_VRAM	A	
9203	GOOD_BYE	A	9196
7650	GOTO_XY	A	7955
7364	GOTO_XY_CHAR	A	7936
3917	GOT_BLOCK	A	3893,3903
2220	GOT_CORRECT_CON	A	2217
3689	GOT_FILE_SIZE	A	3676,3680
4167	GOT_IT	A	4155
4290	GOT_ONE	A	4267
8242	GO_TO_TAPECODE	A	8232
6714	G_OK	A	6708
1711	HAVE_CNT	A	1706
1724	HAVE_CNT2	A	1721
6904	HAVE_THE_KEY	A	6886
762	HI_AUXILLIARY_S	A	783,786,789,792
764	HI_COLECOVISION	A	807,810,813,816
763	HI_EXPANSION_80	A	795,798,801,804
761	HI_INTRINSIC_80	A	771,774,777,780
6520	HOLE_FILE_NAME	A	4091,6217
7365	HOME	A	7933
2480	INACTIVE	A	2942,3201,3251,3298,3450
2438	INCREMENT_0_DIR	A	2434
2453	INCREMENT_1_DIR	A	2449
6494	INITIAL_DIRECTO	A	6346,6529
6529	INIT_INFO_SIZE	A	
5221	INIT_LOOP	A	5224
1193	INIT_PCB_ADDR	A	
298	INIT_TABLE	A	
1559	INIT_TABLE80	A	1521,1528
1571	INIT_TABLE90	A	1539,1542,1552,1555
1358	INPUT_LOOP	A	1363,1366
2425	INT_BIT	A	2431,2445
1364	IN_DEC_HI_BYTE	A	1357
90	IN_EOS	A	
6372	ITS_AN_ETX	A	6360
7022	ITS_ETX	A	6994
5632	ITS_ETX1	A	5611
7186	ITS_ETX2	A	7171
6244	ITS_HOLE	A	6195
2161	JOY_MASK	A	2258
4746	JP_TO_READ3	A	4735,4738,4741,4744
2956	JUKE_BOX	A	3292
2158	KBD_MASK	A	2243
1205	KBD_NAK	A	6889
2157	KBD_NULL	A	2140,2155
497	KEYBOARD_BUFFER	A	6854,6905
1151	KEYBOARD_ID	A	6831,6853,6881,8641,8685
3085	L1	A	3098
3216	L10	A	3207
3286	L13	A	3263

LINE#	SYMBOL	TYPE	REFERENCES
3305	L14	A	3287
3338	L15	A	3308
3366	L16	A	3343
3388	L17	A	3368
3402	L18	A	3397
3123	L2	A	3154
2546	L20	A	2535
3458	L20_LOAD_NEX	A	3455
2636	L23	A	2611
2670	L24	A	2662
3038	L5	A	3032
3077	L6	A	3034,3044,3048
3063	L7	A	3056
3067	L8	A	
3151	L9	A	3131
4787	LAST_BLOCK	A	4764
6198	LAST_ENTRY	A	6169
2237	LEAVE_THEM_OFF	A	2235
7369	LEFT	A	7939
4107	LETS_GET_OUT	A	
7358	LF	A	7543,7931
654	LINEBUFFER_	A	7698,7728,7784,7822,7830,7838,7859
308	LOAD_ASCII	A	
6356	LOAD_DIR	A	6367
4179	LOAD_NEW_ENTRY_	A	3938,4011
3244	LOAD_NEXT_NOTE	A	2978,3224
2718	LOCL_DECLSN	A	2551,2602,2610
2739	LOCL_DECMSN	A	
2758	LOCL_MSNTOLSN	A	2557,2608
2780	LOC_ADD816	A	2569
977	LOC_IN_ALPHA	A	1906
8444	LOOP1	A	8462
8470	LOOP2	A	8488
8544	LOOP3	A	8563,8579
8551	LOOP4	A	8553
8612	LOOP5	A	8615
8401	LOOP7	A	8405
8398	LOOP8	A	8407
2076	LOOP_EVERY_BYTE	A	2080
2056	LOOP_EVERY_SPRI	A	2084
756	LO_BOOT_ROM_N_A	A	771,783,795,807
758	LO_EXPANSION_O_	A	777,789,801,813
757	LO_INTRINSIC_RA	A	774,786,798,810
759	LO_OS7_N_INTRIN	A	780,792,804,816
4108	MAKE_ERROR	A	3750,3765,3859,3902,3940,4015,4040,4060,4104,4120,4125,4129,4133
4213	MAKE_ERR_1	A	4186
5534	MATCH	A	5515
5516	MATCHLESS	A	5510
1073	MATCH_ERR	A	5544
1157	MAX_DEV_ADDR	A	8562,8578
770	MEM_CNFGO0	A	1899,8720
773	MEM_CNFGO1	A	8152
776	MEM_CNFGO2	A	
779	MEM_CNFGO3	A	1969
782	MEM_CNFGO4	A	
785	MEM_CNFGO5	A	
788	MEM_C[illegible]6	A	

LINE#	SYMBOL	TYPE	REFERENCES
791	MEM_CNFGO7	A	
794	MEM_CNFGO8	A	
797	MEM_CNFGO9	A	
800	MEM_CNFGOA	A	
803	MEM_CNFGOB	A	
806	MEM_CNFGOC	A	
809	MEM_CNFGOD	A	
812	MEM_CNFGOE	A	
815	MEM_CNFGOF	A	
841	MEM_SWITCH_PORT	A	1953
3445	MODBO	A	3281,3300,3330,3362,3384,3419
5676	MODE_CHECK	A	4309,4668,4936
1041	MODE_DIRTY	A	4443,4525
1040	MODE_DIRTY_BIT	A	4539,4983,5021,5073
1036	MODE_EXEC	A	1037
1043	MODE_LAST	A	
1042	MODE_LAST_BLOCK	A	4369,4541,4672,4745,4788,4947,4986
1037	MODE_MAX	A	5687
1044	MODE_MODE	A	4355,4555,5039,5594,5642,5684
1033	MODE_READ	A	1034
1039	MODE_REMAINDER_	A	3681,3788,3971,3979
1032	MODE_UNUSED	A	1033,4396,4466,5222
1035	MODE_UPDATE	A	1036,5040
1034	MODE_WRITE	A	1035,4356,4556
531	MOD_FILE_COUNT	A	6200,6236
3967	MORE_BLOCKS	A	3950,3953,3956,3959
5005	MORE_FILE	A	4987
5009	MORE_FILE1	A	5007
7534	MOVE_DOWN	A	7953,7960
7608	MOVE_HOME	A	7958
7577	MOVE_LEFT	A	7952,7962
7556	MOVE_RIGHT	A	7951
7520	MOVE_UP	A	7954
320	MSNTOLSN	A	
1059	NAME_LENGTH	A	5503,5605
1821	NEGTV	A	1813
8363	NET_DELAY_COUNT	A	8374
844	NET_RESET_PORT	A	8369
8376	NET_RES_DELAY	A	8381
7489	NEW_CURSOR	A	7515,7548
589	NEW_HOLE_SIZE	A	3986,4073,6161,6248
586	NEW_HOLE_START	A	3995,3999,4003,4007,4077,4079,4081,4083
5504	NEXT_CHAR	A	5513
5607	NEXT_CHAR1	A	5616
5457	NEXT_CHECK	A	5464
3905	NEXT_DIR	A	3922
3769	NEXT_ENT	A	3826
2345	NEXT_IX	A	2341
2492	NLEN	A	2536,2539,3278
1694	NOT_COLOR_TBL_A	A	1687
9402	NOT_COMPLETED	A	9394
2348	NOT_CONTROLLER_	A	2324
5825	NOT_DELETED	A	5811
5475	NOT_DIR	A	5460
4066	NOT_END	A	4026
6263	NOT_HOLE	A	6193
6759	NOT_IDLE	A	6751

LINE#	SYMBOL	TYPE	REFERENCES
6172	NOT_LAST_ENTRY	A	
7250	NOT_PR_NAK	A	7243
5777	NOT_RENAMED	A	5746,5756,5770
2406	NOT_SAME	A	2403
4371	NOT_SIZE_1	A	4363,4366
7469	NOT_SPECIAL	A	7448
4732	NOT_TAPE1	A	
8459	NOT_TIME1	A	8448,8452
8485	NOT_TIME2	A	8474,8478
1069	NO_DATE_ERR	A	5172
9315	NO_DCB	A	9301
9274	NO_DCB_DEFINED	A	9258
8673	NO_DCB_FOUND	A	8658
7195	NO_DCB_FOUND1	A	7159
8708	NO_DCB_FOUND2	A	8702
8993	NO_DCB_FOUND5	A	8974
1091	NO_DIR_ERR	A	5476
5415	NO_ENT	A	5325,5328,5331,5334,5366
5951	NO_ERRORS	A	5939
4278	NO_FCBS	A	
1072	NO_FCB_ERR	A	4286
1070	NO_FILE_ERR	A	5419,5628
5540	NO_MATCH	A	5518,5524,5533
5625	NO_MATCH1	A	5636
6900	NO_NAK	A	6890
4578	NO_READ	A	4565
5896	NO_READ_ERRORS	A	5883,5887
6438	NO_SECTORS_TO_I	A	6425
3861	NO_TIMEOUT	A	
6275	NO_TRIM	A	6159,6241
6032	NO_WRITE_ERRORS	A	6018,6022
656	NUM_COLUMNS	A	7721,7818,7839
1062	NUM_FCBS	A	4276,4426,4508,4647,4911,5217,5621
655	NUM_LINES	A	7393,7599,7633,7808
7942	NUM_SPCL_CHARS	A	7446
2476	OFF	A	2932,2934,2936,2938,3003,3011,3019,3027
649	OLDCHAR_	A	7410,7606,7759,7853,7885
8125	ONE_BLOCK	A	
4383	OPEN5	A	4357
4395	OP_ERR	A	4307,4310,4381
5828	OTHER_ERR	A	5808,5819
1307	OUTPUT_LOOP	A	1312,1315
1313	OUT_DEC_HI_BYTE	A	1306
3486	OUT_TO_SOUND_PO	A	2675,2691,2704,3033,3059
4264	O_NEXT_FCB	A	4277
3327	PASS1	A	3324
5082	PAST_WRITE_ERR	A	5077
461	PATTRNGENTBL	A	1849
459	PATTRNNAMETBL	A	
676	PCB	A	8283,8291,8292
1164	PCB_IDLE	A	
1175	PCB_RESET	A	1176
1176	PCB_RESET_ACK	A	
1172	PCB_SNA	A	1173,8610
1173	PCB_SNA_ACK	A	8614
1166	PCB_SYNC1	A	1167,8438
1167	PCB_S _ACK	A	8461

LINE#	SYMBOL	TYPE	REFERENCES
1169	PCB_SYNC2	A	1170,8464
1170	PCB_SYNC2_ACK	A	8487
1178	PCB_WAIT	A	1179
1179	PCB_WAIT_ACK	A	
619	PERSONAL_DEBOUN	A	2321
328	PLAY_IT	A	
312	POLLER	A	
2329	POLL_CONTROLLER	A	2355
277	PORT_COLLECTION	A	8143
839	PORT_TABLE	A	1971
2786	POS	A	2783
9137	PREP_DCB	A	8930,9048
1152	PRINTER_ID	A	6978,7007,7030,7156,7181,7192,7233,7274,8645,8689
503	PRINT_BUFFER	A	6942,6944,6945,7069,7071,7073,7097,7099,7101
7042	PRINT_CMND_FAIL	A	7013
7043	PRNT_CMND_FAILE	A	
3199	PROCESS_DATA_AR	A	3092
1090	PROG_NON_EXIST	A	6551
5711	PROTECT	A	5708
5706	PROTECT_FAULT	A	5700
7044	PRTR_DCB_ACTIVE	A	6984,7162
1086	PRT_ERR	A	5707
6977	PR_BUFF2	A	
1207	PR_NAK	A	7012,7035,7241
658	PTRN_NAME_TBL	A	7396,7915
626	PTR_TO_LST_OF_S	A	2812,2890
627	PTR_TO_S_ON_0	A	2914,3029,3116,3139
628	PTR_TO_S_ON_1	A	2915,3006,3117
629	PTR_TO_S_ON_2	A	2916,3014,3118
630	PTR_TO_S_ON_3	A	2917,3022,3119
2807	PT_IX_TO_SxDATA	A	2961,3083,3121
281	PUT_ASCII	A	
7855	PUT_CHAR_ON_SCR	A	7412,7470,7492
7490	PUT_CURSOR_ON	A	7619,7636,7641
7852	PUT_OLD_ON_SCRE	A	7511,7527,7608,7803
300	PUT_VRAM	A	
1805	PX_2_P_P_1	A	1808
306	PX_TO_PTRN_POS	A	
1100	P_COM_STAT	A	8438,8460,8464,8486,8610,8613
1106	P_NUM_DCBS	A	6589,8436,8539,8565,8570,8581
1102	P_REL_ADDR	A	1103,1104
1104	P_REL_ADDR_HI	A	8494,8608
1103	P_REL_ADDR_LO	A	8493,8607
1109	P_SIZE	A	677,6594,8294,8340,8525,8535
3539	QUEER	A	3534
520	QUERY_BUFFER	A	5744,5754,5757,5768,5806,5809,5813
3577	Q_ERROR	A	3559
1082	RANGE_ERR	A	5710
5709	RANGE_NONO	A	5686,5688
9198	RD_DEV_LOOP	A	9201
6793	RD_KBD_LOOP	A	
8815	RD_NOT_COMPLETE	A	8821
5273	RD_TMEOUT	A	
5336	RD_TMEOUT1	A	
4653	READ1	A	4658
4671	READ3	A	4747
4719	READ4	A	4712,4714,4716

LINE#	SYMBOL	TYPE	REFERENCES
8226	READ_AGAIN	A	8237,8240
5867	READ_BLOCK	A	3748,3858,3901,4058,4379,4574,4728,5050,5283,5343
5914	READ_COMMAND_CO	A	5905
4851	READ_ERR1	A	4669,4730
4848	READ_ERR2	A	4646,4648
5957	READ_ERROR	A	5893,5912,5923
7777	READ_FROM_CURSO	A	7750
4567	READ_IT	A	4557
4790	READ_LAST	A	4673
4766	READ_LAST1	A	4755,4758,4761,4806
5877	READ_LOOP	A	5894
2331	READ_N_DEBOUNCE	A	2326
294	READ_REGISTER	A	
4359	READ_TIMEOUT	A	
290	READ_VRAM	A	1668,7789,7823,7886
8569	RECVD_VALID_STA	A	8557
1080	RENAME_ERR	A	
8298	REPEAT_SYNC1	A	8300
8344	REPEAT_SYNC2	A	8346
6653	REQUEST_LOOP	A	6655
4514	RES1	A	4519
4540	RES2	A	4526
682	RESERVED_BYTE	A	
4538	RESET_BIT	A	
4043	RESET_THE_POINT	A	
7527	RESTORE_OLD	A	7541,7562,7570,7582,7590,7668
4399	REST_REGS	A	4388
535	RETRY_COUNT	A	5876,5890,5926,5941,6011,6025
5899	RETRY_FOR_ERROR	A	5908,5910
6036	RETRY_WRITE	A	6046,6048
7511	RETURN	A	7961
440	REV_NUM	A	8141
7367	RIGHT	A	7940
5075	ROOM1_OK	A	
5059	ROOM_OK	A	4959
6911	ROUTINE_DONE	A	6884,6902,6906
9316	ROUTINE_FINISHE	A	9308
4838	RQ_STAT_ERR	A	
1088	RQ_TP_STAT_ERR	A	4844
4586	R_ERROR	A	4507,4509,4523
4594	R_ERROR2	A	4536,4576
4828	R_GET_OUT	A	4859
631	SAVE_CTRL	A	2924,3042
7488	SAVE_CURSOR	A	7475,7481,7528,7610
5316	SCAN1	A	5392
5355	SCAN2	A	5314,5390
5394	SCAN3	A	5380
5409	SCAN_ERROR	A	5284,5301,5344
5252	SCAN_FOR_FILE	A	3558,3609
7812	SCROLL_LOOP	A	7835
7543	SCROLL_POSIBLE	A	7539
7802	SCROLL_UP	A	7485,7546
508	SECTORS_TO_INIT	A	6340,6376,6420
511	SECTOR_NO	A	6414,6430,6480
6429	SECT_INIT_LOOP	A	6436
7029	SEND_ANOTHER_WR	A	7036
1683	SET_C	A	1625,1665

LINE#	SYMBOL	TYPE	REFERENCES
7667	SET_CURSOR	A	7665
2112	SET_READ	A	1345
4948	SET_UP	A	4942,4945
3548	SET_UP_A	A	3545
3712	SET_UP_DIR	A	3683
3752	SET_UP_FCB	A	
2121	SET_WRITE	A	1294,1467,2046
6908	SHOW_COMPLETE	A	6893
1217	SKIP	A	
3078	SND_MANAGER	A	
865	SOUNDPORT	A	2930,3489
330	SOUNDS	A	
324	SOUND_INIT	A	
9233	SD_LONG	A	9226
7950	SPCL_VECTOR_TBL	A	7450
7928	SPECIAL_CHARS	A	7445,7942
2318	SPINNER_ENABLE_	A	2340
617	SPIN_SW0_CT	A	2214,2430
618	SPIN_SW1_CT	A	
455	SPRITEATTRTBL	A	2045
457	SPRITEGENTBL	A	
2874	SR1ATN	A	2932,3003,3004
2873	SR1FRQ	A	3005
2876	SR2ATN	A	2934,3011,3012
2875	SR2FRQ	A	3013
2878	SR3ATN	A	2936,3019,3020
2877	SR3FRQ	A	3021
4152	SRCH_LOOP	A	4159
2883	SRNATN	A	2938,3027,3028
2882	SRNCTL	A	3046
592	STACK_START	A	
582	START_BLOCK	A	
5496	STRCMP	A	5378
862	STROBE_RESET_PO	A	
859	STROBE_SET_PORT	A	2211
6852	STRT_RD_KBD_LOO	A	
91	SUPERGAME	A	444,485
279	SWITCH_MEM	A	1902,1913,1970,2000,8153,8721
768	SWITCH_TABLE	A	
8424	SYNC_DELAY_B	A	8442,8468
8423	SYNC_DELAY_DE	A	8441,8450,8467,8476
3631	S_ERROR	A	3610
4131	TAPE_FULL	A	3965,3976,3985
1153	TAPE_ID	A	7300,8169,8196,8200,8214,8223,8236,8649,8693
3618	TAPE_TIMED_OUT	A	
621	TEMP_STACK	A	1894
1196	THREE1K_BLKS	A	8162
4528	TIMED_OUT	A	
1211	TIMEOUT	A	5885,5907,6020,6044,8239
4050	TIMEOUT_IN_READ	A	
4095	TIME_TO_WRITE	A	3942
3053	TONE_OUT	A	3007,3015,3023
4118	TOO_BIG	A	3705,3709
1076	TOO_BIG_ERR	A	4119
3906	TOO_SMALL	A	3934
87	TRUE	A	90
6167	TRY_AGAIN	A	6170

LINE#	SYMBOL	TYPE	REFERENCES
7564	TRY_MOVE_DOWN	A	7560
7584	TRY_SCROLL_UP	A	7580
6012	TRY_WRITE_AGAIN	A	6030
326	TURN_OFF_SOUND	A	8145
3425	TYPE3	A	3390
6092	UNIT1	A	6086
7366	UP	A	7937
2657	UPATNCTRL	A	3039,3047,3064
314	UPDATE_SPINNER	A	
2687	UPFREQ	A	3065
657	UPPER_LEFT	A	7394,7601,7609,7804
3113	UP_CH_DATA_PTRS	A	2979,3230
577	USER_BUF	A	3553,3564,3603,3612,4640,4700,4704,4772,4815,4904,4976,4980,5068
581	USER_NAME	A	3672,3822,4183,5257,5370
3879	USE_ENTRY	A	3869
3947	USE_HOLE	A	3877
5648	U_GET_OUT	A	
5583	U_NEXT_FCB	A	5623
5701	U_OK	A	
847	VDP_CTRL_PORT	A	1393,1435,2124
850	VDP_DATA_PORT	A	
449	VDP_MODE_WORD	A	1404,1411,1519,1689
451	VDP_STATUS_BYTE	A	1438
709	VECTOR_08H	A	
712	VECTOR_10H	A	
715	VECTOR_18H	A	
718	VECTOR_20H	A	
721	VECTOR_28H	A	
724	VECTOR_30H	A	
727	VECTOR_38H	A	
730	VECTOR_66H	A	
984	VOL_ATTR	A	
554	VOL_BLK_SZ	A	
989	VOL_DAY	A	
990	VOL_DES_LENGTH	A	6348,6392
983	VOL_DIRSIZE	A	3755,5291,6378
985	VOL_DIR_CHECK	A	5459
988	VOL_MONTH	A	
982	VOL_NAME	A	
986	VOL_SIZE	A	6388,6389
987	VOL_YEAR	A	
453	VRAM_ADDR_TABLE	A	1513,1729
6801	WAITING	A	6806
6442	WE_GOT_ERRORS	A	6418,6434
8715	WP	A	8737
4917	WRITE1	A	4919
4924	WRITE2	A	4926
4939	WRITE3	A	5041
5037	WRITE4	A	5030,5032,5034
6476	WRITE_1_BLOCK	A	6416,6433
5043	WRITE_AGAIN	A	
6005	WRITE_BLOCK	A	3625,4038,4102,4454,4535,5016,6272,6487
7196	WRITE_CMND_SENT	A	7184
7005	WRITE_COMMAND_A	A	7015
6052	WRITE_COMMAND_C	A	6042
7017	WRITE_DONE	A	7010
5080	WRITF `1	A	4937,5018,5052

LINE#	SYMBOL	TYPE	REFERENCES
5078	WRITE_ERR2	A	4910,4912
6061	WRITE_ERROR	A	6028,6050
3630	WRITE_ERRORS	A	3627
6064	WRITE_OK	A	6059
292	WRITE_REGISTER	A	1572
4447	WRITE_TIMEOUT	A	
288	WRITE_VRAM	A	1628,1910,7729,7761,7831,7841,7862
8869	WR_NOT_COMPLETE	A	8876
310	WR_SPR_ATTRIBUT	A	
651	X_MAX	A	7402,7473,7558,7588,7655,7779
650	X_MIN	A	7399,7476,7512,7568,7578,7638,7651
653	Y_MAX	A	7407,7479,7537,7564,7663
652	Y_MIN	A	7404,7522,7584,7631,7659
6454	ZERO_OUT_BUFFER	A	6344,6427

FILE/PROG NAME	PROGRAM	DATA	COMMON	ABSOLUTE	DATE	TIME	COMMENTS
A_uOS_00:N_EOS	E000				Thu, 14 Jun 1984,	14:15	Rev 07 - jkl
next address	E618						
DIR_HANDL:N_EOS	E618				Thu, 14 Jun 1984,	14:16	
OPENS:N_EOS	EA00				Thu, 14 Jun 1984,	14:18	
RD_WT:N_EOS	EC17				Thu, 14 Jun 1984,	14:19	
DATES:N_EOS	EEC5				Thu, 14 Jun 1984,	14:16	
UTILS:N_EOS	EEEA				Thu, 14 Jun 1984,	14:19	
EOS_MKDIR:N_EOS	F323				Thu, 14 Jun 1984,	14:17	
next address	F442						
EOS_RET:N_EOS	F442				Thu, 14 Jun 1984,	14:18	REV 1 - RPD
EOS_GEN:N_EOS	F446				Thu, 14 Jun 1984,	14:17	
EOS_KBD:N_EOS	F4BA				Thu, 14 Jun 1984,	14:17	
EOS_PRTR:N_EOS	F4FC				Thu, 14 Jun 1984,	14:18	
EOS_TAPE:N_EOS	F5D7				Thu, 14 Jun 1984,	14:18	
CONS_OUT:N_EOS	F5DC				Thu, 14 Jun 1984,	14:15	Rev 18 - RPD
next address	F832						
EOS_START:N_EOS	F832				Thu, 14 Jun 1984,	14:18	Rev 07 - rfj
EOS_UTIL:N_EOS	F8F6				Thu, 14 Jun 1984,	14:18	
EOS_BLK:N_EOS	FA9E				Thu, 14 Jun 1984,	14:16	
EOS_CHAR:N_EOS	FB64				Thu, 14 Jun 1984,	14:16	
next address	FBFE						
TABLES:N_EOS	FBFF				Thu, 14 Jun 1984,	14:19	Rev 00 - RPD
next address	FC30						
EOSCMN_06:N_EOS	FC30	FD60		FECO-FFFF	Thu, 14 Jun 1984,	14:32	Rev 12 - rfj
next address	FD5F	FEA7					

XFER address= 0000 Defined by DEFAULT
absolute & link_com file name=N_EOS:N_EOS
Total# of bytes loaded= 1FE5

SYMBOL	R	VALUE	DEF BY	REFERENCES
ADD816	P	FD4D	EOSCMN_06:N_EOS	
BASECMP	P	F054	UTILS:N_EOS	DIR_HANDL:N_EOS
BLK_STRT_PTR	D	FDDC	EOSCMN_06:N_EOS	DIR_HANDL:N_EOS
BLOCKS_REQ	D	FE0C	EOSCMN_06:N_EOS	DIR_HANDL:N_EOS
BUF_END	D	FE0A	EOSCMN_06:N_EOS	RD_WT:N_EOS DIR_HANDL:N_EOS
BUF_START	D	FE08	EOSCMN_06:N_EOS	UTILS:N_EOS RD_WT:N_EOS OPENS:N_EOS DIR_HANDL:N_EOS
BYTES_REQ	D	FE02	EOSCMN_06:N_EOS	RD_WT:N_EOS
BYTES_TO_GO	D	FE04	EOSCMN_06:N_EOS	RD_WT:N_EOS
CALC_OFFSET	P	FD32	EOSCMN_06:N_EOS	
CHECK_IF_DIRECT	P	F035	UTILS:N_EOS	DIR_HANDL:N_EOS
CHK_IF_IDLE	P	F4AE	EOS_GEN:N_EOS	EOS_CHAR:N_EOS EOS_BLK:N_EOS
CHK_IF_INACTIVE	P	F496	EOS_GEN:N_EOS	EOS_CHAR:N_EOS EOS_BLK:N_EOS EOS_UTIL:N_EOS EOS_PRTR:N_EOS
CLEAR_RAM_SIZE	A	0147	EOSCMN_06:N_EOS	EOS_START:N_EOS
CLEAR_RAM_START	D	FD60	EOSCMN_06:N_EOS	EOS_START:N_EOS
COLORTABLE	D	FD6C	EOSCMN_06:N_EOS	
CONTROLLER_0_PD	A	FC2B	EOSCMN_06:N_EOS	A_uOS_00:N_EOS
CONTROLLER_1_PD	A	FC2C	EOSCMN_06:N_EOS	
CURRENT_DEV	D	FD6F	EOSCMN_06:N_EOS	
CURRENT_PCB	D	FD70	EOSCMN_06:N_EOS	EOS_UTIL:N_EOS EOS_GEN:N_EOS
CURSOR	D	FEA5	EOSCMN_06:N_EOS	CONS_OUT:N_EOS
CUR_BANK	D	FD6E	EOSCMN_06:N_EOS	A_uOS_00:N_EOS
DCB_IMAGE	D	FD8B	EOSCMN_06:N_EOS	
DECLSN	P	FD44	EOSCMN_06:N_EOS	
DECMSN	P	FD47	EOSCMN_06:N_EOS	
DEFAULT_BT_DEV	D	FD6F	EOSCMN_06:N_EOS	EOS_START:N_EOS
DEVICE_ID	D	FD72	EOSCMN_06:N_EOS	EOS_CHAR:N_EOS EOS_MKDIR:N_EOS
DIRECTORY_CHECK	P	F3E7	EOS_MKDIR:N_EOS	UTILS:N_EOS
DIR_BLOCK_NO	D	FDD9	EOSCMN_06:N_EOS	DIR_HANDL:N_EOS
EFFECT_OVER	P	FD5C	EOSCMN_06:N_EOS	
EOS_DAY	D	FDE2	EOSCMN_06:N_EOS	DATES:N_EOS DIR_HANDL:N_EOS
EOS_MONTH	D	FDE1	EOSCMN_06:N_EOS	DATES:N_EOS DIR_HANDL:N_EOS
EOS_STACK	D	FE58	EOSCMN_06:N_EOS	EOS_START:N_EOS
EOS_YEAR	D	FDE0	EOSCMN_06:N_EOS	DATES:N_EOS DIR_HANDL:N_EOS
FCB_BUFFER	D	FDBA	EOSCMN_06:N_EOS	UTILS:N_EOS
FCB_DATA_ADDR	D	FDFF	EOSCMN_06:N_EOS	EOS_MKDIR:N_EOS UTILS:N_EOS RD_WT:N_EOS OPENS:N_EOS DIR_HANDL:N_EOS
FCB_HEAD_ADDR	D	FDFD	EOSCMN_06:N_EOS	UTILS:N_EOS RD_WT:N_EOS OPENS:N_EOS DIR_HANDL:N_EOS
FILENAME_CMPS	D	FDD8	EOSCMN_06:N_EOS	UTILS:N_EOS DIR_HANDL:N_EOS
FILE_COUNT	D	FDD4	EOSCMN_06:N_EOS	UTILS:N_EOS
FILE_NAME_ADDR	D	FD73	EOSCMN_06:N_EOS	OPENS:N_EOS
FILE_NUMBR	D	FDD7	EOSCMN_06:N_EOS	UTILS:N_EOS
FILL_VRAM	P	FD26	EOSCMN_06:N_EOS	EOS_START:N_EOS
FMGR_DIR_ENT	D	FDE3	EOSCMN_06:N_EOS	OPENS:N_EOS
FNUM	D	FE01	EOSCMN_06:N_EOS	RD_WT:N_EOS
FOUND_AVAIL_ENT	D	FDDB	EOSCMN_06:N_EOS	DIR_HANDL:N_EOS
GET_VRAM	P	FD2F	EOSCMN_06:N_EOS	
HOLE_FILE_NAME	P	F428	EOS_MKDIR:N_EOS	UTILS:N_EOS DIR_HANDL:N_EOS
INIT_TABLE	P	FD29	EOSCMN_06:N_EOS	
INT_VCTR_TBL	A	FBFF	EOSCMN_06:N_EOS	
KEYBOARD_BUFFER	D	FD75	EOSCMN_06:N_EOS	EOS_KBD:N_EOS
LINEBUFFER_	D	FE7E	EOSCMN_06:N_EOS	CONS_OUT:N_EOS
LOAD_ASCII	P	FD38	EOSCMN_06:N_EOS	
MEM_CNFG00	A	FC17	EOSCMN_06:N_EOS	EOS_UTIL:N_EOS A_uOS_00:N_EOS
MEM_CNFG01	A	FC18	EOSCMN_06:N_EOS	EOS_START:N_EOS
MEM_CNFG02	A	FC19	EOSCMN_06:N_EOS	
MEM_CNFG03	A	FC1A	EOSCMN_06:N_EOS	EOS_START:N_EOS A uOS_00:N_EOS

MEM_CNFG04	A	FC1B	EOSCMN_06:N_EOS
MEM_CNFG05	A	FC1C	EOSCMN_06:N_EOS
MEM_CNFG06	A	FC1D	EOSCMN_06:N_EOS
MEM_CNFG07	A	FC1E	EOSCMN_06:N_EOS
MEM_CNFG08	A	FC1F	EOSCMN_06:N_EOS
MEM_CNFG09	A	FC20	EOSCMN_06:N_EOS
MEM_CNFG0A	A	FC21	EOSCMN_06:N_EOS
MEM_CNFG0B	A	FC22	EOSCMN_06:N_EOS
MEM_CNFG0C	A	FC23	EOSCMN_06:N_EOS
MEM_CNFG0D	A	FC24	EOSCMN_06:N_EOS
MEM_CNFG0E	A	FC25	EOSCMN_06:N_EOS
MEM_CNFG0F	A	FC26	EOSCMN_06:N_EOS
MEM_SWITCH_PORT	A	FC27	EOSCMN_06:N_EOS
MODE_CHECK	P	F0D9	UTILS:N_EOS
MOD_FILE_COUNT	D	FDD5	EOSCMN_06:N_EOS
MSNTOLSN	P	FD4A	EOSCMN_06:N_EOS
NET_RESET_PORT	A	FC28	EOSCMN_06:N_EOS
NEW_HOLE_SIZE	D	FE1A	EOSCMN_06:N_EOS
NEW_HOLE_START	D	FE16	EOSCMN_06:N_EOS
NUM_COLUMNS	D	FEA0	EOSCMN_06:N_EOS
NUM_LINES	D	FE9F	EOSCMN_06:N_EOS
OLDCHAR_	D	FE79	EOSCMN_06:N_EOS
PATTRNGENTBL	D	FD6A	EOSCMN_06:N_EOS
PATTRNNAMETBL	D	FD68	EOSCMN_06:N_EOS
PCB	A	FEC0	EOSCMN_06:N_EOS
PERSONAL_DEBOUN	D	FE5A	EOSCMN_06:N_EOS
PLAY_IT	P	FD56	EOSCMN_06:N_EOS
POLLER	P	FD3E	EOSCMN_06:N_EOS
PORT_COLLECTION	P	FD11	EOSCMN_06:N_EOS
PORT_TABLE	A	FC27	EOSCMN_06:N_EOS
PRINT_BUFFER	D	FD76	EOSCMN_06:N_EOS
PTRN_NAME_TBL	D	FEA3	EOSCMN_06:N_EOS
PTR_TO_LST_OF_S	D	FE6E	EOSCMN_06:N_EOS
PTR_TO_S_ON_0	D	FE70	EOSCMN_06:N_EOS
PTR_TO_S_ON_1	D	FE72	EOSCMN_06:N_EOS
PTR_TO_S_ON_2	D	FE74	EOSCMN_06:N_EOS
PTR_TO_S_ON_3	D	FE76	EOSCMN_06:N_EOS
PUT_ASCII	P	FD17	EOSCMN_06:N_EOS
PUT_VRAM	P	FD2C	EOSCMN_06:N_EOS
PX_TO_PTRN_POS	P	FD35	EOSCMN_06:N_EOS
QUERY_BUFFER	D	FDA0	EOSCMN_06:N_EOS
READ_BLOCK	P	F17B	UTILS:N_EOS
READ_REGISTER	P	FD23	EOSCMN_06:N_EOS
READ_VRAM	P	FD1D	EOSCMN_06:N_EOS
RETRY_COUNT	D	FDD6	EOSCMN_06:N_EOS
REV_NUM	D	FD60	EOSCMN_06:N_EOS
SAVE_CTRL	D	FE78	EOSCMN_06:N_EOS
SCAN_FOR_FILE	P	EF08	UTILS:N_EOS
SECTORS_TO_INIT	D	FD86	EOSCMN_06:N_EOS
SECTOR_NO	D	FD87	EOSCMN_06:N_EOS
SOUNDPORT	A	FC2F	EOSCMN_06:N_EOS
SOUNDS	P	FD59	EOSCMN_06:N_EOS
SOUND_INIT	P	FD50	EOSCMN_06:N_EOS
SPIN_SW0_CT	D	FE58	EOSCMN_06:N_EOS
SPIN_SW1_CT	D	FE59	EOSCMN_06:N_EOS
SPRITEATTRTBL	D	FD64	EOSCMN_06:N_EOS
SPRITEGENTBL	D	FD66	EOSCMN_06:N_EOS

A_uOS_00:N_EOS
RD_WT:N_EOS OPENS:N_EOS
UTILS:N_EOS

EOS_UTIL:N_EOS
UTILS:N_EOS DIR_HANDL:N_EOS
DIR_HANDL:N_EOS
CONS_OUT:N_EOS
CONS_OUT:N_EOS
CONS_OUT:N_EOS
A_uOS_00:N_EOS

EOS_UTIL:N_EOS
A_uOS_00:N_EOS

EOS_START:N_EOS
A_uOS_00:N_EOS
EOS_PRTR:N_EOS
CONS_OUT:N_EOS
A_uOS_00:N_EOS
A_uOS_00:N_EOS
A_uOS_00:N_EOS
A_uOS_00:N_EOS
A_uOS_00:N_EOS

UTILS:N_EOS
RD_WT:N_EOS OPENS:N_EOS DIR_HANDL:N_EOS

CONS_OUT:N_EOS A_uOS_00:N_EOS
UTILS:N_EOS
EOS_START:N_EOS
A_uOS_00:N_EOS
DIR_HANDL:N_EOS
EOS_MKDIR:N_EOS
EOS_MKDIR:N_EOS
A_uOS_00:N_EOS

A_uOS_00:N_EOS

A_uOS_00:N_EOS

START_BLOCK	D	FE12	EOSCMN_06:N_EOS	
STRCMP	P	F053	UTILS:N_EOS	DIR_HANDL:N_EOS
STROBE_RESET_PO	A	!C2E	EOSCMN_06:N_EOS	
STROBE_SET_PORT	A	FC2D	EOSCMN_06:N_EOS	A_uOS_00:N_EOS
SWITCH_MEM	P	FD14	EOSCMN_06:N_EOS	EOS_UTIL:N_EOS EOS_START:N_EOS A_uOS_00:N_EOS
SWITCH_TABLE	A	FC17	EOSCMN_06:N_EOS	
TEMP_STACK	D	FE6E	EOSCMN_06:N_EOS	A_uOS_00:N_EOS
TURN_OFF_SOUND	P	FD53	EOSCMN_06:N_EOS	EOS_START:N_EOS
UPDATE_SPINNER	P	FD41	EOSCMN_06:N_EOS	
UPPER_LEFT	D	FEA1	EOSCMN_06:N_EOS	CONS_OUT:N_EOS
USER_BUF	D	FE06	EOSCMN_06:N_EOS	RD_WT:N_EOS DIR_HANDL:N_EOS
USER_NAME	D	FE10	EOSCMN_06:N_EOS	UTILS:N_EOS DIR_HANDL:N_EOS
VDP_CTRL_PORT	A	FC29	EOSCMN_06:N_EOS	A_uOS_00:N_EOS
VDP_DATA_PORT	A	FC2A	EOSCMN_06:N_EOS	
VDP_MODE_WORD	D	FD61	EOSCMN_06:N_EOS	A_uOS_00:N_EOS
VDP_STATUS_BYTE	D	FD63	EOSCMN_06:N_EOS	A_uOS_00:N_EOS
VECTOR_08H	A	FBFF	EOSCMN_06:N_EOS	
VECTOR_10H	A	FC02	EOSCMN_06:N_EOS	
VECTOR_18H	A	FC05	EOSCMN_06:N_EOS	
VECTOR_20H	A	FC08	EOSCMN_06:N_EOS	
VECTOR_28H	A	FC0B	EOSCMN_06:N_EOS	
VECTOR_30H	A	FC0E	EOSCMN_06:N_EOS	
VECTOR_38H	A	FC11	EOSCMN_06:N_EOS	
VECTOR_66H	A	FC14	EOSCMN_06:N_EOS	
VOL_BLK_SZ	D	FDDC	EOSCMN_06:N_EOS	DIR_HANDL:N_EOS
VRAM_ADDR_TABLE	D	FD64	EOSCMN_06:N_EOS	A_uOS_00:N_EOS
WRITE_BLOCK	P	F1E6	UTILS:N_EOS	EOS_MKDIR:N_EOS RD_WT:N_EOS OPENS:N_EOS DIR_HANDL:N_EOS
WRITE_REGISTER	P	FD20	EOSCMN_06:N_EOS	A_uOS_00:N_EOS
WRITE_VRAM	P	FD1A	EOSCMN_06:N_EOS	CONS_OUT:N_EOS A_uOS_00:N_EOS
WR_SPR_ATTRIBUT	P	FD3B	EOSCMN_06:N_EOS	
X_MAX	D	FE7B	EOSCMN_06:N_EOS	CONS_OUT:N_EOS
X_MIN	D	FE7A	EOSCMN_06:N_EOS	CONS_OUT:N_EOS
Y_MAX	D	FE7D	EOSCMN_06:N_EOS	CONS_OUT:N_EOS
Y_MIN	D	FE7C	EOSCMN_06:N_EOS	CONS_OUT:N_EOS
_CHECK_FCB	P	FCF0	EOSCMN_06:N_EOS	
_CLOSE_FILE	P	FCC3	EOSCMN_06:N_EOS	
_CONS_DISP	P	FC33	EOSCMN_06:N_EOS	
_CONS_INIT	P	FC36	EOSCMN_06:N_EOS	
_CONS_OUT	P	FC39	EOSCMN_06:N_EOS	
_CV_A	P	FD0E	EOSCMN_06:N_EOS	
_DELETE_FILE	P	FCE1	EOSCMN_06:N_EOS	
_DLY_AFT_HRD_RE	P	FC3C	EOSCMN_06:N_EOS	
_END_PR_BUFF	P	FC3F	EOSCMN_06:N_EOS	
_END_PR_CH	P	FC42	EOSCMN_06:N_EOS	
_END_RD_1_BLOCK	P	FC45	EOSCMN_06:N_EOS	
_END_RD_CH_DEV	P	FC48	EOSCMN_06:N_EOS	
_END_RD_KBD	P	FC4B	EOSCMN_06:N_EOS	
_END_WR_1_BLOCK	P	FC4E	EOSCMN_06:N_EOS	
_END_WR_CH_DEV	P	FC51	EOSCMN_06:N_EOS	
_EOS_1	P	FD05	EOSCMN_06:N_EOS	
_EOS_2	P	FD08	EOSCMN_06:N_EOS	
_EOS_3	P	FD0B	EOSCMN_06:N_EOS	
_EOS_START	P	FC30	EOSCMN_06:N_EOS	
_FILE_QUERY	P	FCFF	EOSCMN_06:N_EOS	
_FIND_DCB	P	FC54	EOSCMN_06:N_EOS	
_FMGR_INIT	P	FCBA	EOSCMN_06:N_EOS	
_GET_DATE	P	FCDB	EOSCMN_06:N_EOS	

_GET_DCB_ADDR	P	FC57	EOSCMN_06:N_EOS	
_GET_PCB_ADDR	P	FC5A	EOSCMN_06:N_EOS	
_GOTO_WP	P	FCE7	EOSCMN_06:N_EOS	EOS_START:N_EOS
_HARD_INIT	P	FC5D	EOSCMN_06:N_EOS	
_HARD_RESET_NET	P	FC60	EOSCMN_06:N_EOS	
_INIT_TAPE_DIR	P	FCBD	EOSCMN_06:N_EOS	
_MAKE_FILE	P	FCC9	EOSCMN_06:N_EOS	
_MODE_CHECK	P	FCF9	EOSCMN_06:N_EOS	
_OPEN_FILE	P	FCC0	EOSCMN_06:N_EOS	
_POSIT_FILE	P	FD02	EOSCMN_06:N_EOS	
_PR_BUFF	P	FC63	EOSCMN_06:N_EOS	
_PR_CH	P	FC66	EOSCMN_06:N_EOS	
_QUERY_FILE	P	FCCC	EOSCMN_06:N_EOS	
_RD_1_BLOCK	P	FC69	EOSCMN_06:N_EOS	
_RD_DEV_DEP_STA	P	FCE4	EOSCMN_06:N_EOS	
_RD_KBD	P	FC6C	EOSCMN_06:N_EOS	
_RD_KBD_RET_COD	P	FC6F	EOSCMN_06:N_EOS	
_RD_PR_RET_CODE	P	FC72	EOSCMN_06:N_EOS	
_RD_RET_CODE	P	FC75	EOSCMN_06:N_EOS	
_RD_TAPE_RET_CO	P	FC78	EOSCMN_06:N_EOS	
_READ_BLOCK	P	FCF3	EOSCMN_06:N_EOS	
_READ_EOS	P	FCEA	EOSCMN_06:N_EOS	
_READ_FILE	P	FCD2	EOSCMN_06:N_EOS	
_RELOC_PCB	P	FC7B	EOSCMN_06:N_EOS	
_RENAME_FILE	P	FCDE	EOSCMN_06:N_EOS	
_REQUEST_STATUS	P	FC7E	EOSCMN_06:N_EOS	
_REQ_KBD_STAT	P	FC81	EOSCMN_06:N_EOS	
_REQ_PR_STAT	P	FC84	EOSCMN_06:N_EOS	
_REQ_TAPE_STAT	P	FC87	EOSCMN_06:N_EOS	
_RESET_FILE	P	FCC6	EOSCMN_06:N_EOS	
_SCAN_ACTIVE	P	FC8A	EOSCMN_06:N_EOS	
_SCAN_FOR_FILE	P	FCFC	EOSCMN_06:N_EOS	
_SET_DATE	P	FCD8	EOSCMN_06:N_EOS	
_SET_FILE	P	FCCF	EOSCMN_06:N_EOS	
_SOFT_INIT	P	FC8D	EOSCMN_06:N_EOS	
_SOFT_RES_DEV	P	FC90	EOSCMN_06:N_EOS	
_SOFT_RES_KBD	P	FC93	EOSCMN_06:N_EOS	
_SOFT_RES_PR	P	FC96	EOSCMN_06:N_EOS	
_SOFT_RES_TAPE	P	FC99	EOSCMN_06:N_EOS	
_START_PR_BUFF	P	FC9C	EOSCMN_06:N_EOS	
_START_PR_CH	P	FC9F	EOSCMN_06:N_EOS	
_START_RD_1_BLO	P	FCA2	EOSCMN_06:N_EOS	
_START_RD_CH_DE	P	FCA5	EOSCMN_06:N_EOS	
_START_RD_KBD	P	FCA8	EOSCMN_06:N_EOS	
_START_WR_1_BLO	P	FCAB	EOSCMN_06:N_EOS	
_START_WR_CH_DE	P	FCAE	EOSCMN_06:N_EOS	
_SYNC	P	FCB1	EOSCMN_06:N_EOS	
_TRIM_FILE	P	FCED	EOSCMN_06:N_EOS	
_WRITE_BLOCK	P	FCF6	EOSCMN_06:N_EOS	
_WRITE_FILE	P	FCD5	EOSCMN_06:N_EOS	
_WR_1_BLOCK	P	FCB4	EOSCMN_06:N_EOS	
_WR_CH_DEV	P	FCB7	EOSCMN_06:N_EOS	
__ADD816	P	E374	A_uOS_00:N_EOS	EOSCMN_06:N_EOS
__CALC_OFFSET	P	E10A	A_uOS_00:N_EOS	EOSCMN_06:N_EOS
__CHECK_FCB	P	F089	UTILS:N_EOS	EOSCMN_06:N_EOS
__CLOSE_FILE	P	EB04	OPENS:N_EOS	EOSCMN_06:N_EOS
__CONS_DISP	P	F627	CONS_OUT:N_EOS	EOSCMN_06:N_EOS

```
__CONS_INIT      P F5DC    CONS_OUT:N_EOS          EOSCMN_06:N_EOS
__CONS_OUT       P F60A    CONS_OUT:N_EOS          EOSCMN_06:N_EOS
__CV_A           P F442    EOS_RET:N_EOS           EOSCMN_06:N_EOS
__DECLSN         P E355    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__DECMSN         P E35F    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__DECODER        P E206    A_uOS_00:N_EOS
__DELETE_FILE    P F14E    UTILS:N_EOS             EOSCMN_06:N_EOS
__DLY_AFT_HRD_R  P F95F    EOS_UTIL:N_EOS          EOSCMN_06:N_EOS
__EFFECT_OVER    P E4B8    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__END_PR_BUFF    P F5B8    EOS_PRTR:N_EOS          EOSCMN_06:N_EOS
__END_PR_CH      P F57C    EOS_PRTR:N_EOS          EOSCMN_06:N_EOS
__END_RD_1_BLOC  P FAE2    EOS_BLK:N_EOS           EOSCMN_06:N_EOS
__END_RD_CH_DEV  P FBA5    EOS_CHAR:N_EOS          EOSCMN_06:N_EOS  EOS_KBD:N_EOS
__END_RD_KBD     P F4E0    EOS_KBD:N_EOS           EOSCMN_06:N_EOS
__END_WR_1_BLOC  P FB1B    EOS_BLK:N_EOS           EOSCMN_06:N_EOS
__END_WR_CH_DEV  P FBE1    EOS_CHAR:N_EOS          EOSCMN_06:N_EOS  EOS_PRTR:N_EOS
__EOS_1          P F442    EOS_RET:N_EOS           EOSCMN_06:N_EOS
__EOS_2          P F442    EOS_RET:N_EOS           EOSCMN_06:N_EOS
__EOS_3          P F442    EOS_RET:N_EOS           EOSCMN_06:N_EOS
__EOS_START      P F832    EOS_START:N_EOS         EOSCMN_06:N_EOS
__FILE_QUERY     P E618    DIR_HANDL:N_EOS         EOSCMN_06:N_EOS
__FILL_VRAM      P E059    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__FIND_DCB       P F446    EOS_GEN:N_EOS           EOSCMN_06:N_EOS  EOS_CHAR:N_EOS  EOS_BLK:N_EOS  EOS_UTIL:N_EOS
                                                   EOS_START:N_EOS  EOS_PRTR:N_EOS
__FMGR_INIT      P EEEA    UTILS:N_EOS             EOSCMN_06:N_EOS  EOS_START:N_EOS
__GET_DATE       P EED4    DATES:N_EOS             EOSCMN_06:N_EOS
__GET_DCB_ADDR   P F446    EOS_GEN:N_EOS           EOSCMN_06:N_EOS
__GET_PCB_ADDR   P FA4C    EOS_UTIL:N_EOS          EOSCMN_06:N_EOS
__GET_VRAM       P E0CF    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__GOTO_WP        P FA94    EOS_UTIL:N_EOS          EOSCMN_06:N_EOS
__HARD_INIT      P F8F6    EOS_UTIL:N_EOS          EOSCMN_06:N_EOS  EOS_START:N_EOS
__HARD_RESET_NE  P F94B    EOS_UTIL:N_EOS          EOSCMN_06:N_EOS
__INIT_TABLE     P E066    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__INIT_TAPE_DIR  P F323    EOS_MKDIR:N_EOS         EOSCMN_06:N_EOS
__LOAD_ASCII     P E149    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__MAKE_FILE      P E690    DIR_HANDL:N_EOS         EOSCMN_06:N_EOS
__MODE_CHECK     P F0D9    UTILS:N_EOS             EOSCMN_06:N_EOS
__MSNTOLSN       P E369    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__OPEN_FILE      P EA00    OPENS:N_EOS             EOSCMN_06:N_EOS
__PLAY_IT        P E3E7    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__POLLER         P E253    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__PORT_COLLECTI  P E191    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__POSIT_FILE     P F442    EOS_RET:N_EOS           EOSCMN_06:N_EOS
__PR_BUFF        P F515    EOS_PRTR:N_EOS          EOSCMN_06:N_EOS
__PR_CH          P F4FC    EOS_PRTR:N_EOS          EOSCMN_06:N_EOS
__PUT_ASCII      P E153    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__PUT_VRAM       P E0C9    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__PX_TO_PTRN_PO  P E129    A_uOS_00:N_EOS          EOSCMN_06:N_EOS
__QUERY_FILE     P E61B    DIR_HANDL:N_EOS         EOSCMN_06:N_EOS  UTILS:N_EOS  OPENS:N_EOS
__RD_1_BLOCK     P FA9E    EOS_BLK:N_EOS           EOSCMN_06:N_EOS  EOS_START:N_EOS  UTILS:N_EOS
__RD_CH_DEV      P FB64    EOS_CHAR:N_EOS          EOS_KBD:N_EOS
__RD_DEV_DEP_ST  P F488    EOS_GEN:N_EOS           EOSCMN_06:N_EOS
__RD_KBD         P F4BA    EOS_KBD:N_EOS           EOSCMN_06:N_EOS
__RD_KBD_RET_CO  P FA7B    EOS_UTIL:N_EOS          EOSCMN_06:N_EOS
__RD_PR_RET_COD  P FA7F    EOS_UTIL:N_EOS          EOSCMN_06:N_EOS
__RD_RET_CODE    P FA87    EOS_UTIL:N_EOS          EOSCMN_06:N_EOS
__RD_TAPE_RET_C  P FA83    EOS_UTIL:N_EOS          EOSCMN_06:N_EOS
```

__READ_BLOCK	P	F17B	UTILS:N_EOS	EOSCMN_06:N_EOS
__READ_EOS	P	FA9D	EOS_UTIL:N_EOS	EOSCMN_06:N_EOS
__READ_FILE	P	EC17	RD_WT:N_EOS	EOSCMN_06:N_EOS
__READ_REGISTER	P	E04F	A_uOS_OO:N_EOS	EOSCMN_06:N_EOS
__READ_VRAM	P	E01A	A_uOS_OO:N_EOS	EOSCMN_06:N_EOS
__RELOC_PCB	P	FA2F	EOS_UTIL:N_EOS	EOSCMN_06:N_EOS
__RENAME_FILE	P	F10F	UTILS:N_EOS	EOSCMN_06:N_EOS
__REQUEST_STATU	P	F473	EOS_GEN:N_EOS	EOSCMN_06:N_EOS EOS_START:N_EOS EOS_TAPE:N_EOS EOS_PRTR:N_EOS EOS_KBD:N_EOS UTILS:N_EOS
__REQ_KBD_STAT	P	F4CB	EOS_KBD:N_EOS	EOSCMN_06:N_EOS
__REQ_PR_STAT	P	F5D2	EOS_PRTR:N_EOS	EOSCMN_06:N_EOS
__REQ_TAPE_STAT	P	F5D7	EOS_TAPE:N_EOS	EOSCMN_06:N_EOS
__RESET_FILE	P	EB6C	OPENS:N_EOS	EOSCMN_06:N_EOS
__SCAN_ACTIVE	P	F9CB	EOS_UTIL:N_EOS	EOSCMN_06:N_EOS
__SCAN_FOR_FILE	P	EF0B	UTILS:N_EOS	EOSCMN_06:N_EOS
__SETUP_PR_BUFF	P	F55E	EOS_PRTR:N_EOS	
__SET_DATE	P	EEC5	DATES:N_EOS	EOSCMN_06:N_EOS
__SET_FILE	P	E651	DIR_HANDL:N_EOS	EOSCMN_06:N_EOS UTILS:N_EOS OPENS:N_EOS
__SOFT_INIT	P	F922	EOS_UTIL:N_EOS	EOSCMN_06:N_EOS
__SOFT_RES_DEV	P	FA5D	EOS_UTIL:N_EOS	EOSCMN_06:N_EOS
__SOFT_RES_KBD	P	FA51	EOS_UTIL:N_EOS	EOSCMN_06:N_EOS
__SOFT_RES_PR	P	FA55	EOS_UTIL:N_EOS	EOSCMN_06:N_EOS
__SOFT_RES_TAPE	P	FA59	EOS_UTIL:N_EOS	EOSCMN_06:N_EOS
__SOUNDS	P	E406	A_uOS_OO:N_EOS	EOSCMN_06:N_EOS
__SOUND_INIT	P	E3AB	A_uOS_OO:N_EOS	EOSCMN_06:N_EOS
__START_PR_BUFF	P	F580	EOS_PRTR:N_EOS	EOSCMN_06:N_EOS
__START_PR_CH	P	F56D	EOS_PRTR:N_EOS	EOSCMN_06:N_EOS
__START_RD_1_BL	P	FAC6	EOS_BLK:N_EOS	EOSCMN_06:N_EOS
__START_RD_CH_D	P	FB86	EOS_CHAR:N_EOS	EOSCMN_06:N_EOS EOS_KBD:N_EOS
__START_RD_KBD	P	F4D0	EOS_KBD:N_EOS	EOSCMN_06:N_EOS
__START_WR_1_BL	P	FAFF	EOS_BLK:N_EOS	EOSCMN_06:N_EOS
__START_WR_CH_D	P	FBC2	EOS_CHAR:N_EOS	EOSCMN_06:N_EOS EOS_PRTR:N_EOS
__SWITCH_MEM	P	E185	A_uOS_OO:N_EOS	EOSCMN_06:N_EOS
__SYNC	P	F970	EOS_UTIL:N_EOS	EOSCMN_06:N_EOS
__TRIM_FILE	P	F241	UTILS:N_EOS	EOSCMN_06:N_EOS
__TURN_OFF_SOUN	P	E3D1	A_uOS_OO:N_EOS	EOSCMN_06:N_EOS
__UPDATE_SPINNE	P	E2A4	A_uOS_OO:N_EOS	EOSCMN_06:N_EOS
__WRITE_BLOCK	P	F1E6	UTILS:N_EOS	EOSCMN_06:N_EOS
__WRITE_FILE	P	ED8F	RD_WT:N_EOS	EOSCMN_06:N_EOS
__WRITE_REGISTE	P	E034	A_uOS_OO:N_EOS	EOSCMN_06:N_EOS
__WRITE_VRAM	P	E000	A_uOS_OO:N_EOS	EOSCMN_06:N_EOS
__WR_1_BLOCK	P	FAB2	EOS_BLK:N_EOS	EOSCMN_06:N_EOS UTILS:N_EOS
__WR_CH_DEV	P	FB75	EOS_CHAR:N_EOS	EOSCMN_06:N_EOS EOS_PRTR:N_EOS
__WR_SPR_ATTRIB	P	E1C5	A_uOS_OO:N_EOS	EOSCMN_06:N_EOS

SECTION V

OS 7 ABSOLUTE LISTINGS

APPENDIX E
JUMP TABLE

PLAY_SONGS	1F61
ACTIVATEP	1F64
PUTOBJP	1F67
REFLECT_VERTICAL	1F6A
REFLECT_HORIZONTAL	1F6D
ROTATE_90	1F70
ENLARGE	1F73
CONTROLLER_SCAN	1F76
DECODER	1F79
GAME_OPT	1F7C
LOAD_ASCII	1F7F
FILL_VRAM	1F82
MODE_1	1F85
UPDATE_SPINNER	1F88
INIT_TABLEP	1F8B
GET_VRAMP	1F8E
PUT_VRAMP	1F91
INIT_SPR_ORDERP	1F94
WR_SPR_NM_TBLP	1F97
INIT_TIMERP	1F9A
FREE_SIGNALP	1F9D
REQUEST_SIGNALP	1FA0
TEST_SIGNALP	1FA3
WRITE_REGISTERP	1FA6
WRITE_VRAMP	1FA9
READ_VRAMP	1FAC
INIT_WRITERP	1FAF
SOUND_INITP	1FB2
PLAY_ITP	1FB5
INIT_TABLE	1FB8
GET_VRAM	1FBB
PUT_VRAM	1FBE
INIT_SPR_ORDER	1FC1
WR_SPR_NM_TBL	1FC4
INIT_TIMER	1FC7
FREE_SIGNAL	1FCA
REQUEST_SIGNAL	1FCD
TEST_SIGNAL	1FD0
TIME_MGR	1FD3
TURN_OFF_SOUND	1FD6
WRITE_REGISTER	1FD9
READ_REGISTER	1FDC
WRITE_VRAM	1FDF
READ_VRAM	1FE2
INIT_WRITER	1FE5
WRITER	1FE8
POLLER	1FEB
SOUND_INIT	1FEE
PLAY_IT	1FF1
SOUND_MAN	1FF4
ACTIVATE	1FF7
PUTOBJ	1FFA
RAND_GEN	1FFD

APPENDIX F

OS SYMBOLS

```
ACTIVATE        EQU 01FF7H      ;       OS:OS
ACTIVATEP       EQU 01F64H      ;       OS:OS
ADD816          EQU 001B1H      ;       OS:OS
AMERICA         EQU 00069H      ;       OS:OS
ASCII_TABLE     EQU 0006AH      ;       OS:OS
ATN_SWEEP       EQU 0012FH      ;       OS:OS
CARTRIDGE       EQU 08000H      ;       OS:OS
CONTROLLER_MAP  EQU 08008H      ;       OS:OS
CTRL_PORT_PTR   EQU 01D43H
DATA_PORT_PTR   EQU 01D47H
DECLSN          EQU 00190H      ;       OS:OS
DECMSN          EQU 0019BH      ;       OS:OS
DECODER         EQU 01F79H      ;       OS:OS
DEFER_WRITES    EQU 073C6H      ;       OS:OS
EFXOVER         EQU 002EEH      ;       OS:OS
ENLARGE         EQU 01F73H      ;       OS:OS
ENLRG           EQU 01D6CH      ;       OS:OS
FILL_VRAM       EQU 01F82H      ;       OS:OS
FREE_SIGNAL     EQU 01FCAH      ;       OS:OS
FREE_SIGNALP    EQU 01F9DH      ;       OS:OS
FREQ_SWEEP      EQU 000FCH      ;       OS:OS
GAME_NAME       EQU 08024H      ;       OS:OS
GAME_OPT        EQU 01F7CH      ;       OS:OS
GET_VRAM        EQU 01FBBH      ;       OS:OS
GET_VRAMP       EQU 01F8EH      ;       OS:OS
INIT_SPR_ORDER  EQU 01FC1H      ;       OS:OS
INIT_SPR_ORDERP EQU 01F94H      ;       OS:OS
INIT_TABLE      EQU 01FB8H      ;       OS:OS
INIT_TABLEP     EQU 01F8BH      ;       OS:OS
INIT_TIMER      EQU 01FC7H      ;       OS:OS
INIT_TIMERP     EQU 01F9AH      ;       OS:OS
INIT_WRITER     EQU 01FE5H      ;       OS:OS
INIT_WRITERP    EQU 01FAFH      ;       OS:OS
IRQ_INT_VECT    EQU 0801EH      ;       OS:OS
LEAVE_EFFECT    EQU 001D5H      ;       OS:OS
LOAD_ASCII      EQU 01F7FH      ;       OS:OS
LOCAL_SPR_TBL   EQU 08002H      ;       OS:OS
MODE_1          EQU 01F85H      ;       OS:OS
MSNTOLSN        EQU 001A6H      ;       OS:OS
MUX_SPRITES     EQU 073C7H      ;       OS:OS
NMI_INT_VECT    EQU 08021H      ;       OS:OS
NUMBER_TABLE    EQU 0006CH      ;       OS:OS
PLAY_IT         EQU 01FF1H      ;       OS:OS
PLAY_ITP        EQU 01FB5H      ;       OS:OS
PLAY_SONGS      EQU 01F61H      ;       OS:OS
```

```
POLLER              EQU 01FEBH      ;     OS:OS
PUTOBJ              EQU 01FFAH      ;     OS:OS
PUTOBJP             EQU 01F67H      ;     OS:OS
PUT_VRAM            EQU 01FBEH      ;     OS:OS
PUT_VRAMP           EQU 01F91H      ;     OS:OS
RAND_GEN            EQU 01FFDH      ;     OS:OS
RAND_NUM            EQU 073C8H      ;     OS:OS
READ_REGISTER       EQU 01FDCH      ;     OS:OS
READ_VRAM           EQU 01FE2H      ;     OS:OS
READ_VRAMP          EQU 01FACH      ;     OS:OS
REFLECT_HORIZON     EQU 01F6DH      ;     OS:OS
REFLECT_VERTICA     EQU 01F6AH      ;     OS:OS
REQUEST_SIGNAL      EQU 01FCDH      ;     OS:OS
REQUEST_SIGNALP     EQU 01FA0H      ;     OS:OS
ROTATE_90           EQU 01F70H      ;     OS:OS
RST_10H_RAM         EQU 0800FH      ;     OS:OS
RST_18H_RAM         EQU 08012H      ;     OS:OS
RST_20H_RAM         EQU 08015H      ;     OS:OS
RST_28H_RAM         EQU 08018H      ;     OS:OS
RST_30H_RAM         EQU 0801BH      ;     OS:OS
RST_8H_RAM          EQU 0800CH      ;     OS:OS
SOUND_INIT          EQU 01FEEH      ;     OS:OS
SOUND_INITP         EQU 01FB2H      ;     OS:OS
SOUND_MAN           EQU 01FF4H      ;     OS:OS
SPRITE_ORDER        EQU 08004H      ;     OS:OS
STACK               EQU 073B9H      ;     OS:OS
START_GAME          EQU 0800AH      ;     OS:OS
TEST_SIGNAL         EQU 01FD0H      ;     OS:OS
TEST_SIGNALP        EQU 01FA3H      ;     OS:OS
TIME_MGR            EQU 01FD3H      ;     OS:OS
TURN_OFF_SOUND      EQU 01FD6H      ;     OS:OS
UPDATE_SPINNER      EQU 01F88H      ;     OS:OS
VDP_MODE_WORD       EQU 073C3H      ;     OS:OS
VDP_STATUS_BYTE     EQU 073C5H      ;     OS:OS
WORK_BUFFER         EQU 08006H      ;     OS:OS
WRITER              EQU 01FEBH      ;     OS:OS
WRITE_REGISTER      EQU 01FD9H      ;     OS:OS
WRITE_REGISTERP     EQU 01FA6H      ;     OS:OS
WRITE_VRAM          EQU 01FDFH      ;     OS:OS
WRITE_VRAMP         EQU 01FA9H      ;     OS:OS
WR_SPR_NM_TBL       EQU 01FC4H      ;     OS:OS
WR_SPR_NM_TBLP      EQU 01F97H      ;     OS:OS
```

```
GLB ACTIVATE          ;     OS:OS
GLB ACTIVATEP         ;     OS:OS
GLB ADD816            ;     OS:OS
GLB AMERICA           ;     OS:OS
GLB ASCII_TABLE       ;     OS:OS
GLB ATN_SWEEP         ;     OS:OS
GLB CARTRIDGE         ;     OS:OS
GLB CONTROLLER_MAP    ;     OS:OS
GLB CTRL_PORT_PTR
GLB DATA_PORT_PTR
GLB DECLSN            ;     OS:OS
GLB DECMSN            ;     OS:OS
GLB DECODER           ;     OS:OS
GLB DEFER_WRITES      ;     OS:OS
GLB EFXOVER           ;     OS:OS
GLB ENLARGE           ;     OS:OS
GLB ENLRG             ;     OS:OS
GLB FILL_VRAM         ;     OS:OS
GLB FREE_SIGNAL       ;     OS:OS
GLB FREE_SIGNALP      ;     OS:OS
GLB FREQ_SWEEP        ;     OS:OS
GLB GAME_NAME         ;     OS:OS
GLB GAME_OPT          ;     OS:OS
GLB GET_VRAM          ;     OS:OS
GLB GET_VRAMP         ;     OS:OS
GLB INIT_SPR_ORDER    ;     OS:OS
GLB INIT_SPR_ORDERP   ;     OS:OS
GLB INIT_TABLE        ;     OS:OS
GLB INIT_TABLEP       ;     OS:OS
GLB INIT_TIMER        ;     OS:OS
GLB INIT_TIMERP       ;     OS:OS
GLB INIT_WRITER       ;     OS:OS
GLB INIT_WRITERP      ;     OS:OS
GLB IRQ_INT_VECT      ;     OS:OS
GLB LEAVE_EFFECT      ;     OS:OS
GLB LOAD_ASCII        ;     OS:OS
GLB LOCAL_SPR_TBL     ;     OS:OS
GLB MSNTOLSN          ;     OS:OS
GLB MODE_1            ;     OS:OS
GLB MUX_SPRITES       ;     OS:OS
GLB NMI_INT_VECT      ;     OS:OS
GLB NUMBER_TABLE      ;     OS:OS
GLB PLAY_IT           ;     OS:OS
GLB PLAY_ITP          ;     OS:OS
GLB PLAY_SONGS        ;     OS:OS
GLB POLLER            ;     OS:OS
GLB PUTOBJ            ;     OS:OS
GLB PUTOBJP           ;     OS:OS
```

```
        GLB PUT_VRAM            ;       OS:OS
        GLB PUT_VRAMP           ;       OS:OS
        GLB RAND_GEN            ;       OS:OS
        GLB RAND_NUM            ;       OS:OS
        GLB READ_REGISTER       ;       OS:OS
        GLB READ_VRAM           ;       OS:OS
        GLB READ_VRAMP          ;       OS:OS
        GLB REFLECT_HORIZON     ;       OS:OS
        GLB REFLECT_VERTICA     ;       OS:OS
        GLB REQUEST_SIGNAL      ;       OS:OS
        GLB REQUEST_SIGNALP     ;       OS:OS
        GLB ROTATE_90           ;       OS:OS
        GLB RST_10H_RAM         ;       OS:OS
        GLB RST_18H_RAM         ;       OS:OS
        GLB RST_20H_RAM         ;       OS:OS
        GLB RST_28H_RAM         ;       OS:OS
        GLB RST_30H_RAM         ;       OS:OS
        GLB RST_8H_RAM          ;       OS:OS
        GLB SOUND_INIT          ;       OS:OS
        GLB SOUND_INITP         ;       OS:OS
        GLB SOUND_MAN           ;       OS:OS
        GLB SPRITE_ORDER        ;       OS:OS
        GLB STACK               ;       OS:OS
        GLB START_GAME          ;       OS:OS
        GLB TEST_SIGNAL         ;       OS:OS
        GLB TEST_SIGNALP        ;       OS:OS
        GLB TIME_MGR            ;       OS:OS
        GLB TURN_OFF_SOUND      ;       OS:OS
        GLB UPDATE_SPINNER      ;       OS:OS
        GLB VDP_MODE_WORD       ;       OS:OS
        GLB VDP_STATUS_BYTE     ;       OS:OS
        GLB WORK_BUFFER         ;       OS:OS
        GLB WRITER              ;       OS:OS
        GLB WRITE_REGISTER      ;       OS:OS
        GLB WRITE_REGISTERP     ;       OS:OS
        GLB WRITE_VRAM          ;       OS:OS
        GLB WRITE_VRAMP         ;       OS:OS
        GLB WR_SPR_NM_TBL       ;       OS:OS
        GLB WR_SPR_NM_TBLP      ;       OS:OS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                        1 "Z80"
                        3  NAME ^OS_7PRIME^
                        4
                        5 DESCRIPTION MACRO
                        6         .GOTO ENDESCRIPTION
                        7
                        8  Author:          Coleco Industries Inc.
                        9                   Advanced Research & Development - Software Engineering
                       10  Userid:          OS
                       11  Starting date:   A long long time ago in a galaxy far far away . . .
                       12
                       13  Prom release Date: 24 Nov 1982. for internal use only
                       14  Prom release Rev: 7B
                       15
                       16  Prom release Date: December 28, 1982
                       17  Prom release Rev: 7PRIME
                       18
                       19  Header Rev: 2
                       20
                       21  ********************************************************************************
                       22  *                                                                              *
                       23  *                    ColecoVision Operating System                             *
                       24  *                   Absolute Listing ( Rev 7PRIME )                            *
                       25  *                     (c) Coleco Industries 1982                               *
                       26  *                                                                              *
                       27  *                        *** Confidential ***                                  *
                       28  *                                                                              *
                       29  ********************************************************************************
                       30
                       31
                       32        This listing has the actual addresses of the start of OS routines
                       33
                       34        Rev History (one line note indicating the change)
                       35
                       36        Rev.   Date                  Change
                       37         4     14feb1983            Filler locations changed to 0FFH to
                       38                                    reflect OS_7PRIME. Prom release date
                       39                                    changed to December 28, 1982 from May
                       40                                    1982.  Name change to OS_7PRIME to
                       41                                    reflect majority of versions in the
                       42                                    field at this date.
                       43
                       44         3     24nov1982            Timing change to shorten LOGO delay
                       45                                    Title changes to JMPTABLES and OSSR_EQU
                       46         2     6oct1982             Minor comment modifications
                       47         1     23sept1982           OS_7 as one absolute file
                       48         0     May 1982             OS_7 listing by module
                       49
                       50 ENDESCRIPTION:
                       51         MEND
                       52  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      55 ;        Operating system sound routine EQUATES
                      56 ;        FILE NAME: OSSR.EQU
                      57 ;        *** Equates ***
                      58 ;        Dedicated Cartridge RAM locations
            <7020>    59 DEDAREA EQU 7020H        ;the start of the RAM area dedicated to sound routines
            <7020>    60 PTR_TO_LST_OF_SND_ADDRS EQU DEDAREA+0
            <7022>    61 PTR_TO_S_ON_0 EQU DEDAREA+2
            <7024>    62 PTR_TO_S_ON_1 EQU DEDAREA+4
            <7026>    63 PTR_TO_S_ON_2 EQU DEDAREA+6
            <7028>    64 PTR_TO_S_ON_3 EQU DEDAREA+8
            <702A>    65 SAVE_CTRL EQU DEDAREA+10
                      66 ;        Attenuation level codes
            <000F>    67 OFF      EQU 0FH          ;OFF [NO SOUND]
                      68 ;        Sound output port
            <00FF>    69 SOUND_PORT EQU 0FFH     ;data to sound chip thru this port
                      70 ;        Special byte 0 codes
            <00FF>    71 INACTIVE EQU 0FFH
            <003E>    72 SEFFECT EQU 62
            <0000>    73 ENDSDATA EQU 0
                      74 ;        Offsets within an SxDATA song data area
            <0000>    75 CH       EQU 0
            <0000>    76 SONGNO   EQU 0
            <0001>    77 NEXTNOTEPTR EQU 1
            <0003>    78 FREQ     EQU 3
            <0004>    79 ATN      EQU 4
            <0004>    80 CTRL     EQU 4
            <0005>    81 NLEN     EQU 5
            <0006>    82 FPS      EQU 6
            <0006>    83 FPSV     EQU 6
            <0007>    84 FSTEP    EQU 7
            <0008>    85 ALEN     EQU 8
            <0008>    86 ASTEP    EQU 8
            <0009>    87 APS      EQU 9
            <0009>    88 APSV     EQU 9
                      89 ;        song end codes
            <0010>    90 CH0END   EQU 00010000B
            <0050>    91 CH1END   EQU 01010000B
            <0090>    92 CH2END   EQU 10010000B
            <00D0>    93 CH3END   EQU 11010000B
            <0018>    94 CH0REP   EQU 00011000B
            <0058>    95 CH1REP   EQU 01011000B
            <0098>    96 CH2REP   EQU 10011000B
            <00D8>    97 CH3REP   EQU 11011000B
                      98 ;        channel numbers, B7 -B6
            <0000>    99 CH0      EQU 0
            <0040>   100 CH1      EQU 01000000B
            <0080>   101 CH2      EQU 10000000B
            <00C0>   102 CH3      EQU 11000000B
                     103 ;        [page]
                     104  PROG
                     105
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

                     107
                     108 ;              Modified February 14, 1983.  Filler areas were changed
                     109 ;              to 0FFH to reflect OS_7PRIME.  Also minor comment changes
                     110 ;              were made to clarify 055AAH for test cartridge condition.
                     111
                     112 ************************** EXTERNAL SYMBOLS ****************************
                     113
                     114 * EXTERNAL ROUTINES LINKED INTO OS
                     115
                     116                ;EXT REG_WRITE                  ;VIDEO DRIVERS
                     117                ;EXT REG_READ
                     118                ;EXT VRAM_WRITE
                     119                ;EXT VRAM_READ
                     120                ;EXT INIT_QUEUE
                     121                ;EXT WRITER_
                     122                ;EXT REG_WRITEQ                 ;PASCAL CALLS
                     123                ;EXT VRAM_WRITEQ
                     124                ;EXT VRAM_READQ
                     125                ;EXT INIT_QUEUEQ
                     126
                     127                ;EXT POLLER_                    ; CONTROLLER ROUTINE
                     128                ;EXT UPDATE_SPINNER_
                     129                ;EXT CONT_SCAN
                     130                ;EXT DECODER_
                     131
                     132                ;EXT INIT_SOUND                 ; SOUND ROUTINES
                     133                ;EXT ALL_OFF
                     134                ;EXT JUKE_BOX
                     135                ;EXT SND_MANAGER
                     136                ;EXT PLAY_SONGS_
                     137                ;EXT INIT_SOUNDQ                ; PASCAL CALLS
                     138                ;EXT JUKE_BOXQ
                     139
                     140                ;EXT INIT_TIMER_                ; TIME  MGMT ROUTINES
                     141                ;EXT FREE_SIGNAL_
                     142                ;EXT REQUEST_SIGNAL_
                     143                ;EXT TEST_SIGNAL_
                     144                ;EXT TIME_MGR_
                     145                ;EXT INIT_TIMERQ                ; PASCAL CALLS
                     146                ;EXT FREE_SIGNALQ
                     147                ;EXT REQUEST_SIGNALQ
                     148                ;EXT TEST_SIGNALQ
                     149
                     150                ;EXT INIT_TABLE_                ;TABLE MA
                     151                ;EXT GET_VRAM_
                     152                ;EXT PUT_VRAM_
                     153                ;EXT INIT_SPR_ORDER_
                     154                ;EXT WR_SPR_NM_TBL_
                     155                ;EXT INIT_TABLEQ                ;PASCAL CALLS
                     156                ;EXT GET_VRAMQ
                     157                ;EXT PUT_VRAMQ
                     158                ;EXT INIT_SPR_ORDERQ
                     159                ;EXT WR_SPR_NM_TBLQ
                     160
                     161                ;EXT ACTIVATE_                  ; GRAPHICS ROUTINES
                     162                ;EXT PUTOBJ_
                     163                ;EXT RFLCT_VERT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     164              ;EXT RFLCT_HOR
                     165              ;EXT ROT_90
                     166              ;EXT ENLRG
                     167              ;EXT PUTOBJQ                    ; PASCAL CALLS
                     168              ;EXT ACTIVATEQ
                     169
                     170              ;EXT GAME_OPT_                  ;DISPLAYS THE GAME OPTION SCREEN
                     171              ;EXT LOAD_ASCII_                ;LOADS ASCII CHARACTER GENERATORS
                     172              ;EXT FILL_VRAM_                 ;FILLS DESIGNATED AREA OF VRAM WITH VALUE
                     173              ;EXT MODE_1_                    ;SETS UP A DEFAULT GRAPHICS MODE 1
                     174
                     175
                     176 * "HIDDEN EXTERNALS"
                     177
                     178              ;EXT            DISPLAY_LOGO
                     179              ;EXT            CONTROLLER_INIT
                     180              ;EXT            ASCII_TBL
                     181              ;EXT            NUMBER_TBL
                     182
                     183
                     184 ******************************** EXPORTS ********************************
                     185
                     186 * ENTRY POINTS TO OS ROUTINES
                     187
                     188               GLB INIT_TABLE                 ;TABLE MA
                     189               GLB GET_VRAM
                     190               GLB PUT_VRAM
                     191               GLB INIT_SPR_ORDER
                     192               GLB WR_SPR_NM_TBL
                     193               GLB INIT_TABLEP                ;PASCAL CALLS
                     194               GLB GET_VRAMP
                     195               GLB PUT_VRAMP
                     196               GLB INIT_SPR_ORDERP
                     197               GLB WR_SPR_NM_TBLP
                     198
                     199               GLB WRITE_REGISTER             ; VIDEO DRIVERS
                     200               GLB READ_REGISTER
                     201               GLB WRITE_VRAM
                     202               GLB READ_VRAM
                     203               GLB INIT_WRITER
                     204               GLB WRITER
                     205               GLB WRITE_REGISTERP            ; PASCAL CALLS
                     206               GLB WRITE_VRAMP
                     207               GLB READ_VRAMP
                     208               GLB INIT_WRITERP
                     209
                     210               GLB POLLER                     ; CONTROLLER ROUTINES
                     211               GLB UPDATE_SPINNER
                     212               GLB CONTROLLER_SCAN
                     213               GLB DECODER
                     214
                     215               GLB SOUND_INIT                 ; SOUND ROUTINES
                     216               GLB TURN_OFF_SOUND
                     217               GLB PLAY_IT
                     218               GLB SOUND_MAN
                     219               GLB PLAY_SONGS
                     220               GLB SOUND_INITP                ; PASCAL CALLS
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      221               GLB PLAY_ITP
                      222
                      223               GLB INIT_TIMER                ; TIME MGMT ROUTINES
                      224               GLB FREE_SIGNAL
                      225               GLB REQUEST_SIGNAL
                      226               GLB TEST_SIGNAL
                      227               GLB TIME_MGR
                      228               GLB INIT_TIMERP               ; PASCAL CALLS
                      229               GLB FREE_SIGNALP
                      230               GLB REQUEST_SIGNALP
                      231               GLB TEST_SIGNALP
                      232
                      233               GLB STACK                     ; MISC GLOBALS
                      234               GLB VDP_STATUS_BYTE
                      235               GLB VDP_MODE_WORD
                      236               GLB AMERICA
                      237               GLB MUX_SPRITES
                      238               GLB DEFER_WRITES
                      239               GLB RAND_GEN                  ;Can be called from Pascal
                      240                                             ; or assembly language
                      241               GLB PUTOBJ                    ; GRAPHICS ROUTINES
                      242               GLB ACTIVATE
                      243               GLB REFLECT_VERTICAL
                      244               GLB REFLECT_HORIZONTAL
                      245               GLB ROTATE_90
                      246               GLB ENLARGE
                      247               GLB PUTOBJP                   ; PASCAL CALLS
                      248               GLB ACTIVATEP
                      249
                      250               GLB GAME_OPT                  ;GAME OPTIONS DISPLAY
                      251               GLB LOAD_ASCII                ;LOADS ASCII CHARACTER GENERATORS
                      252               GLB FILL_VRAM                 ;FILLS DESIGNATED AREA OF VRAM WITH VALUE
                      253               GLB MODE_1                    ;SETS UP A DEFAULT GRAPHICS MODE 1
                      254               GLB ASCII_TABLE               ;POINTER TO TABLE OF ASCII GENERATORS
                      255               GLB NUMBER_TABLE              ;POINTER TO TABLE OF 0-9 PATTERN GENERATORS
                      256
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     258 ********************** CARTRIDGE ROM DATA AREA ************************
                     259
                     260                 GLB             CARTRIDGE
<8000>               261 CARTRIDGE       EQU             8000H
                     262 * THIS IS THE MEMORY LOCATION TESTED TO SEE IF A CARTRIDGE IS PLUGGED
                     263 * IN. IF IT CONTAINS THE PATTERN AA55H THE OS ASSUMES THAT A GAME
                     264 * CARTRIDGE IS PRESENT. IF IT CONTAINS THE THE PATTERN 55AAH, THE OS
                     265 * ASSUMES THAT A TEST CARTRIDGE IS PRESENT.
                     266
                     267                 GLB             LOCAL_SPR_TBL
<8002>               268 LOCAL_SPR_TBL   EQU             8002H
                     269 * THIS IS A POINTER TO THE CPU RAM COPY OF THE SPRITE NAME TABLE. THE
                     270 * TABLE COPY IS USED WHENEVER ONE LEVEL OF INDIRECTION IS DESIRED IN
                     271 * ADDRESSING THE VRAM TABLE. FOR EXAMPLE WHEN USING THE OS SPRITE
                     272 * MULTIPLEXING SOFTWARE.
                     273
                     274                 GLB             SPRITE_ORDER
<8004>               275 SPRITE_ORDER    EQU             8004H
                     276 * THIS IS A POINTER TO THE CPU RAM SPRITE ORDER TABLE. THIS TABLE IS
                     277 * USED TO ORDER THE LOCAL SPRITE NAME TABLE.
                     278
                     279                 GLB             WORK_BUFFER
<8006>               280 WORK_BUFFER     EQU             8006H
                     281 * THIS IS A POINTER TO A FREE BUFFER SPACE IN RAM. THE OBJECT ORIENTED
                     282 * GRAPHICS ROUTINES USED THIS BUFFER FOR TEMPORARY STORAGE.
                     283
                     284                 GLB             CONTROLLER_MAP
<8008>               285 CONTROLLER_MAP  EQU             8008H
                     286 * THIS IS A POINTER TO THE CONTROLLER MEMORY MAP THAT IS MAINTAINED BY
                     287 * THE HIGH-LEVEL CONTROLLER SCANNING AND DEBOUNCE SOFTWARE.
                     288
                     289                 GLB             START_GAME
<800A>               290 START_GAME      EQU             800AH
                     291 * THIS IS A POINTER TO THE START OF THE GAME.
                     292
                     293 ******************** RESTART AND INTERRUPT VECTORS ********************
                     294 * THESE ARE ADDRESSES IN CARTRIDGE ROM OF VECTORS WHICH MUST BE PLACED
                     295 * THERE BY THE CARTRIDGE PROGRAMMER. WHEN AN INTERRUPT OR RESTART
                     296 * OCCURS, THE OS VECTORS IT THROUGH THIS AREA. THE CARTRIDGE PROGRAMMER
                     297 * SHOULD PLACE A JUMP TO HIS OWN INTERRUPT HANDLER IN THE APPROPRIATE
                     298 * LOCATION.
                     299
                     300                 GLB             RST_8H_RAM
<800C>               301 RST_8H_RAM      EQU             800CH
                     302 * THIS IS THE RESTART 8 SOFT VECTOR.
                     303
                     304                 GLB             RST_10H_RAM
<800F>               305 RST_10H_RAM     EQU             800FH
                     306 * THIS IS THE RESTART 10 SOFT VECTOR.
                     307
                     308                 GLB             RST_18H_RAM
<8012>               309 RST_18H_RAM     EQU             8012H
                     310 * THIS IS THE RESTART 18 SOFT VECTOR.
                     311
                     312                 GLB             RST_20H_RAM
<8015>               313 RST_20H_RAM     EQU             8015H
                     314 * THIS IS THE RESTART 20 SOFT VECTOR.
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      315
                      316                 GLB             RST_28H_RAM
<8018>                317 RST_28H_RAM     EQU             8018H
                      318 * THIS IS THE RESTART 28 SOFT VECTOR.
                      319
                      320                 GLB             RST_30H_RAM
<801B>                321 RST_30H_RAM     EQU             801BH
                      322 * THIS IS THE RESTART 30 SOFT VECTOR.
                      323
                      324                 GLB             IRQ_INT_VECT
<801E>                325 IRQ_INT_VECT    EQU             801EH
                      326 * THIS IS THE MASKABLE INTERRUPT SOFT VECTOR
                      327
                      328                 GLB             NMI_INT_VECT
<8021>                329 NMI_INT_VECT    EQU             8021H
                      330 * THIS IS THE NMI SOFT VECTOR.
                      331
                      332                 GLB             GAME_NAME
<8024>                333 GAME_NAME       EQU             8024H
                      334 * FROM HERE TO START_GAME THERE SHOULD BE A STRING OF ASCII CHARACTERS
                      335 * NAMES THAT HAS THE FOLLOWING FORM:
                      336 *
                      337 *    NAME_OF_THIS_GAME/MAKER_OF_THIS_GAME/COPYWRITE_YEAR.
                      338 *
                      339 *   FOR EXAMPLE:
                      340 *
                      341 *      "DONKEY KONG/NINTENDO/1982"
                      342
                      343 * IMPORTANT NOTE *****
                      344
                      345 *              **** IT IS THE RESPONSIBILITY OF THE ****
                      346 *              **** CARTRIDGE PROGRAMMER TO PLACE   ****
                      347 *              **** THESE CODES IN CARTRIDGE ROM    ****
                      348
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     350
                     351 ********************************************************************************
                     352 *                                                                              *
                     353 *                        OPERATING SYSTEM ROM CODE                             *
                     354 *                                                                              *
                     355 ********************************************************************************
                     356
                     357 ****************************** PAGE ZERO ********************************
                     358 * PAGE ZERO CONTAINS THE RESTART VECTORS, INTERRUPT VECTORS, AND
                     359 * THE INTERRUPT VECTORING SOFTWARE, AS WELL AS THE DEFAULT HANDLERS
                     360 * FOR INTERRUPTS AND RESTARTS.
                     361
                     362 * BOOT-UP ROUTINE
                     363
                     364 * THE BOOT-UP ROUTINE HANDLES POWER ON RESETS AND RESTARTS TO 0. IT
                     365 * INITIALIZES THE STACK AND JUMPS TO  THE POWER_UP ROUTINE.
                     366
                     367 * BEGIN BOOT-UP
                     368 BOOT_UP          PROG
                     369
                     370 *   KICK STACK
0000 317389          371                 LD              SP,STACK
                     372
                     373 *   JUMP TO POWER_UP
                     374
0003 C3006E          375                 JP POWER_UP
0006                 376 END_BOOTUP
                     377 * END BOOT-UP
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     379
                     380 * RESTART VECTORS
                     381
                     382 * THE FOLLOWING ARE THE 8 PROGRAMMABLE RESTARTS. FOR EACH OF THE
                     383 * RESTART LOCATIONS BELOW THERE IS A VECTOR IN CARTRIDGE ROM.
                     384 * TO USE A RESTART, THE PROGRAMMER MUST PLACE THE ADDRESS OF THE
                     385 * ROUTINE WHICH HE/SHE WISHES TO ACCESS THROUGH THE RESTART AT THE
                     386 * CORRSEPONDING VECTOR. THEREAFTER EVERY TIME THAT RESTART IS
                     387 * EXECUTED, THE CARTRIDGE PROGRAMMER'S ROUTINE WILL BE CALLED.
                     388
  0006 FFFF          389                HEX             FF,FF               ;Filler
  0008 C3800C        390 RST_8H         JP RST_8H_RAM
                     391
  0008 FFFFFFFFFF    392                HEX             FF,FF,FF,FF,FF      ;Filler
  0010 C3800F        393 RST_10H        JP RST_10H_RAM
                     394
  0013 FFFFFFFFFF    395                HEX             FF,FF,FF,FF,FF      ;Filler
  0018 C38012        396 RST_18H        JP RST_18H_RAM
                     397
  001B FFFFFFFFFF    398                HEX             FF,FF,FF,FF,FF      ;Filler
  0020 C38015        399 RST_20H        JP RST_20H_RAM
                     400
  0023 FFFFFFFFFF    401                HEX             FF,FF,FF,FF,FF      ;Filler
  0028 C38018        402 RST_28H        JP RST_28H_RAM
                     403
  002B FFFFFFFFFF    404                HEX             FF,FF,FF,FF,FF      ;Filler
  0030 C3801B        405 RST_30H        JP RST_30H_RAM
                     406
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     408
                     409 * MASKABLE INTERRUPT VECTORING SOFTWARE
                     410
                     411 * A MASKABLE INTERRUPT OCCURRING IN THE SYSTEM IS EQUIVALENT TO A
                     412 * RESTART TO 38H. THUS, THE MASKABLE INTERRUPT IS VECTORED IN EXACTLY
                     413 * THE SAME WAY AS THE VARIOUS RESTARTS GIVEN ABOVE. IN ORDER TO USE
                     414 * THE INTERRUPT, THE CARTRIDGE MUST PLACE THE ADDRESS OF HIS/HER
                     415 * INTERRUPT HANDLER IN THE IRQ_INT_VECT LOCATION IN CARTRIDGE ROM.
                     416
                     417 * THE CARTRIDGE PROGRAMMER IS RESPONSIBLE FOR SAVING ANY REGISTERS
                     418 * HIS/HER OWN INTERRUPT HANDLERS MAY USE, AND FOR RE-ENABLING
                     419 * INTERRUPTS IF HE/SHE NEEDS THEM TO BE RE-ENABLED.
                     420
                     421 * MASKABLE INTERRUPT
0033 FFFFFFFFFF      422                 HEX             FF,FF,FF,FF,FF      ;Filler
0038                 423 IRQ_INTERRUPT                   ;38H
0038 C3801E          424                 JP              (IRQ_INT_VECT)
                     425
                     426 ********************** RANDOM NUMBER GENERATOR **********************
                     427
                     428 * (PLACED HERE FOR PURPOSES OF CODE COMPACTION)
                     429
                     430 * Random number generator (psuedo) for a 16 bit value
                     431 * This routine 'exclusive or's the 15th and 8th bit
                     432 * together.  It then rotates the entire quantity to the
                     433 * left and inserts the 'exclusive or'ed bit into the rightmost
                     434 * bit.  Upon leaving it stores the random # in a specified
                     435 * memory location.
                     436
                     437 * The random number can be accessed from the global location
                     438 * RAND_NUM or the HL pair or the Accumulator.
                     439
0038                 440 RAND_GEN_:
0038 2A73C8          441           LD    HL,[RAND_NUM]
003C CB7C            442           BIT   7,H
0040 2806            443           JR    Z,NOT_ON
                     444                                 ;15th bit is on
0042 CB44            445           BIT   0,H
0044 2806            446           JR    Z,SET           ;For 10 quantity then set
0046 1807            447           JR    RESET           ;For 11 reset
0048                 448 NOT_ON:
0048 CB44            449           BIT   0,H
004A 2803            450           JR    Z,RESET         ;For 00 reset
                     451                                 ;For 01 fall through to set
004C                 452 SET:
004C 37              453           SCF
004D 1801            454           JR    CARRY_READY
004F                 455 RESET:
004F B7              456           OR    A
0050                 457 CARRY_READY:
0050 CB15            458           RL    L
0052 CB14            459           RL    H
0054 2273C8          460           LD    [RAND_NUM],HL
0057 7D              461           LD    A,L
0058 C9              462           RET
0059                 463 AFTER_RANDOM
                     464
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      465 ************************************************************************
                      466
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     468
                     469 * THE NMI VECTORING SOFTWARE AND DEFAULT HANDLER
                     470
                     471 * WHEN AN NMI IS RAISED BY THE VDP IN THE COLECOVISION SYSTEM, IT
                     472 * CAUSES THE CPU TO RESTART TO 66H. THE VECTORING SOFTWARE FOR THE
                     473 * NMI IS IDENTICAL TO THAT FOR THE MASKABLE INTERRUPT EXCEPT THAT
                     474 * IT GETS ITS VECTOR FROM NMI_INT_VECT INSTEAD OF IRQ_INT_VECT.
                     475
                     476 * AGAIN THE CARTRIDGE PROGRAMMER IS RESPONSIBLE, IN HIS/HER OWN
                     477 * INTERRUPT HANDLERS FOR SAVING AND RESTORING THE PROCESSOR STATE
                     478 * WHEN NECESSARY, AND FOR CLEARING THE VDP CONDITION BY READING THE
                     479 * VDP STATUS REGISTER.
                     480
0059 FFFFFFFFFF      481                  HEX          FF,FF,FF,FF,FF      ;Filler
005E FFFFFFFFFF      482                  HEX          FF,FF,FF,FF,FF      ;Filler
0063 FFFFFF          483                  HEX          FF,FF,FF            ;Filler
                     484 * NON-MASKABLE INTERRUPT
0066                 485 NMI_INTERRUPT
0066 C38021          486                  JP           (NMI_INT_VECT)
                     487
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     489
                     490 ************************** OS ROM DATA AREA **************************
                     491
    0069 3C          492 AMERICA         DEFB          60
                     493 * THIS BYTE SHOULD BE USED WHENEVER THE CARTRIDGE PROGRAMMER WANTS TO
                     494 * SET UP REAL-TIME COUNTERS. IT HAS A VALUE OF 60 FOR COLECOVISIONS
                     495 * MARKETED IN THE USA AND 50 FOR EUROPEAN UNITS. USE OF THIS BYTE
                     496 * ENSURES CARTRIDGE COMPATIBILITY AT LEAST WHERE REAL-TIME COUNTING
                     497 * IS CONCERNED.
                     498
    006A 16AB        499 ASCII_TABLE     DEFW ASCII_TBL
                     500 * THIS IS THE ADDRESS OF THE ROM PATTERN GENERATORS FOR UPPERCASE
                     501 * ASCII WHICH ARE CONTAINED WITHIN THE OPERATING SYSTEM.
                     502
    006C 1623        503 NUMBER_TABLE    DEFW NUMBER_TBL
                     504 * THIS IS THE ADDRESS OF THE ROM PATTERN GENERATORS FOR THE NUMBERS
                     505 * 0_9 WHICH ARE CONTAINED WITHIN THE OPERATING SYSTEM.
                     506
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      508
                      509 ************************* POWER ON BOOT SOFTWARE ********************
                      510 ;
                      511 ;BOOT_UP         SINCE THE VIDEO GAME SYSTEM MAY BE STARTED UP WITH A
                      512 ;                GAME CARTRIDGE, KEYBOARD MODULE, OR BOTH (OR NOTHING)
                      513 ;                INSTALLED AT BOOT UP, THE SOFTWARE MUST PERFORM THE
                      514 ;                FOLLOWING:
                      515 ;
                      516 ;                     A.  INITIALIZE THE INTERRUPT VECTORS.
                      517 ;
                      518 ;                     B.  INITIALIZE RESTART VECTORS
                      519 ;
                      520 ;                     C.  TURN OFF THE SOUND CHIP.
                      521 ;
                      522 ;                     D.  DETERMINE IF A CARTRIDGE IS PLUGGED IN.
                      523 ;                         IF SO, BRANCH TO THE CARTRIDGE PROGRAM
                      524 ;                         ELSE, WAIT FOR CARTRIDGE.
                      525
          <0000>      526 FALSE           EQU             0
          <0001>      527 TRUE            EQU             1
                      528 * VALUES FOR BOOLEAN  FLAGS
                      529
                      530 * BEGIN POWER_UP
          <006E>      531 POWER_UP        EQU             $
                      532
                      533 *   IF CARTRIDGE = 55AAH THEN  EXIT TO START_GAME (TEST)
 006E 2A8000          534                 LD              HL,[CARTRIDGE]
 0071 7D              535                 LD              A,L
 0072 FE55            536                 CP              55H
 0074 C20081          537                 JP              NZ,NO_TEST_
 0077 7C              538                 LD              A,H
 0078 FEAA            539                 CP              0AAH
 007A C20081          540                 JP              NZ,NO_TEST_
 007D 2A800A          541                 LD              HL,[START_GAME]
 0080 E9              542                 JP              [HL]
                      543
                      544 *   ELSE
 0081                 545 NO_TEST_
                      546
                      547 *   TURN OFF SOUND CHIP
 0081 CD1FD6          548                 CALL TURN_OFF_SOUND
                      549
                      550 *   INITIALIZE RANDOM NUMBER GENERATOR
 0084 210033          551                 LD              HL,33H
 0087 2273C8          552                 LD              [RAND_NUM],HL
                      553
                      554 *   CLEAR CONTROLLER BUFFER AREAS
 008A CD1105          555                 CALL            CONTROLLER_INIT
                      556
                      557 *   DEFER_WRITES := FALSE
 008D 3E00            558                 LD              A,FALSE
 008F 3273C6          559                 LD              [DEFER_WRITES],A
                      560
                      561 *   MUX_SPRITES := FALSE
 0092 3273C7          562                 LD              [MUX_SPRITES],A
                      563
                      564
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     565 *   EXIT TO DISPLAY LOGO AND TEST FOR CARTRIDGE
    0095 C31319      566                 JP              DISPLAY_LOGO
                     567
                     568 * END BOOT-UP
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      570 ************************** SYSTEM RAM AREA ****************************
                      571                 DATA
0000                  572                 DEFS            73BAH ;Added to offset to first location
        <738A>        573 SYSTEM_RAM_AREA EQU             $
                      574 * THIS IS THE RAM AREA DEDICATED TO THE BASIC OS NEEDS. IT INCLUDES THE
                      575 * STACK, VARIOUS STATUS VARIABLES, AND ALL THE VARIABLES USED BY OS
                      576 * ROUTINES.
                      577
        <7389>        578 STACK           EQU             SYSTEM_RAM_AREA-1
                      579 * THIS IS THE TOP OF THE STACK
                      580
                      581 ;               COMN
                      582 ;               DEFS            9
                      583                 DATA
738A                  584 PARAM_AREA:
738A                  585 INIT_SOUND_DATA:
738A                  586 PRN_AREA:
738A                  587 INIT_TIME_DATA:
738A                  588 TEMP1:
738A                  589           DEFS 1
738B                  590           DEFS 1
738C                  591 TEMP2:
738C                  592           DEFS 1
738D                  593           DEFS 1
738E                  594 SIGNAL_NUM:
738E                  595           DEFS 1
738F                  596 REPEAT_SIG_CODE:
738F                  597           DEFS 1
73C0                  598 TIMER_LENGTH:
73C0                  599           DEFS 1
73C1                  600           DEFS 1
73C2                  601 TEST_SIG_NUM:
73C2                  602           DEFS 1
                      603
                      604
                      605 * THIS IS THE COMMON PARAMETER PASSING AREA AND THE HOLE IN THE DATA
                      606 * AREA THAT IS PROVIDED TO MAKE ROOM FOR IT.
                      607
73C3                  608 VDP_MODE_WORD   DEFS            2
                      609 * THE VDP MODE WORD CONTAINS A COPY OF THE DATA IN THE FIRST TWO VDP
                      610 * REGISTERS. BY EXAMINING THIS DATA, THE OS AND CARTRIDGE PROGRAMS
                      611 * CAN MAKE MODE-DEPENDENT DECISIONS ABOUT THE SPRITE SIZE OR VRAM
                      612 * TABLE ARRANGEMENT. THIS WORD IS MAINTAINED BY THE WRITE_REGISTER
                      613 * ROUTINE WHENEVER THE CONTENTS OF REGISTERS 0 OR 1 ARE CHANGED.
                      614
                      615 *   IMPORTANT NOTE *****
                      616
                      617 *               **** IT IS THE RESPONSIBILITY OF THE ****
                      618 *               **** CARTRIDGE PROGRAMMER TO MAKE    ****
                      619 *               **** SURE THAT NON-STANDARD USE OF   ****
                      620 *               **** THE VDP REGISTERS DOES NOT MAKE ****
                      621 *               **** THE DATA IN THIS WORD INVALID   ****
                      622
73C5                  623 VDP_STATUS_BYTE DEFS            1
                      624 * THE DEFAULT HANDLER FOR THE NMI, WHICH MUST READ THE VDP STATUS
                      625 * REGISTER TO CLEAR THE INTERRUPT CONDITION, PLACES ITS CONTENTS
                      626 * HERE. THIS BYTE IS THE MOST ACCURATE REPRESENTATION OF THE ACTUAL
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     627 * VDP STATUS THAT IS AVAILABLE TO THE CARTRIDGE PROGRAMMER PROVIDED
                     628 * THAT THE VDP INTERRUPT IS ENABLED ON-CHIP
                     629
    73C6             630 DEFER_WRITES    DEFS            1
                     631 * DEFER_WRITES IS A BOOLEAN FLAG WHICH IS SET TO FALSE AT POWER UP
                     632 * TIME, SHOULD BE SET TO TRUE ONLY IF THE CARTRIDGE PROGRAMMER WISHES
                     633 * TO DEFER WRITES TO VRAM. IF THIS FLAG  IS TRUE THEN THE WRITER
                     634 * ROUTINE MUST BE CALLED REGULARLY TO PERFORM DEFERRED WRITES.
                     635
    73C7             636 MUX_SPRITES     DEFS            1
                     637 * THIS BOOLEAN FLAG WITH DEFAULT FALSE VALUE SHOULD BE SET TO TRUE IF
                     638 * THE CARTRIDGE PROGRAMMER WISHES ONE LEVEL OF INDIRECTION TO BE
                     639 * INSERTED INTO SPRITE PROCESSING BY HAVING ALL SPRITES WRITTEN TO
                     640 * A LOCAL SPRITE_NAME_TABLE BEFORE BEING WRITTEN TO VRAM. THIS  AIDS
                     641 * SPRITE MULTIPLEXING SOLUTIONS TO THE FIFTH SPRITE PROBLEM.
                     642
                     643                 GLB             RAND_NUM
    73C8             644 RAND_NUM        DEFS  2
                     645 * THIS IS THE SHIFT REGISTER USED BY THE RANDOM NUMBER GENERATOR.
                     646 * IT IS INITIALIZED AT POWER-UP.
                     647         GLB PARAM_
                     648         PROG
    0098             649 PARAM_
    0098 E1E3E50A6F  650         HEX E1,E3,E5,0A,6F,03,0A,03,67,E3,D5,5E,23,56,23,E5
    009D 030A0367E3
    00A2 D55E235623
    00A7 E5
    00A8 7B82C2B700  651         HEX 7B,B2,C2,B7,00,E1,5E,23,56,23,E5,EB,5E,23,56,03
    00AD E15E235623
    00B2 E5EB5E2356
    00B7 03
    00B8 0A07D2DA00  652         HEX 0A,07,D2,DA,00,03,E1,E3,73,23,72,23,D1,E3,2B,AF
    00B0 03E1E37323
    00C2 7223D1E32B
    00C7 AF
    00C8 BCC2D000BD  653         HEX BC,C2,D0,00,BD,CA,D6,00,E3,E5,EB,C3,A3,00,E1,EB
    00CD CAD600E3E5
    00D2 EBC3A300E1
    00D7 EB
    00D8 E3E9E1E3E5  654         HEX E3,E9,E1,E3,E5,0F,67,0B,0A,6F,E3,03,03,1A,77,23
    00D0 0F67080A6F
    00E2 E303031A77
    00E7 23
    00E8 13E32BAFBD  655         HEX 13,E3,2B,AF,BD,C2,F4,00,BC,CA,F8,00,E3,C3,E5,00
    00ED C2F400BCCA
    00F2 F800E3C3E5
    00F7 00
    00F8 E1C3C400    656         HEX E1,C3,C4,00
                     657  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     659 ;        .IDENT FREQSWE          ;includes FREQ_SWEEP
                     660 ;****************************
                     661 ;*       FREQ_SWEEP         *
                     662 ;****************************
                     663 ;.COMMENT )
                     664 ;See Users' Manual for description
                     665 ;RETs Z SET: if note over
                     666 ;RETs Z RESET: if sweep in progress or note not over
                     667 ;)
                     668         GLB FREQ_SWEEP
                     669        ;EXT DECLSN,MSNTOLSN,DECMSN,ADD816
                     670        ;INCLUDE OSSR_EQU:OS:0  ;equates
  00FC               671 FREQ_SWEEP
                     672 ;       * if freq not swept, dec NLEN and RET [setting Z flag]
  00FC DD7E07        673         LD A,[IX+FSTEP]         ;check  for no sweep code
  00FF FE00          674         CP 0                    ;SET Z flag if FSTEP = 0
                     675 ;       IF [PSW,IS,ZERO]        ;note not to be swept
  0101 2009          676         JR NZ,L20
  0103 DD7E05        677           LD A,[IX+NLEN]          ;dec NLEN and
  0106 3D            678           DEC A                   ;SET Z flag if NLEN = 0
  0107 C8            679           RET Z                   ;leave if note over with Z flag SET
  0108 DD7705        680             LD [IX+NLEN],A          ;store decremented NLEN
  010B C9            681             RET                     ;RET with Z flag RESET [note not over]
                     682 ;       ENDIF
                     683 ;       * sweep going, so dec FPSV
  010C DDE5          684 L20     PUSH IX                 ;point HL to FPSV
  010E E1            685         POP HL
  010F 1E06          686         LD E,FPSV
  0111 1600          687         LD D,0
  0113 19            688         ADD HL,DE
  0114 CD0190        689         CALL DECLSN             ;dec FPSV
                     690 ;       IF [PSW,IS,ZERO]        ;FPSV has timed out
  0117 2015          691         JR NZ,L21
                     692 ;         * dec NLEN and leave if sweep is over
  0119 CD01A6        693           CALL MSNTOLSN           ;reload FPSV from FPS
  011C 2B            694           DEC HL                  ;point to NLEN [# steps in the sweep]
  011D 7E            695           LD A,[HL]               ;dec NLEN and
  011E 3D            696           DEC A                   ;SET Z flag if NLEN = 0
  011F C8            697           RET Z                   ;leave if sweep over with Z flag SET
                     698 ;           * sweep not over, so add FSTEP to FREQ
  0120 77            699             LD [HL],A               ;store decremented NLEN
  0121 2B            700             DEC HL                  ;point HL
  0122 2B            701             DEC HL                  ;to FREQ
  0123 DD7E07        702             LD A,[IX+FSTEP]         ;A = FSTEP [two's complement step size]
  0126 CD01B1        703             CALL ADD816             ;FREQ = FREQ + FSTEP
  0129 23            704             INC HL                  ;point HL to hi FREQ
  012A CB96          705             RES 2,[HL]              ;RESET B2 in hi FREQ in case add caused > 10 bit FREQ
  012C F6FF          706             OR 0FFH                 ;RESET Z flag, sweep not over yet
                     707 ;       ENDIF
  012E C9            708 L21     RET
                     709 ;       END ;FREQSWE
                     710  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     712 ;        .IDENT ATNSWEE          ;Includes ATN_SWEEP
                     713 ;*************************
                     714 ;*       ATN_SWEEP         *
                     715 ;*************************
                     716 ;.COMMENT )
                     717 ;See User's Manual for description
                     718 ;RETs Z SET: if byte 8 is 0 [means sweep is over, or note was never swept]
                     719 ;RETs Z RESET: if sweep in progress
                     720 ;)
                     721         GLB ATN_SWEEP
                     722        ;EXT DECLSN,DECMSN,MSNTOLSN
                     723        ;INCLUDE OSSR_EQU:OS:0  ;equates
  012F               724 ATN_SWEEP
                     725 ;       * RET with Z SET if byte 8 = 00
  012F DD7E08        726         LD A,[IX+8]             ;check byte 8 for no sweep code
  0132 FE00          727         CP 0                    ;Z is set if byte 8 = 0
  0134 C8            728         RET Z                   ;leave if Z set, sweep not going
                     729 ;         * sweep going, so dec APSV
  0135 DDE5          730           PUSH IX                ;point HL to APSV
  0137 E1            731           POP HL
  0138 1600          732           LD D,0
  013A 1E09          733           LD E,APSV
  013C 19            734           ADD HL,DE
  013D CD0190        735           CALL DECLSN            ;dec APSV [LSN of byte 9]
                     736 ;         IF [PSW,IS,ZERO]       ;APSV has timed out
  0140 2021          737           JR NZ,L22
                     738 ;           * dec ALEN to see if sweep over
  0142 CD01A6        739             CALL MSNTOLSN         ;reload APSV from APS
  0145 2B            740             DEC HL                ;point to ALEN [# of steps in the sweep]
  0146 CD0190        741             CALL DECLSN           ;dec ALEN [LSN byte 8]
                     742 ;           IF [PSW,IS,NZERO]     ;sweep not over yet
  0149 2816          743             JR Z,L23
                     744 ;             * add ASTEP to ATN
  014B 7E            745               LD A,[HL]           ;MSN A = ASTEP
  014C E6F0          746               AND 0F0H            ;mask LSN
  014E 5F            747               LD E,A              ;E = ASTEP | 0
  014F 2B            748               DEC HL              ;point HL to ATN
  0150 2B            749               DEC HL
  0151 2B            750               DEC HL
  0152 2B            751               DEC HL
  0153 7E            752               LD A,[HL]           ;MSN A = ATN
  0154 E6F0          753               AND 0F0H            ;A = ATN | 0
  0156 83            754               ADD A,E             ;MSN A = [ASTEP + ATN] | 0
  0157 5F            755               LD E,A              ;E = [ASTEP + ATN] | 0
  0158 7E            756               LD A,[HL]           ;A = ATN | freq or CTRL
  0159 E60F          757               AND 0FH             ;mask old ATN  A = 0 | freq or CTRL
  015B B3            758               OR E                ;OR in new ATN
  015C 77            759               LD [HL],A           ;store updated value back into song data area
  015D F6FF          760               OR 0FFH             ;RESET Z flag, sweep not over yet
  015F 1802          761               JR L22
                     762 ;           ELSE                 ;Z flag is SET: sweep over
  0161 3600          763 L23           LD [HL],0           ;set byte 8 to 0 to indicate end sweep
                     764 ;           ENDIF
                     765 ;         ENDIF
  0163 C9            766 L22     RET
                     767 ;       END ;ATNSWEE
                     768 PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     770 ;        .IDENT UTIL              ;Includes UPATNCTRL,UPFREQ,
                     771                                   ;DECLSN,DECMSN,MSNTOLSN,ADD816,PT_IX_TO_SxDATA,
                     772                                   ;LEAVE_EFFECT,AREA_SONG_IS
                     773 ;*****************************
                     774 ;*       UPATNCTRL           *
                     775 ;*****************************
                     776 ;.COMMENT )
                     777 ;Perform single byte update of the snd chip noise control register or any
                     778 ;attenuation register. IX is passed pointing to byte 0 of a song data area, MSN
                     779 ;register C = formatted channel attenuation code.
                     780 ;)
                     781         GLB UPATNCTRL
                     782        ;INCLUDE OSSR_EQU:OS:0  ;equates
   0164              783 UPATNCTRL
   0164 DD7E04       784         LD A,[IX+4]             ;MSN A = ATN, LSN may be CTRL data
   0167 CB61         785         BIT 4,C                 ;test for ATN
                     786 ;       IF [PSW,IS,NZERO]       ;ATN is to be sent, move it to the LSN
   0169 2804         787         JR Z,L24
   0168 0F           788           RRCA                    ;swap nibbles
   016C 0F           789           RRCA
   016D 0F           790           RRCA
   016E 0F           791           RRCA                    ;LSN A = ATN
                     792 ;       ENDIF
   016F E60F         793 L24     AND 0FH                 ;mask MSN
   0171 B1           794         OR C                    ;A = formatted register# | ATN or CTRL
   0172 D3FF         795         OUT [SOUND_PORT],A      ;output ATN or CTRL data
   0174 C9           796         RET
                     797 ;*****************************
                     798 ;*       UPFREQ              *
                     799 ;*****************************
                     800 ;.COMMENT )
                     801 ;Perform double byte update of a sound chip frequency register.  IX is passed
                     802 ;pointing to byte0 of a song data area, MSN register D = formatted channel
                     803 ;frequency code.
                     804 ;)
                     805         GLB UPFREQ
   0175              806 UPFREQ
   0175 DD7E03       807         LD A,[IX+FREQ]          ;A = F2 F3 F4 F5 F6 F7 F8 F9
   0178 E60F         808         AND 0FH                 ;A =  0  0  0  0 F6 F7 F8 F9
   017A B2           809         OR D                    ;A =  formatted reg# | F6 F7 F8 F9
   017B D3FF         810         OUT [SOUND_PORT],A      ;output first freq byte
   017D DD7E03       811         LD A,[IX+FREQ]          ;A = F2 F3 F4 F5 F6 F7 F8 F9 again
   0180 E6F0         812         AND 0F0H                ;A = F2 F3 F4 F5  0  0  0  0
   0182 57           813         LD D,A                  ;save in D
   0183 DD7E04       814         LD A,[IX+FREQ+1]        ;LSN A = 0  0 F0 F1
   0186 E60F         815         AND 0FH                 ;A =  0  0  0  0  0  0 F0 F1
   0188 B2           816         OR D                    ;A = F2 F3 F4 F5  0  0 F0 F1
   0189 0F           817         RRCA                    ;swap nibbles
   018A 0F           818         RRCA
   018B 0F           819         RRCA
   018C 0F           820         RRCA                    ;A =  0  0 F0 F1 F2 F3 F4 F5
   018D D3FF         821         OUT [SOUND_PORT],A      ;output 2nd [most significant] freq byte
   018F C9           822         RET
                     823 ;*****************************
                     824 ;*       DECLSN              *
                     825 ;*****************************
                     826 ;.COMMENT )
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     827 ;Without affecting the MSN, decrement the LSN of the byte pointed to by HL.  HL
                     828 ;remains the same.
                     829 ;RET with Z flag set if dec LSN results in 0, reset otherwise.
                     830 ;RET with C flag set if dec LSN results in -1, reset otherwise.
                     831 ;)
                     832         GLB DECLSN
  0190 3E00          833 DECLSN  LD A,0
  0192 ED67          834         RRD                     ;A = 0 | LSN [HL]
  0194 D601          835         SUB 1                   ;Z flag set if dec to 0, C flag if dec to -1
  0196 F5            836         PUSH AF                 ;save Z and C flags
  0197 ED6F          837         RLD                     ;[HL] = old MSN | new LSN
  0199 F1            838         POP AF                  ;restore Z and C flags, A = 0 | new LSN
  019A C9            839         RET
                     840 ;*************************
                     841 ;*       DECMSN          *
                     842 ;*************************
                     843 ;.COMMENT )
                     844 ;Without affecting the LSN, decrement the MSN of the byte pointed to by HL.  HL
                     845 ;remains the same.
                     846 ;RET with Z flag set if dec MSN results in 0, reset otherwise.
                     847 ;RET with C flag set if dec MSN results in -1, reset otherwise.
                     848 ;)
                     849         GLB DECMSN
  019B 3E00          850 DECMSN  LD A,0
  019D ED6F          851         RLD                     ;A = 0 | MSN [HL]
  019F D601          852         SUB 1                   ;Z flag set if dec to 0, C flag set if dec -1
  01A1 F5            853         PUSH AF                 ;save Z and C flags
  01A2 ED67          854         RRD                     ;[HL] = new MSN | old LSN
  01A4 F1            855         POP AF                  ;restore Z and C flags, A = 0 | new MSN
  01A5 C9            856         RET
                     857 ;*************************
                     858 ;*       MSNTOLSN        *
                     859 ;*************************
                     860 ;.COMMENT )
                     861 ;Copy MSN of the byte pointed to by HL to the LSN of that byte.  HL remains
                     862 ;the same.
                     863 ;)
                     864         GLB MSNTOLSN
  01A6               865 MSNTOLSN
  01A6 7E            866         LD A,[HL]             ;A = MSN | LSN to be changed
  01A7 E6F0          867         AND 0F0H                ;A = MSN | 0
  01A9 47            868         LD B,A                  ;save in B
  01AA 0F            869         RRCA                    ;swap nibbles
  01AB 0F            870         RRCA
  01AC 0F            871         RRCA
  01AD 0F            872         RRCA                    ;A = 0 | MSN
  01AE B0            873         OR B                    ;A = MSN | MSN
  01AF 77            874         LD [HL],A               ;[HL] = MSN | MSN
  01B0 C9            875         RET
                     876 ;*************************
                     877 ;*       ADD816          *
                     878 ;*************************
                     879 ;.COMMENT )
                     880 ;Adds 8 bit two's complement signed value passed in A to the 16 bit location
                     881 ;pointed to by HL.
                     882 ;)
                     883         GLB ADD816
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   01B1 0600          884 ADD816  LD B,0                ;set B for positive value in A
   01B3 CB7F          885         BIT 7,A               ;if A is positive
   01B5 2802          886         JR Z,POS              ;skip
   01B7 06FF          887           LD B,0FFH             ;A is neg: extend sign bit thru B
   01B9 86            888 POS     ADD A,[HL]            ;do 8 bit add [and set Carry]
   01BA 77            889         LD [HL],A             ;store result into LSB 16 bit number
   01BB 23            890         INC HL                ;put MSB
   01BC 7E            891         LD A,[HL]             ;into A
   01BD 88            892         ADC A,B               ;A = MSB + Carry + B [B is 0 or FF]
   01BE 77            893         LD [HL],A             ;store result into MSB
   01BF 2B            894         DEC HL                ;re-point HL to LSB 16 bit number
   01C0 C9            895         RET
                      896 ;*****************************
                      897 ;*     PT_IX_TO_SxDATA       *
                      898 ;*****************************
                      899 ;.COMMENT )
                      900 ;SONGNO passed in B.
                      901 ;Point IX to byte 0 in SONGNO's song data area.
                      902 ;RET with both DE and IX pointing to SxDATA,
                      903 ;HL pointing to MSB SxDATA entry in LST_OF_SND_ADDRS.
                      904 ;)
                      905         GLB PT_IX_TO_SxDATA
   01C1               906 PT_IX_TO_SxDATA
                      907 ;       * IX & DE := addr of byte 0 in SONGNO's song  data area,
                      908 ;         HL pointing to MSB SxDATA entry in LST_OF_SND_ADDRS.
   01C1 2A7020        909         LD HL,(PTR_TO_LST_OF_SND_ADDRS) ;point HL to start LST_OF_SND_ADDRS
   01C4 2B            910         DEC HL                ;init HL for addition
   01C5 2B            911         DEC HL
   01C6 48            912         LD C,B                ;form 4 * SONGNO in C
   01C7 0600          913         LD B,0
   01C9 CB01          914         RLC C
   01CB CB01          915         RLC C
   01CD 09            916         ADD HL,BC             ;HL pts to SxDATA's entry in LST_OF_SND_ADDRS
   01CE 5E            917         LD E,[HL]             ;move addr SxDATA to IX thru DE
   01CF 23            918         INC HL
   01D0 56            919         LD D,[HL]
   01D1 D5            920         PUSH DE
   01D2 DDE1          921         POP IX
   01D4 C9            922         RET
                      923 ;
                      924 ;                 LEAVE EFFECT
                      925 ;4/19/82
                      926 ;
                      927 ;*****************************
                      928 ;*     LEAVE_EFFECT          *
                      929 ;*****************************
                      930 ;.COMMENT )
                      931 ;LEAVE_EFFECT, called by a special sound effect routine when it's finished,
                      932 ;restores the SONGNO of the song to which the effect note belongs to B5 - B0 of
                      933 ;byte 0 in the effect's data area, and loads bytes 1 and 2 with the address of
                      934 ;the next note in the song.  The address of the 1 byte SONGNO (saved by the
                      935 ;effect when first called) is passed in DE.  The 2 byte address of the next note
                      936 ;in the song, also saved by the effect, is passed in HL.  IX is assumed to be
                      937 ;pointing to byte 0 of the data area to which the song number is to be restored.
                      938 ;Bits 7 and 6 of the saved SONGNO are ignored, and therefore may be used by the
                      939 ;effect to store flag information during the course of the note.
                      940 ;)
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     941        GLB LEAVE_EFFECT
    01D5             942 LEAVE_EFFECT
    01D5 DD7501      943        LD [IX+1],L          ;LSB NEXT_NOTE_PTR := LSB addr next note in song
    01D8 DD7402      944        LD [IX+2],H          ;MSB NEXT_NOTE_PTR := MSB addr next note in song
    01DB 1A          945        LD A,[DE]            ;A := x x SONGNO (i.e., the saved, original SONGNO)
    01DC E63F        946        AND 03FH             ;A := 0 0 SONGNO
    01DE 47          947        LD B,A               ;B := 0 0 SONGNO (B now = original SONGNO)
    01DF DD7E00      948        LD A,[IX+0]          ;A := CH# | 62 (all effect notes have SONGNO = 62)
    01E2 E6C0        949        AND 0C0H             ;A := CH# 0 0 0 0 0 0
    01E4 B0          950        OR B                 ;A := CH# | SONGNO
    01E5 DD7700      951        LD [IX+0],A          ;restore the song number
    01E8 C9          952        RET
                     953
                     954 ;*****************************
                     955 ;*      AREA_SONG_IS         *
                     956 ;*****************************
                     957 ;.COMMENT )
                     958 ;The address of byte 0 of a song data area is passed in IX.  The song number of
                     959 ;the song using that area is returned in A [0FFH if inactive].  If a special
                     960 ;effect was using that area, 62 is returned in A and HL is returned with the
                     961 ;address of the special sound effect routine.
                     962 ;)
                     963        GLB AREA_SONG_IS
    01E9             964 AREA_SONG_IS
    01E9 DD7E00      965        LD A,[IX+0]          ;A := CH# | SONGNO or 62, or A := FF
    01EC FEFF        966        CP 0FFH
    01EE C8          967        RET Z                ;leave with A = FF if area inactive
    01EF E63F        968          AND 00111111B        ;mask CH#
    01F1 FE3E        969          CP 62
    01F3 C0          970          RET NZ               ;leave with A = SONGNO if not a special effect [62]
                     971 ;          special effect, so set HL to addr effect, stored in bytes 1&2
    01F4 DDE5        972          PUSH IX              ;point HL to byte 1
    01F6 E1          973          POP HL
    01F7 23          974          INC HL
    01F8 5E          975          LD E,[HL]            ;save LSB effect addr in E
    01F9 23          976          INC HL               ;HL to byte 2
    01FA 56          977          LD D,[HL]            ;save MSB effect addr in D
    01FB EB          978          EX DE,HL             ;HL := addr special effect
    01FC C9          979          RET
                     980 ;       END ;UTIL
                     981  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      983 ;         .IDENT INITSOU          ;includes INIT_SOUND,ALL_OFF
                      984 ;*****************************
                      985 ;*       INIT_SOUND          *
                      986 ;*****************************
                      987 ;.COMMENT )
                      988 ;See Users' Manual for description; includes ENTRY POINT ALL_OFF
                      989 ;addr LST_OF_SND_ADDRS passed in HL
                      990 ;n = # of song data areas to init, passed in B
                      991 ;)
                      992          GLB INIT_SOUND,ALL_OFF,DUMAREA
                      993         ;INCLUDE OSSR_EQU:OS  ;equates
                      994 ;         *** Sound chip register code EQUATES
                      995 ;         Tone generator frequency and attenuation formatted register codes
          <0080>      996 SR1FRQ   EQU 10000000B          ;BIT7 = 1, BIT6-4 = TONE GEN 1 FREQ CODE
          <0090>      997 SR1ATN   EQU 10010000B          ;BIT7 = 1, BIT6-4 = TONE GEN 1 ATTN CODE
          <00A0>      998 SR2FRQ   EQU 10100000B          ;BIT7 = 1, BIT6-4 = TONE GEN 2 FREQ CODE
          <00B0>      999 SR2ATN   EQU 10110000B          ;BIT7 = 1, BIT6-4 = TONE GEN 2 ATTN CODE
          <00C0>     1000 SR3FRQ   EQU 11000000B          ;BIT7 = 1, BIT6-4 = TONE GEN 3 FREQ CODE
          <00D0>     1001 SR3ATN   EQU 11010000B          ;BIT7 = 1, BIT6-4 = TONE GEN 1 ATTN CODE
                     1002 ;         Noise generator control and attenuation formatted register codes
          <00E0>     1003 SRNCTL   EQU 11100000B          ;BIT7 = 1, BIT6-4 = NOISE GEN CONTROL CODE
          <00F0>     1004 SRNATN   EQU 11110000B          ;BIT7 = 1, BIT6-4 = NOISE GEN ATTN CODE
                     1005
                     1006 * PROCEDURE INIT_SOUNDQ (AREA_COUNT:BYTE;LIST_OF_ADDR:INTEGER)
                     1007
                     1008 * THIS IS THE PASCAL ENTRY POINT TO INIT_SOUND
                     1009
                     1010           ;EXT  PARAM
                     1011           GLB  INIT_SOUNDQ
  01FD 00020001      1012 INIT_SOUND_PAR:     DEFW  2,1,2
  0201 0002
                     1013
                     1014 ;          COMN
                     1015 ;INIT_SOUND_DATA:    DEFS  3    ;Moved to OS
                     1016
                     1017           PROG
  0203               1018 INIT_SOUNDQ:
  0203 0101FD        1019           LD   BC,INIT_SOUND_PAR
  0206 11738A        1020           LD   DE,INIT_SOUND_DATA
  0209 CD0098        1021           CALL PARAM
  020C 3A738A        1022           LD   A,[INIT_SOUND_DATA]
  020F 47            1023           LD   B,A
  0210 2A738B        1024           LD   HL,[INIT_SOUND_DATA+1]
                     1025
  0213               1026 INIT_SOUND
                     1027 ;       * initialize PTR_TO_LST_OF_SND_ADDRS with value passed in HL
  0213 227020        1028         LD [PTR_TO_LST_OF_SND_ADDRS],HL
                     1029 ;       * store inactive code at byte 0 of each of the n data areas [B = n]
  0216 23            1030         INC HL                  ;pt HL to song 1 data area entry in LST_OF_SND_ADDRS
  0217 23            1031         INC HL
  0218 5E            1032         LD E,[HL]               ;pt DE to byte 0 in first song data area
  0219 23            1033         INC HL
  021A 56            1034         LD D,[HL]
  021B E8            1035         EX DE,HL                ;pt HL to byte 0 in first song data area
  021C 1E0A          1036         LD E,10                 ;set DE for 10 byte increment
  021E 1600          1037         LD D,0
  0220 36FF          1038 B1:     LD [HL],0FFH            ;deactivate area
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0222 19          1039         ADD HL,DE               ;pt HL to byte 0 next area (10 bytes away)
    0223 10FB        1040         DJNZ B1                 ;do this for the n (passed in B) data areas
                     1041 ;       * store end of data area code (0) at first byte after last song data area
    0225 3600        1042         LD (HL),0               ;store end of data area code in byte 0 data area n + 1
                     1043 ;       * set the 4 channel data area pointers to a dummy, inactive data area
    0227 21024C      1044         LD HL,DUMAREA           ;point HL to inactive byte below (after the RET)
    022A 227022      1045         LD (PTR_TO_S_ON_0),HL   ;store addr DUMAREA at PTR_TO_S_ON_0
    022D 227024      1046         LD (PTR_TO_S_ON_1),HL   ;store addr DUMAREA at PTR_TO_S_ON_1
    0230 227026      1047         LD (PTR_TO_S_ON_2),HL   ;store addr DUMAREA at PTR_TO_S_ON_2
    0233 227028      1048         LD (PTR_TO_S_ON_3),HL   ;store addr DUMAREA at PTR_TO_S_ON_3
                     1049 ;       * initialize SAVE_CTRL
    0236 3EFF        1050         LD A,0FFH               ;note: this is only time MSN SAVE_CTRL will be non zero,
    0238 32702A      1051         LD (SAVE_CTRL),A        ;thus ensuring PLAY_SONGS will output 1st real CTRL data
    0238             1052 ALL_OFF
                     1053 ;       * turn off all 4 sound generators
    0238 3E9F        1054         LD A,OFF+SR1ATN         ;form off code for tone generator 1
    023D D3FF        1055         OUT (SOUND_PORT),A      ;send it out
    023F 3EBF        1056         LD A,OFF+SR2ATN         ;form off code for tone generator 2
    0241 D3FF        1057         OUT (SOUND_PORT),A      ;send it out
    0243 3EDF        1058         LD A,OFF+SR3ATN         ;form off code for tone generator 3
    0245 D3FF        1059         OUT (SOUND_PORT),A      ;send it out
    0247 3EFF        1060         LD A,OFF+SRNATN         ;form off code for noise generator, N
    0249 D3FF        1061         OUT (SOUND_PORT),A      ;send it out
    024B C9          1062         RET
    024C FF          1063 DUMAREA DEFB INACTIVE
                     1064 ;       END ;INITSOU
                     1065 PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1067 ;        .IDENT JUKEBOX          ;includes JUKE_BOX
                     1068 ;*****************************
                     1069 ;*       JUKE_BOX            *
                     1070 ;*****************************
                     1071 ;.COMMENT )
                     1072 ;see Users' Manual for description
                     1073 ;SONGNO passed in B
                     1074 ;)
                     1075         GLB JUKE_BOX
                     1076        ;EXT PT_IX_TO_SxDATA,LOAD_NEXT_NOTE,UP_CH_DATA_PTRS
                     1077        ;INCLUDE OSSR_EQU:OS  ;equates
                     1078
                     1079 * PROCEDURE JUKE_BOXQ (SONG_NUM:BYTE)
                     1080
                     1081 * THIS IS THE PASCAL ENTRY POINT FOR JUKE_BOX
                     1082
  024D 00010001      1083 JUKE_BOX_PAR    DEFW            1,1
                     1084 * THE PARAMETER DESCRIPTOR FOR JUKE_BOXQ
                     1085
                     1086                 ;EXT            PARAM_
                     1087 * THE PARAMETER PASSING ROUTINE
                     1088
                     1089 ;                COMN
                     1090 ;PARAM_AREA      DEFS            1
                     1091 * THIS IS WHERE THE SONG NUMBER IS ACTUALLY PASSED
                     1092
                     1093                 PROG
                     1094                 GLB             JUKE_BOXQ
  0251               1095 JUKE_BOXQ
  0251 01024D        1096           LD    BC,JUKE_BOX_PAR
  0254 11738A        1097           LD    DE,PARAM_AREA
  0257 CD0098        1098           CALL  PARAM_
  025A 3A738A        1099           LD    A,[PARAM_AREA]
  025D 47            1100           LD    B,A
                     1101
  025E               1102 JUKE_BOX
                     1103 ;       * RET if song already in progress
  025E C5            1104         PUSH BC                 ;save SONGNO on stack
  025F CD01C1        1105         CALL PT_IX_TO_SxDATA    ;point IX to SONGNO's song data area
  0262 DD7E00        1106         LD A,[IX+0]             ;A := CH# [if any] | SONGNO [if any]
  0265 E63F          1107         AND 3FH                 ;A := 0 0 SONGNO
  0267 C1            1108         POP BC                  ;B := SONGNO
  0268 B8            1109         CP B                    ;test if already in progress
  0269 C8            1110         RET Z                   ;if so, leave
                     1111 ;         * load first note and set NEXT_NOTE_PTR [thru LOAD_NEXT_NOTE]
  026A DD7000        1112           LD [IX+0],B             ;store SONGNO in byte 0
  026D 2B            1113           DEC HL                  ;-HL left by PT_IX_TO_SxDATA pointing to MSB SxDATA
  026E 2B            1114           DEC HL                  ;-entry in LST_OF_SND_ADDRS; point HL to note list
  026F 56            1115           LD D,[HL]               ;-starting addr entry in LST_OF_SND_ADDRS and save this
  0270 2B            1116           DEC HL                  ;-addr in DE
  0271 5E            1117           LD E,[HL]               ;DE now has the initial value for NEXT_NOTE_PTR
  0272 DD7301        1118           LD [IX+1],E             ;set NEXT_NOTE_PTR for first note in song
  0275 DD7202        1119           LD [IX+2],D
  0278 CD035F        1120           CALL LOAD_NEXT_NOTE     ;load note, byte 0 := CH#|SONGNO, set new NEXT_NOTE_PTR
  027B CD0295        1121           CALL UP_CH_DATA_PTRS    ;new song, so update channel data ptrs
  027E C9            1122           RET
                     1123 ;       END ;JUKEBOX
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1124 ;       [page]
                     1125  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1127 ;        .IDENT SNDMAN           ;includes SND_MANAGER,PROCESS_DATA_AREA,
                     1128                                  ;UP_CH_DATA_PTRS
                     1129 ;*****************************
                     1130 ;*       SND_MANAGER         *
                     1131 ;*****************************
                     1132 ;.COMMENT )
                     1133 ;See Users' Manual for description
                     1134 ;)
                     1135         GLB SND_MANAGER
                     1136        ;EXT PT_IX_TO_SxDATA,AREA_SONG_IS
                     1137        ;INCLUDE OSSR_EQU:OS  ;equates
     027F            1138 SND_MANAGER
                     1139 ;       * IX := addr of song #1 data area [S1DATA]
     027F 0601       1140         LD B,1                ;pt IX to byte 0 song data area for song # 1
     0281 CD01C1     1141         CALL PT_IX_TO_SxDATA
                     1142 ;       LOOP until end of song data areas
     0284 3E00       1143 L1        LD A,ENDSDATA         ;check for end of song data areas
     0286 DDBE00     1144           CP [IX+0]             ;set Z flag if inactive
     0289 C8         1145           RET Z                 ;leave [Z set], if all data areas have been processed
                     1146 ;           * process active song data areas
     028A CD02D6     1147             CALL PROCESS_DATA_AREA  ;update counters or call effect; get next note
                     1148 ;           * point IX to byte 0 next song data area
     028D 1E0A       1149             LD E,10
     028F 1600       1150             LD D,0
     0291 DD19       1151             ADD IX,DE
     0293 18EF       1152         JR L1 ;REPEAT LOOP
                     1153 ;*****************************
                     1154 ;*     UP_CH_DATA_PTRS     *
                     1155 ;*****************************
                     1156 ;.COMMENT )
                     1157 ;For each active data area, starting with S1DATA and proceeding in order, load
                     1158 ;the associated channel data area pointer [PTR_TO_S_ON_x] with the address of
                     1159 ;byte 0.  This routine is called by JUKE_BOX, when a song starts and
                     1160 ;PROCESS_DATA_AREA when the channel using a data area has changed as a result of
                     1161 ;calling LOAD_NEXT_NOTE [this happens when a song finishes and when it switches
                     1162 ;back and forth between noise and tone notes].
                     1163 ;)
                     1164         GLB UP_CH_DATA_PTRS
                     1165        ;EXT DUMAREA
     0295            1166 UP_CH_DATA_PTRS
     0295 DDE5       1167         PUSH IX               ;save curent IX
     0297 21024C     1168         LD HL,DUMAREA         ;set all 4 ch data ptrs to dummy inactive area
     029A 227022     1169         LD [PTR_TO_S_ON_0],HL
     029D 227024     1170         LD [PTR_TO_S_ON_1],HL
     02A0 227026     1171         LD [PTR_TO_S_ON_2],HL
     02A3 227028     1172         LD [PTR_TO_S_ON_3],HL
     02A6 0601       1173         LD B,1                ;set IX to byte 0 S1DATA
     02A8 CD01C1     1174         CALL PT_IX_TO_SxDATA  ;RETs with IX addr byte 0 song 1
                     1175 ;       LOOP until end of song data areas
     02AB DD7E00     1176 L2        LD A,[IX+0]           ;test for end of song data araes
     02AE FE00       1177           CP ENDSDATA
     02B0 2821       1178           JR Z,DONE_SNDMAN        ;leave loop if all data areas checked
                     1179 ;           * if area active, set appropriate channel data area pointer
     02B2 FEFF       1180             CP INACTIVE           ;check for inactive data area: don't up date ptr if so
                     1181 ;           IF [PSW,IS,NZERO]     ;area is active: update channel data ptrs
     02B4 2815       1182             JR Z,L9
     02B6 DD7E00     1183               LD A,[IX+0]           ;get CH# in A
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    02B9 E6C0       1184              AND 0C0H                ;B7 - B6  in A  = CH#
    02BB 07         1185              RLCA                    ;form CH#  * 2  in A, i.e., the offset from
    02BC 07         1186              RLCA                    ;PTR_TO_S_ON_0  of the  channel data area pointer
    02BD 07         1187              RLCA                    ;that points  to channel  CH#
    02BE 5F         1188              LD E,A                  ;add offset to addr of PTR_TO_ S_ON_0
    02BF 1600       1189              LD D,0
    02C1 217022     1190              LD HL,PTR_TO_S_ON_0
    02C4 19         1191              ADD HL,DE               ;HL points to  proper channel  data area pointer
    02C5 DDE5       1192              PUSH IX                 ;store this song data area's byte  0  addr there
    02C7 D1         1193              POP DE
    02C8 73         1194              LD (HL),E
    02C9 23         1195              INC HL
    02CA 72         1196              LD (HL),D
                    1197 ;          ENDIF
                    1198 ;          * point IX to byte 0 next song data area
    02CB 1E0A       1199 L9         LD E,10
    02CD 1600       1200            LD D,0
    02CF DD19       1201            ADD IX,DE
    02D1 18D8       1202          JR L2 ;REPEAT LOOP
    02D3            1203 DONE_SNDMAN
    02D3 DDE1       1204        POP IX                  ;restore IX
    02D5 C9         1205        RET
                    1206 ;*************************
                    1207 ;*   PROCESS_DATA_AREA     *
                    1208 ;*************************
                    1209 ;.COMMENT )
                    1210 ;See Users' Manual for description
                    1211 ;Terminology: SFX = address of sound effect routine
                    1212 ;)
                    1213        GLB PROCESS_DATA_AREA,EFXOVER
                    1214       ;EXT LOAD_NEXT_NOTE,ATN_SWEEP,FREQ_SWEEP
    02D6            1215 PROCESS_DATA_AREA
    02D6 CD01E9     1216        CALL AREA_SONG_IS       ;return area's SONGNO in A [and addr SFX in HL]
    02D9 FEFF       1217        CP INACTIVE             ;test for inactive code
    02DB C8         1218        RET Z                   ;RET, no processing if area inactive
                    1219 ;        * if special effect, call it to process the data area
    02DC FE3E       1220          CP 62                   ;test for special sound effect
                    1221 ;        IF [PSW,IS,ZERO]          ;data area used by sound effect
    02DE 2006       1222          JR NZ,L10
    02E0 1E07       1223            LD E,7                  ;pt HLto SFX+7, starting adr of the effect's cod e
    02E2 1600       1224            LD D,0
    02E4 19         1225            ADD HL,DE
    02E5 E9         1226            JP (HL)                 ;do 1 pass thru effect, RET from effect
                    1227 ;        ENDIF
                    1228 ;          * else process a non-effect note
    02E6 CD012F     1229 L10        CALL ATN_SWEEP          ;process atn sweep data, if any
    02E9 CD00FC     1230            CALL FREQ_SWEEP         ;proc frq sweep data, if any, & note dura timer s
                    1231 ;          IF [PSW,IS,ZERO]          ;note is over
    02EC 2011       1232            JR NZ,L12
    02EE DD7E00     1233 EFXOVER      LD A,[IX+0]           ;A := CH# | SONGNO this note
    02F1 F5         1234              PUSH AF               ;save on stack
    02F2 CD035F     1235              CALL LOAD_NEXT_NOTE   ;load data for next note
    02F5 C1         1236              POP BC                ;B := CH# | SONGNO previous note
    02F6 DD7E00     1237              LD A,[IX+0]           ;A := CH# | SONGNO new note [may be inactive]
    02F9 B8         1238              CP B                  ;check against new note's CH# | SONGNO
                    1239 ;            IF [PSW,IS,NZERO]     ;change to/from tone/efx/noise
    02FA 2803       1240              JR Z,L12
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    02FC CD0295      1241                 CALL UP_CH_DATA_PTRS    ;to maintain data area priority system
                     1242 ;             ENDIF
                     1243 ;           ENDIF
    02FF C9          1244 L12         RET
                     1245 ;       END ;SNDMAN
                     1246  PROG
```

```
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

                     1305 ;       ELSE
   0339 CD0164       1306 L5        CALL UPATNCTRL          ;send out current ATN
   033C DD7E04       1307           LD A,[IX+CTRL]          ;LSN A = current CTRL data
   033F E60F         1308           AND 0FH                 ;mask MSN
   0341 21702A       1309           LD HL,SAVE_CTRL         ;point to last CTRL data sent
   0344 BE           1310           CP [HL]                 ;compare
                     1311 ;         IF [PSW,IS,NZERO]       ;CTRL has changed
   0345 2806         1312           JR Z,L6
   0347 77           1313             LD [HL],A               ;SAVE_CTRL = new CTRL data
   0348 0EE0         1314             LD C,SRNCTL             ;send new CTRL data
   034A CD0164       1315             CALL UPATNCTRL
                     1316 ;         ENDIF
                     1317 ;       ENDIF
   034D C9           1318 L6      RET
   034E              1319 TONE_OUT
   034E DD5E00       1320         LD E,[IX+0]             ;look for inactive code, 0FFH
   0351 1C           1321         INC E                   ;this sets Z flag if E = 0FFH
                     1322 ;       IF [PSW,IS,ZERO]        ;song data area is inactive
   0352 2004         1323         JR NZ,L7
   0354 D3FF         1324           OUT [SOUND_PORT],A      ;turn off CHx
   0356 1806         1325           JR L8
                     1326 ;       ELSE                    ;send out current ATN and FREQ
   0358 CD0164       1327 L7        CALL UPATNCTRL          ;send out attenuation
   035B CD0175       1328           CALL UPFREQ             ;send out frequency
                     1329 ;       ENDIF
   035E C9           1330 L8      RET
                     1331 ;       END ;PLAYSON
                     1332  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    1334 ;        .IDENT LOADNEX  ;includes LOAD_NEXT_NOTE
                    1335 ;****************************
                    1336 ;*     LOAD_NEXT_NOTE       *
                    1337 ;****************************
                    1338 ;.COMMENT )
                    1339 ;see Users' Manual for description
                    1340 ;SFX refers to the beginning address of a special sound effect routine
                    1341 ;)
                    1342         GLB LOAD_NEXT_NOTE,REST,ENDREP,ENDNOREP,EFFECT,TYPE0,TYPE1,TYPE2,TYPE3
                    1343         GLB MODB0,DE_TO_DEST,PASS1
                    1344        ;EXT JUKE_BOX,UP_CH_DATA_PTRS
                    1345        ;INCLUDE OSSR_EQU:OS:0  ;equates
    035F            1346 LOAD_NEXT_NOTE
                    1347 ;       * deactivate area, save SONGNO on stack
    035F DD7E00     1348         LD A,[IX+0]             ;A := byte 0
    0362 E63F       1349         AND 00111111B           ;mask CH#, if any
    0364 F5         1350         PUSH AF                 ;save SONGNO on stack
    0365 DD3600FF   1351         LD [IX+0],INACTIVE      ;deactivate area
                    1352 ;       * A := header new note
    0369 DD6E01     1353         LD L,[IX+1]             ;HL := addr new note in ROM
    036C DD6602     1354         LD H,[IX+2]
    036F 7E         1355         LD A,[HL]               ;A := header new note
                    1356 ;       * save header of new note in song on stack and load its data CASE note type
    0370 47         1357         LD B,A                  ;save header new note in B
                    1358 ;       - test for rest
    0371 CB6F       1359         BIT 5,A                 ;test for rest
                    1360 ;       IF [PSW,IS,NZERO]       ;note is a rest
    0373 281C       1361         JR Z,L13
                    1362           ;--CASE-- rest
    0375 C5         1363 REST      PUSH BC                 ;save header on stack
    0376 E61F       1364           AND 00011111B           ;mask all but duration bits
                    1365 ;         * set up NEXT_NOTE_PTR
    0378 23         1366           INC HL                  ;HL = address of the header of the note after this note
    0379 DD7501     1367           LD [IX+1],L             ;store in NEXT_NOTE_PTR
    037C DD7402     1368           LD [IX+2],H
                    1369 ;         * move this note's data and fill in bytes where necessary
    037F DD3604F0   1370           LD [IX+ATN],0F0H        ;set atn off
    0383 DD7705     1371           LD [IX+NLEN],A          ;NLEN := 5 bit duration
    0386 DD360700   1372           LD [IX+FSTEP],0         ;indicate freq not to be swept
    038A DD360800   1373           LD [IX+ASTEP],0         ;indicate atn not to be swept
    038E C30461     1374           JP MODB0
                    1375 ;       ENDIF
                    1376 ;       - test for end of song
    0391 CB67       1377 L13     BIT 4,A                 ;test for end
                    1378 ;       IF [PSW,IS,NZERO]       ;end of song
    0393 280F       1379         JR Z,L14
    0395 CB5F       1380           BIT 3,A                 ;test for repeat
                    1381 ;         IF [PSW,IS,NZERO]       ;repeat song
    0397 2805       1382           JR Z,ENDNOREP
                    1383             ;--CASE-- end of song, repeat
    0399 C1         1384 ENDREP      POP BC                  ;B := SONGNO
    039A CD025E     1385             CALL JUKE_BOX           ;to reload 1st note of this song
    039D C9         1386             RET                     ;to PROCESS_DATA_AREA, don't save header
                    1387 ;         ENDIF
                    1388             ;--CASE-- end of song, no repeat
    039E 3EFF       1389 ENDNOREP    LD A,INACTIVE
    03A0 F5         1390             PUSH AF                 ;save inactive code to end song
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    03A1 C30461      1391            JP MODBO                ;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,MODBO           ;RET to MODBO
    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
    03F4 DD360800    1445           LD [IX+ASTEP],0       ;set for no atn sweep
    03F8 1867        1446           JR MODBO
                     1447 ;       ENDIF
```

```
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 MODBO
                     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
    042A DD7501      1482 L18       LD [IX+1],L             ;put addr in NEXT_NOTE_PTR
    042D 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 MODBO
                     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

    044D 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

                     1545 ;       .IDENT  ACTIVATE
                     1546 ;       .ZOP
                     1547 ;       .EPOP
                     1548 ;       .COMMENT )
                     1549 ;********************* ACTIVATE ***********************************************
                     1550 ;
                     1551 ;                                                                  4/22/82
                     1552 ;                                                                  13:50:00
                     1553 ;
                     1554 ;THE FOLLOWING CHANGES/REVISIONS WERE MADE:
                     1555 ;
                     1556 ;       1. ELIMINATE CODE PLACING OLD.SCREEN ADDRESS IN STATUS AREA
                     1557 ;       2. INIT X.PAT.POS IN OLD.SCREEN WHEN IN VRAM AS WELL AS WHEN IN CRAM
                     1558 ;       3. USE VDP.MODE.WORD TO TEST GRAPHICS MODE
                     1559 ;       4. ADD CODE TO EXPAND ONE COLOR GENERATOR BYTE TO 8
                     1560 ;       5. ADDED C_BUFF DEFS 8 FOR COLOR EXPANDING CODE
                     1561 ; 5/02  6. FIX COLOR GEN MOVE IN MODE I
                     1562 ;       7. USE CONTROLER_MAP FOR BUFFER AREA
                     1563 ;
                     1564 *  ACTIVATE is used to initialize the RAM status area for the passed
                     1565 *object and move its pattern and color generators to the PATTERN and
                     1566 *COLOR GENERATOR tables in VRAM_  The second function is enabled or
                     1567 *disabled by setting or resetting the carry flag in the PSW_  this is
                     1568 *necessary to prevent sending the same graphics data to VRAM more than
                     1569 *once when creating identical objects_  The calling sequence for act-
                     1570 *ivating an object is as follows:
                     1571 *
                     1572 *               LD              HL,OBJ_n        ;->OBJ TO ACTIVATE
                     1573 *               SCF                             ;SIGNAL MV TO VRAM
                     1574 *               CALL            ACTIVATE
                     1575 *
                     1576 *
                     1577 *OR
                     1578 *
                     1579 *               LD              HL,OBJ_n        ;->OBJ TO ACTIVATE
                     1580 *               OR              A               ;DON'T MV TO VRAM
                     1581 *               CALL            ACTIVATE
                     1582 *
                     1583 *)
                     1584                ;EXT            PUT_VRAM_,VRAM_WRITE,VDP_MODE_WORD
                     1585                ;EXT            WORK_BUFFER
                     1586 ;
                     1587                GLB             ACTIVATE_
                     1588 ;
                     1589 ;REGISTER USAGE: FOLLOWING WILL BE CHANGED BY ACTIVATE, ADDITIONAL
                     1590 ;MAY BE CHANGED BY CALLED SUBR
                     1591 ;               AF,HL,DE,BC,IY
                     1592 ;
                     1593
                     1594
                     1595 ; PROCEDURE ACTIVATEQ[VAR OBJ:OBJECT;MOVE:BOOLEAN];
                     1596
                     1597 ; ACTIVATEQ IS THE PASCAL ENTRY POINT TO ACTIVATE
                     1598
                     1599               ;EXT     PARAM_
                     1600 ; THE PASCAL PARAMETER PASSING PROCEDURE
                     1601 ;               COMN
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     1602
                     1603 ;PRM_AREA:     DEFS            3       ;Moved to OS
                     1604 ; THIS IS THE COMMON PARAMETER PASSING AREA
                     1605
                     1606                PROG
 0482 0002FFFE       1607 ACTIVATE_P:    DEFW            2,-2,1
 0486 0001
                     1608
                     1609                GLB             ACTIVATEQ
          <0488>     1610 ACTIVATEQ      EQU             $
 0488 010482         1611                LD              BC,ACTIVATE_P
 048B 1173BA         1612                LD              DE,PRM_AREA
 048E CD0098         1613                CALL            PARAM_
 0491 2A73BA         1614                LD              HL,[PRM_AREA]
 0494 5E             1615                LD              E,[HL]
 0495 23             1616                INC             HL
 0496 56             1617                LD              D,[HL]
 0497 EB             1618                EX              DE,HL
 0498 3A73BC         1619                LD              A,[PRM_AREA+2]
 049B FE00           1620                CP              0
 049D 2803           1621                JR              Z,NTZZZ_
 049F 37             1622                SCF
 04A0 1801           1623                JR              TZZZ_
 04A2 B7             1624 NTZZZ_:         OR              A
 04A3                1625 TZZZ_:
                     1626
          <04A3>     1627 ACTIVATE_      EQU             $
                     1628 ;SUP POINTERS ETC_ COMMON TO ALL SUBCASES
                     1629 ;              HL->OBJ DEF CROM
                     1630 ;              C FLG=SUP VRAM FLG
 04A3 5E             1631                LD              E,[HL]          ;->OBJ GEN CROM
 04A4 23             1632                INC             HL
 04A5 56             1633                LD              D,[HL]
 04A6 23             1634                INC             HL
 04A7 4E             1635                LD              C,[HL]          ;->OBJ CRAM
 04A8 23             1636                INC             HL
 04A9 46             1637                LD              B,[HL]
 04AA 23             1638                INC             HL
 04AB 3E00           1639                LD              A,0             ;ZERO FRAME
 04AD 02             1640                LD              [BC],A
 04AE 1A             1641                LD              A,[DE]          ;GET OBJ_TYPE
 04AF F5             1642                PUSH            AF              ;SV OBJ_TYPE & FLG
 04B0 E60F           1643                AND             0FH             ;GET OBJ_TYPE NUM
 04B2 CA04E7         1644                JP              Z,ACT_SEMI                  ;TYPE=0
 04B5 3D             1645                DEC             A
 04B6 CA05F1         1646                JP              Z,ACT_MOBILE                ;TYPE=1
 04B9 3D             1647                DEC             A
 04BA CA0600         1648                JP              Z,ACT_OSPRT                 ;TYPE=2
 04BD 3D             1649                DEC             A
 04BE CA0600         1650                JP              Z,ACT_1SPRT                 ;TYPE=3
 04C1 3D             1651                DEC             A
 04C2 2802           1652                JR              Z,ACT_CMPLX                 ;TYPE=4
 04C4 F1             1653                POP             AF              ;SUBCASE ELSE
 04C5 C9             1654                RET
                     1655 ;ON ENTRY TO SUBCASES:
                     1656 ;              STACK=OBJ_TYPE & SUP VRAM FLG
                     1657 ;              HL->OBJ_n+4
```

```
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 20DC        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 200D        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 2A73D1      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 2273D1      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 2A73D1      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 1173BA      2175         LD   DE,PARAM_AREA
    06CD CD0098      2176         CALL PARAM_
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    06D0 DD2A738A    2177         LD   IX,[PARAM_AREA]
    06D4 3A738C      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 CA08DF      2197         JP Z,PUTOSPRITE         ;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 CD0808         2257        CALL PUTFRAME
073A C9             2258        RET
                    2259
0738                2260 S_OLD_SCRN
                    2261
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0738 C5          2262         PUSH BC          ;SAVE REGS
    073C D5          2263         PUSH DE
    0730 E5          2264         PUSH HL
                     2265
    073E FE70        2266               CP              70H
    0740 2802        2267               JR              Z,EQUAL_TO
    0742 3807        2268               JR              C,ELSE_1
                     2269
                     2270 ;       IF [.A,GE,70H]        ;THEN OLD_SCREEN IN CPU RAM
    0744             2271 EQUAL_TO
    0744 67          2272           LD H,A
    0745 DD6E04      2273           LD L,[IX+4]           ;HL := OLD_SCREEN ADDRESS
    0748 7E          2274           LD A,[HL]
                     2275
    0749 1835        2276               JR              END_IF_1
    074B             2277 ELSE_1
                     2278
                     2279 ;       ELSE                  ;OLD_SCREEN IN VRAM
    074B 2A8006      2280           LD HL,[WORK_BUFFER]   ;GET ADDRESS OF FREE BUFFER SPACE
    074E DD5605      2281           LD D,[IX+5]           ;DE := OLD_SCREEN ADDRESS
    0751 DD5E04      2282           LD E,[IX+4]
    0754 E5          2283           PUSH HL               ;SAVE 2 COPIES FREE BUFFER ADDR
    0755 D5          2284           PUSH DE               ;SAVE OLD_SCREEN ADDR
    0756 E5          2285           PUSH HL
    0757 010004      2286           LD BC,4               ;READ 4 BYTES [X,Y_PAT_POSs, X,Y_EXTETNs]
    075A CD1D3E      2287           CALL VRAM_READ
    075D E1          2288           POP HL                ;HL := FREE BUFFER ADDR
    075E 7E          2289           LD  A,[HL]
    075F FE80        2290           CP  80H
    0761 2003        2291           JR  NZ,GET_OLD
    0763 D1          2292           POP  DE
    0764 1819        2293           JR  SKIP_OLD
    0766 23          2294 GET_OLD   INC HL
    0767 23          2295           INC HL
    0768 46          2296           LD B,[HL]             ;B := X_EXTENT OF OLD_SCREEN
    0769 23          2297           INC HL
    076A 5E          2298           LD E,[HL]             ;E := Y_EXTENT
    076B 1600        2299           LD D,0
    076D 23          2300           INC HL
    076E EB          2301           EX DE,HL              ;MULTIPLY X_EXTENT*Y_EXTENT IN HL
                     2302
    076F 1801        2303               JR              M_XY+1
    0771 29          2304 M_XY:         ADD             HL,HL
    0772 10FD        2305               DJNZ            M_XY
                     2306
                     2307 ;         DJNZ M.XY
    0774 E5          2308           PUSH HL
    0775 C1          2309           POP BC                ;BC := NUMBER OF BYTES TO READ
    0776 EB          2310           EX DE,HL              ;HL := FREE BUFF ADDR + 4
    0777 D1          2311           POP DE                ;DE := OLD_SCREEN ADDR.
    0778 13          2312           INC DE
    0779 13          2313           INC DE
    077A 13          2314           INC DE
    077B 13          2315           INC DE
    077C CD1D3E      2316           CALL VRAM_READ         ;READ SAVED NAMES FOR BACKGROUND
    077F E1          2317 SKIP_OLD  POP HL                ;HL := FREE BUFF ADDR.
                     2318 ;       ENDIF
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2319
0780                 2320 END_IF_1
                     2321
0780 7E              2322          LD A,[HL]               ;A := X_PAT_POS
                     2323
                     2324
0781 FE80            2325               CP                80H
0783 280F            2326               JR                Z,END_IF_2
                     2327
                     2328 ;        IF [.A,NE,80H]         ;THEN THERE IS AN OLD_SCREEN
0785 5E              2329            LD E,[HL]             ;E := X_PAT_POS
0786 23              2330            INC HL
0787 56              2331            LD D,[HL]             ;D := Y_PAT_POS
0788 23              2332            INC HL
0789 4E              2333            LD C,[HL]             ;C := X_EXTENT
078A 23              2334            INC HL
078B 46              2335            LD B,[HL]             ;B := Y_EXTENT
078C 23              2336            INC HL                ;HL POINTS TO FIRST NAME IN LIST
                     2337
078D DDE5            2338               PUSH               IX              ;SAVE OBJECT POINTER
                     2339
078F CD080B          2340            CALL PUTFRAME         ;RESTORE OLD_SCREEN TO DISPLAY
                     2341
0792 DDE1            2342               POP                IX              ;RESTORE OBJECT POINTER
                     2343
0794                 2344 END_IF_2
                     2345
                     2346 ;        ENDIF
0794 E1              2347 SV1:     POP HL                  ;HL := ADDR OF FIRST NAME IN FRAME
0795 D1              2348          POP DE                  ;DE := Y,X_PAT_POSs
0796 C1              2349          POP BC                  ;BC := Y,X_EXTENTs
0797 C5              2350          PUSH BC
0798 D5              2351          PUSH DE
0799 E5              2352          PUSH HL
079A DD6605          2353          LD H,[IX+5]             ;HL := OLD_SCREEN ADDRESS
079D DD6E04          2354          LD L,[IX+4]
                     2355
07A0 3E70            2356               LD                A,70H
07A2 BC              2357               CP                H
07A3 3803            2358               JR                C,END_IF_3
                     2359
                     2360 ;        IF [.H,LT,70H]         ;THE OLD_SCREEN NOW IN FREE BUFFER
07A5 2A8006          2361            LD HL,[WORK_BUFFER]   ;THEREFORE, MOVE BACKGROUND TO BUFFER
                     2362
07A8                 2363 END_IF_3
                     2364
                     2365 ;        ENDIF
07A8 73              2366          LD [HL],E               ;OLD_SCREEN +   0 := X_PAT_POS
07A9 23              2367          INC HL
07AA 72              2368          LD [HL],D               ;       "       1 := Y_PAT_POS
07AB 23              2369          INC HL
07AC 71              2370          LD [HL],C               ;       "       2 := X_EXTENT
07AD 23              2371          INC HL
07AE 70              2372          LD [HL],B               ;       "       3 := Y_EXTENT
07AF 23              2373          INC HL                  ;HL := ADDRESS TO STORE NAMES
                     2374
07B0 DDE5            2375               PUSH               IX              ;SAVE OBJECT POINTER
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2376
    07B2 CD0898      2377          CALL GET_BKGRND
                     2378
    07B5 DDE1        2379                POP           IX              ;RESTORE OBJECT POINTER
                     2380
    07B7 E1          2381          POP HL                ;WHERE NAMES ARE IN CPU RAM
    07B8 D1          2382          POP DE                ;WHERE TO MOVE THEM TO IN VRAM (NAME TABLE)
    07B9 C1          2383          POP BC                ;HOW MANY TO MOVE
                     2384
    07BA DDE5        2385                PUSH          IX              ;SAVE OBJECT POINTER
                     2386
    07BC CD0808      2387          CALL PUTFRAME
                     2388
    07BF DDE1        2389                POP           IX              ;RESTORE OBJECT POINTER
                     2390
    07C1 DD5605      2391          LD D,(IX+5)           ;SEE IF SAVED BACKGROUND TO BE MOVED TO VRAM
                     2392
    07C4 3E70        2393 SV2:           LD            A,70H
    07C6 BA          2394                CP            D
    07C7 281E        2395                JR            Z,END_IF_4
    07C9 381C        2396                JR            C,END_IF_4
                     2397
                     2398 ;SV2::    IF (.D,LT,70H)
    07CB DD5E04      2399           LD E,(IX+4)           ;DE := OLD_SCREEN ADDR
    07CE D9          2400           EXX                   ;USE 'REG FOR CALCULATION
    07CF 2A8006      2401           LD HL,(WORK_BUFFER)   ;WHERE NEXT OLD_SCREEN DATA IS
    07D2 E5          2402           PUSH HL
    07D3 23          2403           INC HL
    07D4 23          2404           INC HL
    07D5 5E          2405           LD E,(HL)             ;E := X_EXTENT
    07D6 1600        2406           LD D,0
    07D8 23          2407           INC HL
    07D9 46          2408           LD B,(HL)             ;B := Y_EXTENT
    07DA EB          2409           EX DE,HL              ;HL := X_EXTENT
    07DB 1801        2410           JR M_XY2+1
    07D0 29          2411 M_XY2:    ADD HL,HL             ;HL := X_EXTENT*Y_EXTENT
    07DE 10FD        2412           DJNZ M_XY2
    07E0 E5          2413           PUSH HL
    07E1 D9          2414           EXX
    07E2 C1          2415           POP BC                ;BC := NUMBER OF BYTES TO WRITE
    07E3 E1          2416           POP HL                ;HL := FREE BUFFER ADDRESS
    07E4 CD1D01      2417           CALL VRAM_WRITE
                     2418
    07E7             2419 END_IF_4
                     2420
                     2421 ;         ENDIF
    07E7 C9          2422           RET
                     2423
                     2424 ****************** PX.TO.PTRN.POS ***************************************
                     2425 ;DESCRIPTION:    DIVIDES REG DE BY 8, IF SIGNED RESULT > 127 THEN E := MAX SIGNED
                     2426 ;                POSITIVE NUMBER. IF RESULT < -128, THEN E := MIN NEGATIVE NUM
                     2427 ;INPUT:          DE = 16 BIT SIGNED NUMBER
                     2428 ;OUTPUT:         DE/8 < -128      E = -128
                     2429 ;        -128 <= DE/8 <=+127      E = DE/8
                     2430 ;        +127 <  DE/8             E = +127
                     2431 *************************************************************************
                     2432 ;
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2433
                     2434                    GLB             PX_TO_PTRN_POS
   07E8              2435 PX_TO_PTRN_POS
                     2436
   07E8 E5           2437          PUSH HL                   ;HL USED TO TEST MAGNITUDE
   07E9 CB2A         2438          SRA D                     ;16 BIT SHIFT LEFT
   07EB CB1B         2439          RR E
   07ED CB2A         2440          SRA D
   07EF CB1B         2441          RR E
   07F1 CB2A         2442          SRA D                     ;               X3
   07F3 CB1B         2443          RR E
   07F5 CB7A         2444          BIT 7,D                   ;IS RESULT NEGATIVE
   07F7 2009         2445          JR NZ,NEGTV
   07F9 21FF80       2446          LD HL,0FF80H              ;SEE IF RESULT < 127
   07FC 19           2447          ADD HL,DE
   07FD E1           2448          POP HL
   07FE D0           2449          RET NC
   07FF 1E7F         2450          LD E,7FH                  ;IF > 128, THEN E := MAX SIGNED + NUM.
   0801 C9           2451          RET
                     2452 ;XXXXX
   0802 210080       2453 NEGTV:   LD HL,080H                ;IS RESULT > -128
                     2454 ;XXXXX
   0805 19           2455          ADD HL,DE
   0806 E1           2456          POP HL
   0807 D8           2457          RET C
   0808 1E80         2458          LD E,80H                  ;IF < -128, THE E := MIN SIGNED - NUM.
   080A C9           2459          RET
                     2460 ;
                     2461 ************************** PUT FRAME ***************************************
                     2462 ;DESCRIPTION:    THE NAMES WHICH CONSTITUTE A FRAME ARE MOVED TO THE NAME TABLE
                     2463 ;               IN VRAM.  THE UPPER LEFT HAND CORNER OF THE FRAME IS POSITIONED
                     2464 ;               AT X_PAT_POS, Y_PAT_POS.
                     2465 ;INPUT:         HL = ADDRESS OF LIST OF NAMES (IN CPU RAM)
                     2466 ;               E = X_PAT_POS
                     2467 ;               D = Y_PAT_POS
                     2468 ;               C = X_EXTENT
                     2469 ;               B = Y_EXTENT
                     2470 ****************************************************************************
                     2471 ;
                     2472                    GLB             PUTFRAME
   080B C5           2473 PUTFRAME:  PUSH BC                 ;COPY PARAMETERS INTO PRIMED REGISTERS
   080C D5           2474          PUSH DE
   080D E5           2475          PUSH HL                   ;AND FRAME ADDRESS INTO DE'
   080E D9           2476          EXX
   080F E1           2477          POP HL
   0810 D1           2478          POP DE
   0811 C1           2479          POP BC
   0812 CD08C0       2480          CALL CALC_OFFSET
   0815 D9           2481          EXX
                     2482 ;XXXXX
                     2483 ; TEST FOR THE FOLLOWING CONDITION: (X_PAT_POS SLE 32) AND (X_PAT_POS + X_EXTENT
                     2484 ;                                                    SGT 0)
   0816 7B           2485 PF1:     LD A,E                    ;IS X_PAT_POS < 0?
   0817 CB7F         2486          BIT 7,A
   0819 2003         2487          JR NZ,XP_NEG
   081B FE20         2488          CP 32                     ;IS X_PAT_POS < 32?
   081D D0           2489          RET NC                    ;IF NOT, RET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    081E 81          2490 XP_NEG: ADD A,C              ;A := X_PAT_POS + X_EXTENT
    081F CB7F        2491         BIT 7,A              ;IS A NEG?
    0821 C0          2492         RET NZ               ;YES, RETURN
    0822 B7          2493         OR A                 ;A = 0?
    0823 C8          2494         RET Z                ;RETURN IF 0
                     2495 ;XXXXX
                     2496 ;X.IN.BOUNDS::
                     2497
    0824             2498 X_IN_BOUNDS
                     2499
                     2500 ;       IF [.E,IS,MINUS]     ;IF X_PAT_POS < 0 , FRAME BLEEDING ON FROM LEFT
                     2501
    0824 CB7B        2502                BIT           7,E
    0826 2820        2503                JR            Z,ELSE_8
                     2504
    0828 79          2505           LD A,C               ;CALCULATE AMOUNT OF FRAME ON SCREEN
    0829 83          2506           ADD A,E              ;A := X_EXTENT + X_PAT_POS
    082A D5          2507           PUSH DE
                     2508 ;XXXXX
    082B FE21        2509           CP 33                ;IF NUMBER OF NAMES > 32
    082D 3802        2510           JR C,LT33
    082F 3E20        2511           LD A,32              ;THEN NUMBER OF NAMES := 32
    0831 5F          2512 LT33:     LD E,A               ;NUMBER OF NAMES ON SCREEN EACH ROW
                     2513 ;XXXXX
    0832 1600        2514           LD D,0
    0834 D5          2515           PUSH DE              ;GET COUNT INTO IY
    0835 FDE1        2516           POP IY
    0837 D1          2517           POP DE               ;RESTORE DE
    0838 7B          2518           LD A,E               ;A := X_PAT_POS
    0839 D9          2519           EXX                  ;NOW ADJUST STARTING POINTS IN FRAME LIST AND
                     2520                                ;NAME TABLE
    083A C5          2521           PUSH BC              ;SAVE X AND Y EXTENT
    083B ED44        2522           NEG                  ;2'S COMPLIMENT OF X_PAT_POS
    083D 4F          2523           LD C,A
    083E 0600        2524           LD B,0
    0840 09          2525           ADD HL,BC            ;ADD DISPLACEMENT TO FRAME POINTER
    0841 EB          2526           EX DE,HL
    0842 09          2527           ADD HL,BC            ;ADD DISPLACEMENT TO NAME TABLE POINTER
    0843 EB          2528           EX DE,HL
    0844 C1          2529           POP BC
    0845 D9          2530           EXX
                     2531
    0846 181C        2532                JR            END_IF_8
                     2533
                     2534 ;       ELSE
                     2535
    0848             2536 ELSE_8
                     2537
    0848 7B          2538 PF2:      LD A,E               ;IS X_PAT_POS + X_EXTENT > 31
    0849 81          2539           ADD A,C
                     2540 ;         IF [.A,GT,31]
                     2541
    084A FE1F        2542                CP            31
    084C 280F        2543                JR            Z,ELSE_9
    084E 3800        2544                JR            C,ELSE_9
                     2545
    0850 3E20        2546             LD A,32              ;SUBTRACT X_PAT_POS FROM 31
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0852 93          2547            SUB E
    0853 D5          2548            PUSH DE
    0854 5F          2549            LD E,A                  ;GET THIS NUMBER INTO IY
    0855 1600        2550            LD D,0
    0857 D5          2551            PUSH DE
    0858 FDE1        2552            POP IY
    085A D1          2553            POP DE
                     2554
    085B 1807        2555                JR              END_IF_9
                     2556
                     2557 ;          ELSE                    ;BOTH ENDS OF FRAME WITHIN PATTERN PLANE
                     2558
    085D             2559 ELSE_9
                     2560
    085D C5          2561 PF3:       PUSH BC
    085E 0600        2562            LD B,0
    0860 C5          2563            PUSH BC
    0861 FDE1        2564            POP IY
    0863 C1          2565            POP BC
                     2566 ;          ENDIF
                     2567
    0864             2568 END_IF_9
                     2569
                     2570 ;        ENDIF
                     2571
    0864             2572 END_IF_8
                     2573
    0864 1E00        2574          LD E,0
                     2575 ;        REPEAT                  ;Y_EXTENT-1 TIMES
                     2576
    0866             2577 RPT_1
                     2578
    0866 7A          2579 PF4:       LD A,D                  ;GET Y_PAT_POS
    0867 83          2580            ADD A,E                 ;ADD Y
                     2581 ;          IF [.A,IS,PLUS]
                     2582
    0868 CB7F        2583                BIT             7,A
    086A 2019        2584                JR              NZ,END_IF_10
                     2585
                     2586 ;            IF [.A,LE,23]           ;IS 0<=Y_PAT_POS + Y <=23
                     2587
    086C FE18        2588                CP              24
    086E 3015        2589                JR              NC,END_IF_10
                     2590
    0870 C5          2591              PUSH BC
    0871 D5          2592              PUSH DE
    0872 D9          2593              EXX
    0873 C5          2594              PUSH BC
    0874 D5          2595              PUSH DE
    0875 E5          2596              PUSH HL
    0876 FDE5        2597              PUSH IY
    0878 3E02        2598              LD A,2                  ;CODE FOR PATTERN NAME TABLE ADDED 4/20
    087A CD1C27      2599              CALL PUT_VRAM_
    087D FDE1        2600              POP IY
    087F E1          2601              POP HL
    0880 D1          2602              POP DE
    0881 C1          2603              POP BC
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0882 D9          2604              EXX
    0883 D1          2605              POP DE
    0884 C1          2606              POP BC
                     2607 ;          ENDIF
                     2608 ;        ENDIF
                     2609
    0885             2610 END_IF_10
                     2611
    0885 D9          2612          EXX
    0886 C5          2613          PUSH BC
    0887 0600        2614          LD B,0                ;INCREMENT POINTER INTO FRAME BY X_EXTENT
    0889 09          2615          ADD HL,BC
    088A EB          2616          EX DE,HL
    088B 010020      2617          LD BC,32              ;INCREMENT OFFSET BY 32
    088E 09          2618          ADD HL,BC
    088F EB          2619          EX DE,HL
    0890 C1          2620          POP BC
    0891 D9          2621          EXX
    0892 1C          2622          INC E
                     2623 ;      UNTIL [.E,EQ,.B]        ;UNTIL Y=Y_EXTENT
                     2624
    0893 7B          2625                LD          A,E
    0894 B8          2626                CP          B
    0895 20CF        2627                JR          NZ,RPT_1
                     2628
    0897 C9          2629        RET
                     2630 ;
                     2631 ;       .COMMENT )
                     2632 *************** GET.BKGRND ***************************************************
                     2633 ;DESCRIPTION:    THIS ROUTINE GETS THE NAMES FROM THE NAME TABLE WHICH CONSTITUTE
                     2634 ;                THE BACKGROUND ON WHICH AN OBJECT IS TO BE MOVED
                     2635 ;INPUT:           HL = LOCATION IN CPU RAM TO WHICH THE NAMES ARE MOVED
                     2636 ;                 D = Y_PAT_POS [TOP ROW OF PATTERN]
                     2637 ;                 E = X_PAT_POS [LEFT HAND COLUMN]
                     2638 ;                 B = Y_EXTENT OF PATTERN
                     2639 ;                 C = X_EXTENT OF PATTERN
                     2640 ********************************************************************************
                     2641 ;
                     2642                   GLB           GET_BKGRND
    0898             2643 GET_BKGRND:
    0898 CD08C0      2644          CALL CALC_OFFSET      ;OFFSET INTO NAME TABLE OF POSITION OF UPPER LEFT
    089B C5          2645          PUSH BC               ;                       HAND PATTERN
    089C 0600        2646          LD B,0                ;GET X_EXTENT INTO IY
    089E C5          2647          PUSH BC               ;   NUMBER OF NAMES PER ROW
    089F FDE1        2648          POP IY
    08A1 C1          2649          POP BC
                     2650 ;        REPEAT                ;Y_EXTENT-1 TIMES
                     2651
    08A2             2652 RPT_2
                     2653
    08A2 C5          2654            PUSH BC
    08A3 D5          2655            PUSH DE
    08A4 E5          2656            PUSH HL
    08A5 FDE5        2657            PUSH IY
    08A7 3E02        2658            LD   A,2            ;TABLE CODE FOR PATTERN NAME TABLE
    08A9 CD18A3      2659            CALL GET_VRAM_
    08AC FDE1        2660            POP IY
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    08AE E1          2661           POP HL
    08AF D1          2662           POP DE
    08B0 C1          2663           POP BC
    08B1 C5          2664           PUSH BC
    08B2 0600        2665           LD B,0              ;BC := X_EXTENT
    08B4 09          2666           ADD HL,BC           ;POINT HL TO BEGINNING OF NEXT ROW
    08B5 010020      2667           LD BC,32
    08B8 EB          2668           EX DE,HL
    08B9 09          2669           ADD HL,BC           ;INCREMENT OFFSET BY 32
    08BA EB          2670           EX DE,HL
    08BB C1          2671           POP BC
    08BC 05          2672           DEC B
                     2673 ;       UNTIL [.B,EQ,0]
                     2674
    08BD 20E3        2675                 JR            NZ,RPT_2
                     2676
    08BF C9          2677           RET
                     2678 ;
                     2679 ;         .COMMENT }
                     2680 ************** CALC.OFFSET ************************************************
                     2681 ;DESCRIPTION:    THIS ROUTINE CALCULATES THE PROPER OFFSET INTO THE NAME TABLE
                     2682 ;                FOR THE PATTERN POSITION GIVEN BY X_PAT_POS, Y_PAT_POS.  THE
                     2683 ;                FORMULA USED IS: OFFSET = 32*Y_PAT_POS + X_PAT_POS
                     2684 ;INPUT:           D = Y_PAT_POS
                     2685 ;                 E = X_PAT_POS
                     2686 ;OUTPUT:          DE = OFFSET
                     2687 ****************************************************************************
                     2688 ;
    08C0             2689 CALC_OFFSET:
    08C0 E5          2690           PUSH HL
                     2691 ;       IF [.D,IS,MINUS]      ;EXTEND SIGN
                     2692
    08C1 CB7A        2693                 BIT           7,D
    08C3 2804        2694                 JR            Z,ELSE_11
                     2695
    08C5 26FF        2696             LD H,0FFH
                     2697
    08C7 1802        2698                 JR            END_IF_11
                     2699
                     2700 ;       ELSE
                     2701
    08C9             2702 ELSE_11
                     2703
    08C9 2600        2704             LD H,0
                     2705 ;       ENDIF
                     2706
    08CB             2707 END_IF_11
                     2708
    08CB 6A          2709           LD L,D              ;  OFFSET = 32*Y_PAT_POS + X_PAT_POS
    08CC 29          2710           ADD HL,HL           ;HL=2*Y_PAT_POS
    08CD 29          2711           ADD HL,HL           ;   4*    "
    08CE 29          2712           ADD HL,HL           ;   8*    "
    08CF 29          2713           ADD HL,HL           ;  16*    "
    08D0 29          2714           ADD HL,HL           ;  32*    "
                     2715 ;       IF [.E,IS,MINUS]      ;EXTEND SIGN
                     2716
    08D1 CB7B        2717                 BIT           7,E
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    08D3 2804        2718                JR              Z,ELSE_12
                     2719
    08D5 16FF        2720           LD D,0FFH
                     2721
    08D7 1802        2722                JR              END_IF_12
                     2723
                     2724 ;       ELSE
                     2725
    08D9             2726 ELSE_12
                     2727
    08D9 1600        2728           LD D,0
                     2729 ;       ENDIF
                     2730
    08DB             2731 END_IF_12
                     2732
    08DB 19          2733         ADD HL,DE              ;HL=32*Y_PAT_POS + X_PAT_POS
    08DC EB          2734         EX DE,HL
    08DD E1          2735         POP HL
    08DE C9          2736         RET
                     2737
                     2738 ************************* EXTERNALS *********************************
                     2739
                     2740 ;EXT  GET_VRAM
                     2741 ;EXT  PUT_VRAM
                     2742 ;EXT  VRAM_READ
                     2743 ;EXT  VRAM_WRITE
                     2744 ;EXT  WORK_BUFFER
                     2745  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2747
                     2748 * THIS MODULE CONTAINS CODE FOR THE PUT1SPRITE AND PUT0SPRITE
                     2749 * ROUTINES. THESE ROUTINES TURN OUT TO BE ESSENTIALLY THE SAME CODE
                     2750 * WITH TWO SLIGHTLY DIFFERENT ENTRY POINTS
                     2751
                     2752
                     2753 * IT IS CALLED WITH THE ADDRESS OF THE SPRITE OBJECT IN THE IX REGISTER.
                     2754
                     2755 * THE FORMAT FOR SPRITE OBJECTS IS
                     2756
                     2757 * SPRITE_OBJECT = RECORD
                     2758 *   GRAPHICS:^SPRITE_GRAPHICS
                     2759 *   STATUS:^SPRITE_STATUS
                     2760 *   SPRITE_INDEX:BYTE                  (SPRITE_NAME_TABLE INDEX OF THIS SPRITE)
                     2761 * END SPRITE_OBJECT
                     2762
                     2763 * SPRITE_GRAPHICS = RECORD
                     2764 *   OBJECT_TYPE:BYTE
                     2765 *   FIRST_GEN_NAME:BYTE                (NAME OF FIRST SPRITE GENERATOR)
                     2766 *   PTRN_POINTER:^PATTERN_GENERATOR     (POINTER TO ROM'ED GENERATORS)
                     2767 *   NUMGEN:BYTE                        (NUMBER OF ROM'ED GENERATORS)
                     2768 *   FRAME_TABLE_PTR:^ARRAY[0..nn] OF FRAME (TABLE OF ANIMATION FRAMES)
                     2769 * END SPRITE_ROM_GRAPHICS
                     2770
                     2771 * SPRITE_STATUS  = RECORD
                     2772 *   FRAME:BYTE                 (CURRENT ANIMATION FRAME)
                     2773 *   X_LOCATION:INTEGER
                     2774 *   Y_LOCATION:INTEGER
                     2775 *   NEXT_GEN:BYTE              (INDEX OF FREE SPACE IN GENERATOR TABLE)
                     2776 * END SPRITE_STATUS
                     2777
                     2778 * FRAME = RECORD
                     2779 *   COLOR:BYTE                 (SPRITE'S COLOR FOR THIS FRAME)
                     2780 *   SHAPE:BYTE                 (THIS FRAME'S OFFSET FROM NAME FROM FIRST_GEN_NAME)
                     2781 * END FRAME
                     2782
                     2783 * SPRITE = RECORD
                     2784 *   Y:BYTE
                     2785 *   X:BYTE
                     2786 *   NAME:BYTE
                     2787 *   COLOR_AND_TAG:BYTE
                     2788 * END SPRITE
                     2789
                     2790 ************************************************************************
                     2791
                     2792 ************************** DICTIONARY *****************************
                     2793
                     2794 ;              EXT             WORK_BUFFER
                     2795 * WORK_BUFFER IS A POINTER IN CARTRIDGE ROM, LOCATED AT 8006H, TO THE
                     2796 * FREE BUFFER AREA TO BE USED BY THE GRAPHICS ROUTINES.
                     2797
<000F>               2798 SPRITE_PTR      EQU             IY
                     2799 * SPRITE_PTR IS A POINTER TO THE NEW SPRITE NAME TABLE ENTRY BEING
                     2800 * BUILT BY THIS ROUTINE.
                     2801
<000E>               2802 THIS_SPRITE     EQU             IX
                     2803 * THIS_SPRITE IS A POINTER TO THE SPRITE OBJECT BEING PUT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2804
            <0000>   2805 GRAPHICS        EQU             0
            <0002>   2806 STATUS          EQU             2
            <0004>   2807 SPRITE_INDEX    EQU             4
                     2808 * FIELD OFFSETS FOR SPRITE_OBJECT RECORDS
                     2809
            <0000>   2810 OBJECT_TYPE     EQU             0
            <0001>   2811 FIRST_GEN_NAME  EQU             1
            <0002>   2812 PTRN_POINTER    EQU             2
            <0004>   2813 NUMGEN          EQU             4
            <0005>   2814 FRAME_TABLE_PTR EQU             5
                     2815 * FIELD OFFSETS FOR SPRITE GRAPHICS RECORDS
                     2816
            <0000>   2817 FRAME           EQU             0
            <0001>   2818 X_LOCATION      EQU             1
            <0003>   2819 Y_LOCATION      EQU             3
            <0005>   2820 NEXT_GEN        EQU             5
                     2821 * FIELD OFFSETS FOR SPRITE_STATUS RECORDS
                     2822
            <0000>   2823 COLOR           EQU             0
            <0001>   2824 SHAPE           EQU             1
                     2825 * FIELD OFFSETS FOR FRAME RECORDS
                     2826
            <0000>   2827 Y               EQU             0
            <0001>   2828 X               EQU             1
            <0002>   2829 NAME            EQU             2
            <0003>   2830 COLOR_AND_TAG   EQU             3
                     2831 * FIELD OFFSETS FOR SPRITE RECORDS
                     2832 ************************ EXTERNAL PROCEDURES ************************
                     2833
                     2834 ; EXT PUT_VRAM,GET_VRAM
                     2835 * EXTERNAL PROCEDURE PUT_VRAMP (TABLE_CODE:BYTE; START_INDEX,SLICE:BYTE;
                     2836 *                               VAR DATA:BUFFER;ITEM_COUNT:INTEGER);
                     2837
                     2838 * EXTERNAL PROCEDURE GET_VRAMP (TABLE_CODE:BYTE; START_INDEX,SLICE:BYTE;
                     2839 *                               VAR DATA:BUFFER;ITEM_COUNT:INTEGER);
                     2840
                     2841 * PUT_VRAM SENDS A BLOCK OF DATA TO THE TABLE SPECIFIED BY TABLE_CODE.
                     2842 * THE SLICE, START_INDEX, AND ITEM_COUNT ARE TABLE DEPENDANT. GET_VRAM
                     2843 * DOES THE INVERSE OPERATION.
                     2844
                     2845 *    - TABLE_CODE IS PASSED IN A
                     2846 *    - START_INDEX,SLICE IN DE
                     2847 *    - DATA BUFFER ADDRESS IN HL
                     2848 *    - BYTE COUNT PASSED IN IY
                     2849
                     2850 ************************ PROCEDURE BODY *****************************
                     2851
                     2852                 PROG
                     2853   GLB PUTOSPRITE,PUT1SPRITE
                     2854
                     2855 * BEGIN PUTOSPRITE
            <08DF>   2856 PUTOSPRITE      EQU             $
                     2857
                     2858 *   SPRITE_PTR := WORK_BUFFER
   08DF FD2A8006     2859                 LD              SPRITE_PTR,[WORK_BUFFER]
                     2860
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     2861 *   WITH THIS_SPRITE^,SPRITE_PTR^ DO
                     2862
                     2863 *     IF (STATUS^.X_LOCATION > -8) AND (STATUS^.X_LOCATION < 256) AND
                     2864 *        (STATUS^.Y_LOCATION > -8) AND (STATUS^.Y_LOCATION < 192) THEN
08E3 DD6E02          2865                LD              L,[THIS_SPRITE+STATUS]
08E6 DD6603          2866                LD              H,[THIS_SPRITE+STATUS+1]
08E9 110001          2867                LD              DE,X_LOCATION
08EC 19              2868                ADD             HL,DE           ; [HL] = X_LOCATION
08ED 4E              2869                LD              C,[HL]
08EE 23              2870                INC             HL
08EF 46              2871                LD              B,[HL]          ; BC = X_LOCATION
08F0 78              2872                LD              A,B             ; COMPARE BC WITH -8
08F1 FE00            2873                CP              0
08F3 2808            2874                JR              Z,OK__1
08F5 FEFF            2875                CP              -1
08F7 C20A54          2876                JP              NZ,DONT_PUT
08FA 79              2877                LD              A,C
08FB FEF9            2878                CP              -7
08FD FA0A54          2879                JP              M,DONT_PUT
0900                 2880 OK__1
0900 23              2881                INC             HL              ; [HL] = Y_LOCATION
0901 4E              2882                LD              C,[HL]
0902 23              2883                INC             HL
0903 46              2884                LD              B,[HL]          ; BC = Y_LOCATION
0904 78              2885                LD              A,B             ; COMPARE BC WITH -8
0905 FE00            2886                CP              0
0907 2808            2887                JR              Z,OK__2
0909 FEFF            2888                CP              -1
090B C20A54          2889                JP              NZ,DONT_PUT
090E 79              2890                LD              A,C
090F FEF9            2891                CP              -7
0911 FA0A54          2892                JP              M,DONT_PUT
0914                 2893 OK__2
                     2894
                     2895 *       IF STATUS^.X_LOCATION < 0 THEN
0914 2B              2896                DEC             HL
0915 2B              2897                DEC             HL              ; [HL] = HI(X_LOCATION)
0916 7E              2898                LD              A,[HL]          ; COMPARE WITH 0
0917 FE00            2899                CP              0
0919 CA09CA          2900                JP              Z,CONTINUE
                     2901
                     2902 *         X := BYTE(STATUS^.X_LOCATION) + 8
091C 2B              2903                DEC             HL              ;[HL] = ^X_LOCATION
091D 4E              2904                LD              C,[HL]
091E 23              2905                INC             HL
091F 46              2906                LD              B,[HL]
0920 210008          2907                LD              HL,8
0923 09              2908                ADD             HL,BC
0924 7D              2909                LD              A,L
0925 FD7701          2910                LD              [SPRITE_PTR+X],A
                     2911
                     2912 *         COLOR_AND_TAG := GRAPHICS^.FRAME_TABLE[STATUS^.FRAME].COLOR OR 80H
0928 DD6E00          2913                LD              L,[THIS_SPRITE+GRAPHICS]
092B DD6601          2914                LD              H,[THIS_SPRITE+GRAPHICS+1]
092E 110005          2915                LD              DE,FRAME_TABLE_PTR
0931 19              2916                ADD             HL,DE           ;[HL] = FRAME_TABLE_PTR
0932 EB              2917                EX              DE,HL
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0933 1A          2918                 LD              A,[DE]
    0934 6F          2919                 LD              L,A
    0935 13          2920                 INC             DE
    0936 1A          2921                 LD              A,[DE]
    0937 67          2922                 LD              H,A             ;[HL] = FRAME_TABLE_PTR^
    0938 E5          2923                 PUSH            HL
    0939 DD6E02      2924                 LD              L,[THIS_SPRITE+STATUS]
    093C DD6603      2925                 LD              H,[THIS_SPRITE+STATUS+1]
    093F 110000      2926                 LD              DE,FRAME
    0942 19          2927                 ADD             HL,DE           ;[HL] = FRAME
    0943 7E          2928                 LD              A,[HL]          ;CALCULATE OFFSET OF
    0944 CB27        2929                 SLA             A               ;COLOR ENTRY
    0946 010000      2930                 LD              BC,0
    0949 4F          2931                 LD              C,A
    094A E1          2932                 POP             HL
    094B 09          2933                 ADD             HL,BC           ;[HL] = COLOR
    094C 7E          2934                 LD              A,[HL]          ;OR IN 80H
    094D F680        2935                 OR              80H
    094F FD7703      2936                 LD              [SPRITE_PTR+COLOR_AND_TAG],A
                     2937
    0952 C30A00      2938                 JP              PUT_Y_AND_NAME
                     2939 *       ELSE
                     2940 ********* CONTINUE BELOW
                     2941
                     2942
                     2943 * BEGIN PUT1SPRITE
          <0955>     2944 PUT1SPRITE      EQU             $
                     2945
                     2946 *   SPRITE_PTR := WORK_BUFFER
    0955 FD2A8006    2947                 LD              SPRITE_PTR,[WORK_BUFFER]
                     2948
                     2949 *   WITH THIS_SPRITE^,SPRITE_PTR^ DO
                     2950
                     2951 *     IF (STATUS^.X_LOCATION > -32) AND (STATUS^.X_LOCATION < 256) AND
                     2952 *        (STATUS^.Y_LOCATION > -32) AND (STATUS^.Y_LOCATION < 192) THEN
    0959 DD6E02      2953                 LD              L,[THIS_SPRITE+STATUS]
    095C DD6603      2954                 LD              H,[THIS_SPRITE+STATUS+1]
    095F 110001      2955                 LD              DE,X_LOCATION
    0962 19          2956                 ADD             HL,DE           ; [HL] = X_LOCATION
    0963 4E          2957                 LD              C,[HL]
    0964 23          2958                 INC             HL
    0965 46          2959                 LD              B,[HL]          ; BC = X_LOCATION
    0966 78          2960                 LD              A,B             ; COMPARE BC WITH -32
    0967 FE00        2961                 CP              0
    0969 2808        2962                 JR              Z,OK__3
    096B FEFF        2963                 CP              -1
    096D C20A54      2964                 JP              NZ,DONT_PUT
    0970 79          2965                 LD              A,C
    0971 FEE1        2966                 CP              -31
    0973 FA0A54      2967                 JP              M,DONT_PUT
    0976             2968 OK__3
    0976 23          2969                 INC             HL              ; [HL] = Y_LOCATION
    0977 4E          2970                 LD              C,[HL]
    0978 23          2971                 INC             HL
    0979 46          2972                 LD              B,[HL]          ; BC = Y_LOCATION
    097A 78          2973                 LD              A,B             ; COMPARE BC WITH -32
    097B FE00        2974                 CP              0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    097D 280B        2975                 JR              Z,OK__4
    097F FEFF        2976                 CP              -1
    0981 C20A54      2977                 JP              NZ,DONT_PUT
    0984 79          2978                 LD              A,C
    0985 FE1E        2979                 CP              -31
    0987 FA0A54      2980                 JP              M,DONT_PUT
    098A             2981 OK__4
                     2982
                     2983 *       IF STATUS^.X_LOCATION < 0 THEN
    098A 2B          2984                 DEC             HL
    098B 2B          2985                 DEC             HL              ; [HL] = HI(X_LOCATION)
    098C 7E          2986                 LD              A,[HL]          ; COMPARE WITH 0
    098D FE00        2987                 CP              0
    098F CA09CA      2988                 JP              Z,CONTINUE
                     2989
                     2990 *         X := BYTE(STATUS^.X_LOCATION) + 32
    0992 2B          2991                 DEC             HL              ;[HL] = ^X_LOCATION
    0993 4E          2992                 LD              C,[HL]
    0994 23          2993                 INC             HL
    0995 46          2994                 LD              B,[HL]
    0996 210020      2995                 LD              HL,32
    0999 09          2996                 ADD             HL,BC
    099A 7D          2997                 LD              A,L
    099B FD7701      2998                 LD              [SPRITE_PTR+X],A
                     2999
                     3000 *         COLOR_AND_TAG := GRAPHICS^.FRAME_TABLE[STATUS^.FRAME].COLOR OR 80H
    099E DD6E00      3001                 LD              L,[THIS_SPRITE+GRAPHICS]
    09A1 DD6601      3002                 LD              H,[THIS_SPRITE+GRAPHICS+1]
    09A4 110005      3003                 LD              DE,FRAME_TABLE_PTR
    09A7 19          3004                 ADD             HL,DE           ;[HL] = FRAME_TABLE_PTR
    09A8 EB          3005                 EX              DE,HL
    09A9 1A          3006                 LD              A,[DE]
    09AA 6F          3007                 LD              L,A
    09AB 13          3008                 INC             DE
    09AC 1A          3009                 LD              A,[DE]
    09AD 67          3010                 LD              H,A             ;[HL] = FRAME_TABLE_PTR^
    09AE E5          3011                 PUSH            HL
    09AF DD6E02      3012                 LD              L,[THIS_SPRITE+STATUS]
    09B2 DD6603      3013                 LD              H,[THIS_SPRITE+STATUS+1]
    09B5 110000      3014                 LD              DE,FRAME
    09B8 19          3015                 ADD             HL,DE           ;[HL] = FRAME
    09B9 7E          3016                 LD              A,[HL]          ;CALCULATE OFFSET OF
    09BA CB27        3017                 SLA             A               ;COLOR ENTRY
    09BC 010000      3018                 LD              BC,0
    09BF 4F          3019                 LD              C,A
    09C0 E1          3020                 POP             HL
    09C1 09          3021                 ADD             HL,BC           ;[HL] = COLOR
    09C2 7E          3022                 LD              A,[HL]          ;OR IN 80H
    09C3 F680        3023                 OR              80H
    09C5 FD7703      3024                 LD              [SPRITE_PTR+COLOR_AND_TAG],A
                     3025
    09C8 1836        3026                 JR              PUT_Y_AND_NAME
                     3027 *       ELSE
                     3028 ********** CONTINUE FROM HERE
    09CA             3029 CONTINUE
                     3030
                     3031
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3032
                     3033 *         X := BYTE(STATUS^.X_LOCATION)
    09CA DD6E02      3034                LD              L,[THIS_SPRITE+STATUS]
    09CD DD6603      3035                LD              H,[THIS_SPRITE+STATUS+1]
    09D0 110001      3036                LD              DE,X_LOCATION
    09D3 19          3037                ADD             HL,DE           ; [HL] = X_LOCATION
    09D4 7E          3038                LD              A,[HL]
    09D5 FD7701      3039                LD              [SPRITE_PTR+X],A
                     3040
                     3041 *         COLOR_AND_TAG := GRAPHICS^.FRAME_TABLE[STATUS^.FRAME].COLOR
    09D8 DD6E00      3042                LD              L,[THIS_SPRITE+GRAPHICS]
    09DB DD6601      3043                LD              H,[THIS_SPRITE+GRAPHICS+1]
    09DE 110005      3044                LD              DE,FRAME_TABLE_PTR
    09E1 19          3045                ADD             HL,DE           ;[HL] = FRAME_TABLE_PTR
    09E2 EB          3046                EX              DE,HL
    09E3 1A          3047                LD              A,[DE]
    09E4 6F          3048                LD              L,A
    09E5 13          3049                INC             DE
    09E6 1A          3050                LD              A,[DE]
    09E7 67          3051                LD              H,A             ;[HL] = FRAME_TABLE_PTR^
    09E8 E5          3052                PUSH            HL
    09E9 DD6E02      3053                LD              L,[THIS_SPRITE+STATUS]
    09EC DD6603      3054                LD              H,[THIS_SPRITE+STATUS+1]
    09EF 110000      3055                LD              DE,FRAME
    09F2 19          3056                ADD             HL,DE           ;[HL] = FRAME
    09F3 7E          3057                LD              A,[HL]          ;CALCULATE OFFSET OF
    09F4 CB27        3058                SLA             A               ;COLOR ENTRY
    09F6 010000      3059                LD              BC,0
    09F9 4F          3060                LD              C,A
    09FA E1          3061                POP             HL
    09FB 09          3062                ADD             HL,BC           ;[HL] = COLOR
    09FC 7E          3063                LD              A,[HL]
    09FD FD7703      3064                LD              [SPRITE_PTR+COLOR_AND_TAG],A
                     3065
                     3066 *       END_IF
    0A00             3067 PUT_Y_AND_NAME
                     3068
                     3069 *       Y := BYTE(STATUS^.Y_LOCATION)
    0A00 DD6E02      3070                LD              L,[THIS_SPRITE+STATUS]
    0A03 DD6603      3071                LD              H,[THIS_SPRITE+STATUS+1]
    0A06 110003      3072                LD              DE,Y_LOCATION
    0A09 19          3073                ADD             HL,DE           ; [HL] = Y_LOCATION
    0A0A 7E          3074                LD              A,[HL]
    0A0B FD7700      3075                LD              [SPRITE_PTR+Y],A
                     3076
                     3077 *       NAME := GRAPHICS^.FRAME_TABLE[STATUS^.FRAME].SHAPE
                     3078 *               + GRAPHICS^.FIRST_GEN_NAME
    0A0E DD6E00      3079                LD              L,[THIS_SPRITE+GRAPHICS]
    0A11 DD6601      3080                LD              H,[THIS_SPRITE+GRAPHICS+1]
    0A14 110005      3081                LD              DE,FRAME_TABLE_PTR
    0A17 19          3082                ADD             HL,DE           ;[HL] = FRAME_TABLE_PTR
    0A18 EB          3083                EX              DE,HL
    0A19 1A          3084                LD              A,[DE]
    0A1A 6F          3085                LD              L,A
    0A1B 13          3086                INC             DE
    0A1C 1A          3087                LD              A,[DE]
    0A1D 67          3088                LD              H,A             ;[HL] = FRAME_TABLE_PTR^
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0A1E E5          3089                PUSH            HL
    0A1F DD6E02      3090                LD              L,[THIS_SPRITE+STATUS]
    0A22 DD6603      3091                LD              H,[THIS_SPRITE+STATUS+1]
    0A25 110000      3092                LD              DE,FRAME
    0A28 19          3093                ADD             HL,DE           ;[HL] = FRAME
    0A29 7E          3094                LD              A,[HL]          ;CALCULATE OFFSET OF
    0A2A CB27        3095                SLA             A               ;SHAPE ENTRY
    0A2C 010000      3096                LD              BC,0
    0A2F 4F          3097                LD              C,A
    0A30 E1          3098                POP             HL
    0A31 09          3099                ADD             HL,BC
    0A32 23          3100                INC             HL              ;[HL] = SHAPE
    0A33 7E          3101                LD              A,[HL]
    0A34 DD6E00      3102                LD              L,[THIS_SPRITE+GRAPHICS]
    0A37 DD6601      3103                LD              H,[THIS_SPRITE+GRAPHICS+1]
    0A3A 110001      3104                LD              DE,FIRST_GEN_NAME
    0A3D 19          3105                ADD             HL,DE           ;[HL] = FIRST_GEN_NAME
    0A3E 86          3106                ADD             A,[HL]
    0A3F FD7702      3107                LD              [SPRITE_PTR+NAME],A
                     3108
                     3109 *      PUT_VRAM  (0,THIS_SPRITE^.SPRITE_INDEX,SPRITE_PTR,1)
    0A42 AF          3110                XOR             A
    0A43 1600        3111                LD              D,0
    0A45 DD5E04      3112                LD              E,[THIS_SPRITE+SPRITE_INDEX]
    0A48 FDE5        3113                PUSH            SPRITE_PTR
    0A4A E1          3114                POP             HL
    0A4B FD210001    3115                LD              IY,1            ;COUNT OF ONE ITEM
    0A4F CD1FBE      3116                CALL            PUT_VRAM
                     3117
    0A52 1832        3118                JR              EXIT_PUT_SPR
                     3119 *    ELSE
    0A54             3120 DONT_PUT  ; PUT SPRITE OFF THE SCREEN BY SETTING ITS X AND EARLY CLOCK
                     3121
                     3122 *      GET_VRAM  (0,THIS_SPRITE^.SPRITE_INDEX,SPRITE_PTR,1)
    0A54 FDE5        3123                PUSH            SPRITE_PTR      ; SAVE INDEX REGS.
    0A56 DDE5        3124                PUSH            THIS_SPRITE
    0A58 FDE5        3125                PUSH            SPRITE_PTR
    0A5A FDE5        3126                PUSH            SPRITE_PTR
    0A5C AF          3127                XOR             A
    0A5D 1600        3128                LD              D,0
    0A5F DD5E04      3129                LD              E,[THIS_SPRITE+SPRITE_INDEX]
    0A62 E1          3130                POP             HL
    0A63 FD210001    3131                LD              IY,1            ;COUNT OF ONE ITEM
    0A67 CD1FBB      3132                CALL            GET_VRAM
                     3133
                     3134 *      SPRITE_PTR.X := 0
    0A6A 3E00        3135                LD              A,0
    0A6C FDE1        3136                POP             SPRITE_PTR
    0A6E FD7701      3137                LD              [SPRITE_PTR+X],A
                     3138
                     3139 *      SPRITE_PTR.COLOR_AND_TAG := 80H
    0A71 3E80        3140                LD              A,80H
    0A73 FD7703      3141                LD              [SPRITE_PTR+COLOR_AND_TAG],A
                     3142
                     3143 *      PUT_VRAM  (0,THIS_SPRITE^.SPRITE_INDEX,SPRITE_PTR,1)
    0A76 AF          3144                XOR             A
    0A77 1600        3145                LD              D,0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0A79 DDE1       3146                POP             THIS_SPRITE
    0A7B DD5E04     3147                LD              E,[THIS_SPRITE+SPRITE_INDEX]
    0A7E E1         3148                POP             HL
    0A7F FD210001   3149                LD              IY,1            ;COUNT OF ONE ITEM
    0A83 CD1F8E     3150                CALL            PUT_VRAM
                    3151
                    3152 *     END_IF
                    3153
                    3154 * END PUTOSPRITE,PUT1SPRITE
    0A86            3155 EXIT_PUT_SPR
    0A86 C9         3156                RET
                    3157  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    3159 ;
                    3160 ;********** MODIFIED VERSION TO RUN ON HP ASSEMBLER **************************
                    3161 ;
                    3162 ;                                                      4/16/82
                    3163 ;                                                      13:50:00
                    3164 ;****************** PUT_MOBILE ***************************************************
                    3165 ;
                    3166 ;DESCRIPTION:   THIS PROCEDURE PLACES A MOBILE OBJECT ON THE PATTERN PLANE
                    3167 ;               AT THE X,Y PIXEL LOCATION SPECIFIED IN THAT OBJECT'S RAM STATUS
                    3168 ;               AREA_
                    3169 ;
                    3170 ;               A BUFFER AREA OF 204 BYTES (GRAPHICS MODE II) OR 141 BYTES
                    3171 ;               (GRAPHICS MODE I) IS REQUIRED FOR FORMING THE NEW GENERATORS
                    3172 ;               REPRESENTING THE OBJECT ON IT'S BACKGROUND_  THE PROCEDURE
                    3173 ;               USES RAM STARTING AT (F_BUF_SPACE) FOR THIS BUFFER
                    3174 ;
                    3175 ;INPUT:         IX = ADDRESS OF OBJECT TO BE PROCESSED
                    3176 ;               HL = ADDRESS OF OBJECT'S GRAPHICS TABLES IN ROM
                    3177 ;                B = SELECTOR FOR METHOD OF COMBINING OBJECT GENERATORS
                    3178 ;                    WITH BACKGROUND GENERATORS
                    3179 ;
                    3180 ;                        1 = OBJECT PATTERN GENS ORed WITH BACKGROUND PATTERN GENS
                    3181 ;                            COLOR1 OF BACKGROUND CHANGED TO MOBILE OBJECT'S COLOR
                    3182 ;                            IF CORRESPONDING PATTERN BYTE NOT ZERO
                    3183 ;
                    3184 ;                        2 = REPLACE BACKGROUND PATTERN GENS WITH OBJECT PATTERN GENS
                    3185 ;                            TREAT COLOR SAME AS #1
                    3186 ;
                    3187 ;                        3 = SAME AS #1 EXCEPT COLOR0 CHANGED TO TRANSPARENT
                    3188 ;
                    3189 ;                        4 = SAME AS #2 EXCEPT COLOR0 CHANGED TO TRANSPARENT
                    3190 ;
                    3191 ;*******************************************************************************
                    3192 ;
                    3193 ;)
                    3194 ;       EXT     READ_VRAM,WRITE_VRAM,WORK_BUFFER,GET_VRAM,PUT_VRAM
                    3195 ;       EXT     PX_TO_PTRN_POS,GET_BKGRND,VDP_MODE_WORD,PUTFRAME
                    3196        GLB     PUT_MOBILE
                    3197 ; THE FOLLOWING ARE OFFSETS FROM THE START OF THE FREE BUFFER AREA
                    3198 ; THESE LOCATIONS USED TO STORE VARIABLES AND PATTERN AND COLOR DATA
          <0000>    3199 YDISP   EQU     0       ;Y DISPLACEMENT
          <0001>    3200 XDISP   EQU     1       ;X DISPLACEMENT
          <0002>    3201 COLR    EQU     2       ;COLOR
          <0003>    3202 FLAGS   EQU     3       ;BITS 0,1 = SELECTOR #, BIT X = GRAPHICS MODE [I/II]
          <0004>    3203 FRM     EQU     4       ;FRM   TO BE DISPLAYED
          <0005>    3204 F_GEN   EQU     5       ;NAME OF FIRST GENERATOR IN OBJECT'S GEN TABLE
          <0007>    3205 YP_OS   EQU     7       ;Y_PAT_POS OF OLD_SCREEN
          <0006>    3206 XP_OS   EQU     6       ;X_PAT_POS OF OLD_SCREEN
          <0012>    3207 YP_BK   EQU     18      ;Y_PAT_POS OF BACKGROUND
          <0011>    3208 XP_BK   EQU     17      ;X_PAT_POS OF BACKGROUND
          <001C>    3209 BK_PTN  EQU     28      ;START OF BACKGROUND PATTERN GENERATORS
          <0064>    3210 OBJ_PTN EQU     100     ;START OF OBJECT'S PATTERN GENERATORS
          <0084>    3211 BK_CLR  EQU     132     ;START OF BACKGROUND COLOR GENERATORS
    0A87            3212 PUT_MOBILE
                    3213 ; GET X AND Y LOCATIONS, CONVERT TO X AND Y PATTERN POSITIONS AND X AND Y
                    3214 ; DISPLACEMENTS [AMOUNT BY WHICH OBJECT SHIFTED OFF PATTERN POSITION BOUNDARY]
    0A87 FD2A8006   3215        LD IY,[WORK_BUFFER]     ;GET START OF FREE BUFFER AREA
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0A88 3A73C3      3216         LD A,[VDP_MODE_WORD]      ;FIND OUT WHICH GRAPHICS MODE WE ARE IN
    0A8B CB4F        3217         BIT 1,A
                     3218 ;       IF [PSW,IS,ZERO]          ;THEN MODE I
    0A90 2004        3219         JR NZ,ELSE1
    0A92 CBB8        3220           RES 7,B
    0A94 1802        3221         JR END1
                     3222 ;       ELSE                      ;MODE II
    0A96 CBF8        3223 ELSE1     SET 7,B
                     3224 ;       ENDIF
    0A98 FD7003      3225 END1    LD [IY+FLAGS],B           ;SAVE SELECTOR
    0A9B E5          3226         PUSH HL                   ;SAVE GRAPHICS ADDRESS
    0A9C DD6603      3227         LD H,[IX+3]               ;HL := ADDR_ OF STATUS
    0A9F DD6E02      3228         LD L,[IX+2]
    0AA2 7E          3229         LD A,[HL]                 ;GET FRAME #
    0AA3 FD7704      3230         LD [IY+FRM],A            ;AND SAVE
    0AA6 EE80        3231         XOR 80H                   ;COMPLEMENT TABLE_IN_USE FLAG
    0AA8 77          3232         LD [HL],A                 ;SAVE BACK IN STATUS AREA
    0AA9 23          3233         INC HL                    ;POINT TO X_LOCATION
    0AAA 5E          3234         LD E,[HL]                 ;E := LOW X_LOCATION
    0AAB 7B          3235         LD A,E
    0AAC E607        3236         AND 7                     ;A := #PIXELS TO RIGHT OF PATTERN BOUNDARY
    0AAE ED44        3237         NEG
    0AB0 C608        3238         ADD A,8                   ;AMOUNT TO SHIFT PATTERN LEFT FROM NEXT PAT BOUNDARY
    0AB2 FD7701      3239         LD [IY+XDISP],A           ;SAVE
    0AB5 23          3240         INC HL
    0AB6 56          3241         LD D,[HL]                 ;DE := X_LOCATION
    0AB7 CD07E8      3242         CALL PX_TO_PTRN_POS       ;CALCULATE X_PAT_POS OF BACKGROUND
    0ABA FD7311      3243         LD [IY+XP_BK],E           ;AND SAVE
    0ABD 23          3244         INC HL                    ;POINT TO Y_LOCATION
    0ABE 5E          3245         LD E,[HL]                 ;E := LOW Y_LOCATION
    0ABF 7B          3246         LD A,E
    0AC0 E607        3247         AND 7                     ;A := #PIXELS TO RIGHT OF PATTERN BOUNDARY
    0AC2 FD7700      3248         LD [IY+YDISP],A           ;SAVE
    0AC5 23          3249         INC HL
    0AC6 56          3250         LD D,[HL]                 ;DE := Y_LOCATION
    0AC7 CD07E8      3251         CALL PX_TO_PTRN_POS       ;CALCULATE Y_PAT_POS
    0ACA FD7312      3252         LD [IY+YP_BK],E
                     3253
                     3254 ; NOW GET THE NINE NAMES THAT CONSTITUTE THE BACKGROUND ON WHICH THE MOBILE OBJECT
                     3255 ; WILL BE SUPERIMPOSED
    0ACD 2A8006      3256 PM1     LD HL,[WORK_BUFFER]
    0AD0 110013      3257         LD DE,YP_BK+1             ;POINT TO SPACE FOR BACKGROUND NAMES
    0AD3 19          3258         ADD HL,DE
    0AD4 FD5612      3259         LD D,[IY+YP_BK]           ;D := Y_PAT_POS
    0AD7 FD5E11      3260         LD E,[IY+XP_BK]           ;E := X_PAT_POS
    0ADA 010303      3261         LD BC,303H                ;B := Y_EXTENT, C := X_EXTENT
    0ADD CD0898      3262         CALL GET_BKGRND           ;GET BACKGROUND NAMES
                     3263 ; READ OLD_SCREEN INTO BUFFER AND GET COLOR AND FIRST_GEN_NAME
    0AE0 DD5605      3264 PM2     LD D,[IX+5]               ;DE := OLD_SCREEN ADDRESS
    0AE3 DD5E04      3265         LD E,[IX+4]
    0AE6 DD7E06      3266         LD A,[IX+6]               ;GET FIRST GEN NAME
    0AE9 DDE1        3267         POP IX                    ;IX := ADDRESS OF GRAPHICS
    0AEB FD2A8006    3268         LD IY,[WORK_BUFFER]
    0AEF FD7705      3269         LD [IY+F_GEN],A           ;SAVE IN BUFFER
    0AF2 D5          3270         PUSH DE                   ;SAVE OLD_SCREEN ADDRESS
    0AF3 2A8006      3271         LD HL,[WORK_BUFFER]       ;HL := ADDR OF START OF BUFFER
    0AF6 010006      3272         LD BC,XP_OS               ;SPACE TO MOVE OLD_SCREEN TO
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0AF9 09          3273         ADD HL,BC
    0AFA 010008      3274         LD BC,11                ;GET 9 NAMES FROM VRAM
                     3275 ;       IF [_D,LT,70H]          ;THEN OLD_SCREEN IS IN VRAM
    0AFD 7A          3276         LD A,D
    0AFE FE70        3277         CP 70H
    0B00 3005        3278         JR NC,ELSE2
    0B02 CD1FE2      3279           CALL READ_VRAM
    0B05 1803        3280           JR END2
                     3281 ;       ELSE                    ;OLD_SCREEN IN CPU RAM
    0B07 EB          3282 ELSE2     EX DE,HL
    0B08 EDB0        3283           LDIR
    0B0A             3284 END2  ;ENDIF
                     3285
                     3286 ; AT THIS POINT, IX = GRAPHICS, [SP] = OLD_SCREEN
                     3287 ; BACKGROUND PATTERN POSITION AND NAMES STARTING AT YP_BK
                     3288 ; OLD_SCREEN PATTERN POSITION AND NAMES STARTING AT YP_OS
                     3289 ; FIND ALL NAMES IN BACKGROUND WHICH BELONG TO THIS OBJECT'S PATTERN GENERATORS
                     3290 ; AND REPLACE WITH NAME FROM OLD_SCREEN WHICH CORRESPONDS TO THAT PATTERN POSITION
    0B0A 2A8006      3291 PM3     LD HL,[WORK_BUFFER]     ;HL := BUFFER BASE
    0B0D 110013      3292         LD DE,YP_BK+1
    0B10 19          3293         ADD HL,DE               ;POINT TO FIRST OF BACKGROUND NAMES
    0B11 D9          3294         EXX
    0B12 ED5B8006    3295         LD DE,[WORK_BUFFER]     ;DE' := BUFFER BASE
    0B16 210008      3296         LD HL,YP_OS+1
    0B19 19          3297         ADD HL,DE
    0B1A EB          3298         EX DE,HL                ;POINTS TO FIRST OF OLD_SCREEN NAMES
    0B1B D9          3299         EXX
    0B1C FD2A8006    3300         LD IY,[WORK_BUFFER]
    0B20 FD4E05      3301         LD C,[IY+F_GEN]         ;C := FIRST_GEN_NAME
                     3302 ;       DO B,9
    0B23 0609        3303         LD B,9
    0B25 7E          3304 DLP1      LD A,[HL]               ;GET A NAME
    0B26 91          3305           SUB C                 ;SUBTRACT FIRST_GEN_NAME
                     3306 ;         IF [_A,LT,18]         ;THEN NAME FALLS IN RANGE OF NAMES FOR OBJECT
    0B27 FE12        3307           CP 18
    0B29 300E        3308           JR NC,END3
                     3309 ;           IF [_A,GE,9]          ;THEN SUB 9 TO FIND CORRECT
    0B2B FE09        3310             CP 9
    0B2D 3802        3311             JR C,END4
    0B2F D609        3312               SUB 9               ;   POSITION IN OLD_SCREEN
    0B31             3313 END4        ;ENDIF
    0B31 D9          3314             EXX
    0B32 6F          3315             LD L,A              ;FORM A POINTER INTO OLD_SCREEN NAMES
    0B33 2600        3316             LD H,0
    0B35 19          3317             ADD HL,DE
    0B36 7E          3318             LD A,[HL]           ;GET OLD_SCREEN NAME
    0B37 D9          3319             EXX
    0B38 77          3320             LD [HL],A           ;REPLACE BACKGROUND NAME WITH OLD_SCREEN NAME
    0B39             3321 END3      ;ENDIF
    0B39 23          3322         INC HL                  ;POINT TO NEXT NAME IN BACKGROUND
                     3323 ;       ENDDO
    0B3A 10E9        3324         DJNZ DLP1
                     3325
                     3326 ; NOW NEW VERSION OF BACKGROUND NAMES WILL NOT CONTAIN ANY NAMES OF THIS OBJECT
                     3327 ; REPLACE PREVIOUS VERSION OF OLD_SCREEN WITH THIS NEW BACKGROUND
    0B3C D1          3328 PM4     POP DE                  ;DE := OLD_SCREEN ADDRESS
    0B3D 2A8006      3329         LD HL,[WORK_BUFFER]     ;HL := BUFFER BASE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0840 010011      3330         LD BC,XP_BK
    0843 09          3331         ADD HL,BC              ;HL POINTS TO BACKGROUND POSITION AND NAMES
    0844 010008      3332         LD BC,11               ;NUMBER OF BYTES TO MOVE
                     3333 ;       IF [_D,LT,70H]         ;THEN MOVE DATA TO VRAM
    0847 7A          3334         LD A,D
    0848 FE70        3335         CP 70H
    084A 3005        3336         JR NC,ELSE5
    084C CD1FDF      3337           CALL WRITE_VRAM
    084F 1802        3338         JR END5
    0851             3339 ELSE5   ;ELSE                  ;MOVE TO CPU RAM
    0851 EDB0        3340           LDIR
    0853             3341 END5    ;ENDIF
                     3342 ; GET THE PATTERN AND COLOR GENERATORS SPECIFIED BY THE NINE BACKGROUND NAMES
                     3343 ; IX = GRAPHICS
                     3344
    0853 DDE5        3345                PUSH            IX              ;SAVE GRAPHICS POINTER
                     3346
    0855 ED5B8006    3347 PM5     LD DE,[WORK_BUFFER]    ;DE := BUFFER BASE
    0859 210013      3348         LD HL,YP_BK+1          ;DISPLACEMENT TO BEGINNING OF BKGND NAMES
    085C 19          3349         ADD HL,DE
    085D EB          3350         EX DE,HL               ;DE POINT TO FIRST BACKGROUND NAME
    085E 010014      3351         LD BC,BK_PTN-8         ;HL POINTS TO BACKGROUND PAT GEN STORAGE AREA -8
    0861 09          3352         ADD HL,BC              ;  WILL BE INCREMENTED BEFORE FIRST USE
                     3353 ;       DO B,9
    0862 0609        3354         LD B,9
    0864 1A          3355 DLP2      LD A,[DE]            ;GET NAME
    0865 13          3356           INC DE               ;POINT TO NEXT NAME
    0866 D5          3357           PUSH DE              ;SAVE NAME POINTER
    0867 110008      3358           LD DE,8              ;INCREMENT BUFFER POINTER
    086A 19          3359           ADD HL,DE
    086B E5          3360           PUSH HL              ;SAVE BUFFER POINTER
    086C 5F          3361           LD E,A               ;INDEX INTO PATTERN TABLE
    086D 1600        3362           LD D,0
    086F 4F          3363           LD C,A               ;SAVE PATTERN NAME IN C
    0870 C5          3364           PUSH BC              ;SAVE COUNTER
    0871 3E09        3365           LD A,9               ;ADD (B-9)/3 TO YP_BK TO FIND
    0873 90          3366           SUB B                ;    OUT WHICH THIRD OF TABLES GENs ARE IN
    0874 0600        3367           LD B,0
    0876 D603        3368 PM52      SUB 3
    0878 3803        3369           JR C,PM51
    087A 04          3370           INC B
    087B 18F9        3371           JR PM52
    087D 78          3372 PM51      LD A,B
    087E FD2A8006    3373           LD IY,[WORK_BUFFER]
    0882 FD8612      3374           ADD A,[IY+YP_BK]
    0885 FDCB037E    3375           BIT 7,[IY+FLAGS]     ;TEST GRAPHICS MODE
    0889 FD210001    3376           LD IY,1              ;NUMBER OF ELEMENTS TO READ
    088D             3377 PM6       ;IF [PSW,IS,ZERO]    ;THEN MODE I
    088D 2029        3378           JR NZ,ELSE6
    088F 3E03        3379             LD A,3             ;CODE FOR PATTERN GENERATOR TABLE
    0891 CD1FBB      3380             CALL GET_VRAM
    0894 C1          3381             POP BC             ;GET COUNT AND NAME
    0895 2A8006      3382             LD HL,[WORK_BUFFER]
    0898 C5          3383             PUSH BC
    0899 110084      3384             LD DE,BK_CLR       ;DISPLACEMENT TO COLOR GEN AREA
    089C 19          3385             ADD HL,DE          ;POINT TO IT
    089D 59          3386             LD E,C             ;PATTERN NAME
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0B9E CB38        3387            SRL E                ;DIVIDE NAME BY 8
    0BA0 CB38        3388            SRL E
    0BA2 CB38        3389            SRL E
    0BA4 1600        3390            LD D,0
    0BA6 3E09        3391            LD A,9               ;CALC POSITION IN BUFFER TO MOVE GEN TO
    0BA8 90          3392            SUB B
    0BA9 4F          3393            LD C,A
    0BAA 0600        3394            LD B,0
    0BAC 09          3395            ADD HL,BC
    0BAD FD210001    3396            LD IY,1
    0BB1 3E04        3397            LD A,4               ;COLOR GENERATOR TABLE CODE
    0BB3 CD1FBB      3398            CALL GET_VRAM
    0BB6 1821        3399            JR END6
    0BB8             3400 ELSE6     ;ELSE                 ;MUST BE MODE II
    0BB8 CB2F        3401            SRA A                ;DIVIDE Y_POS BY 8
    0BBA CB2F        3402            SRA A
    0BBC CB2F        3403            SRA A                ;A := THIRD OF TABLE, 0=1ST, 1=2ND, 2=3RD
    0BBE             3404 PM7        ;IF [_A,LT,3]        ;IF A > 2, THEN Y_POS > 23 AND THEREFORE OFF SCREEN
    0BBE FE03        3405            CP 3
    0BC0 3017        3406            JR NC,END7
    0BC2 57          3407              LD D,A             ;DE := 256*A + NAME
    0BC3 D5          3408              PUSH DE
    0BC4 E5          3409              PUSH HL
    0BC5 3E03        3410              LD A,3             ;PATTERN GENERATOR TABLE CODE
    0BC7 CD1FBB      3411              CALL GET_VRAM
    0BCA E1          3412              POP HL             ;HL := PATTERN BUFFER ADDRESS
    0BCB 110068      3413              LD DE,BK_CLR-BK_PTN ;DISPLACEMENT BETWEEN PATTERN AND COLOR BUFFERS
    0BCE 19          3414              ADD HL,DE          ;HL := POINTER TO COLOR BUFFER
    0BCF D1          3415              POP DE
    0BD0 FD210001    3416              LD IY,1
    0BD4 3E04        3417              LD A,4             ;CODE FOR COLOR TABLE
    0BD6 CD1FBB      3418              CALL GET_VRAM
    0BD9             3419 END7       ;ENDIF
    0BD9             3420 END6      ;ENDIF
    0BD9 C1          3421           POP BC               ;RESTORE REGISTERS
    0BDA E1          3422           POP HL
    0BDB D1          3423           POP DE
                     3424 ;        ENDDO
    0BDC 1086        3425           DJNZ DLP2
                     3426
                     3427 ; NOW THE PATTERN AND COLOR GENERATORS ARE IN THEIR RESPECTIVE BUFFERS
                     3428 ; SO GET THE FOUR GENERATORS FOR THIS FRAME OF THE OBJECT
                     3429
    0BDE DDE1        3430                POP             IX              ;RESTORE GRAPHICS POINTER
                     3431
    0BE0 D9          3432 PM8       EXX
    0BE1 DD5603      3433           LD D,[IX+3]
    0BE4 DD5E02      3434           LD E,[IX+2]          ;DE' := NEW_GEN
    0BE7 DD4605      3435           LD B,[IX+5]
    0BEA DD4E04      3436           LD C,[IX+4]          ;BC' := PTRN_POINTER
    0BED D9          3437           EXX
    0BEE DDE5        3438           PUSH IX
    0BF0 E1          3439           POP HL               ;HL := ADDRESS OF GRAPHICS
    0BF1 FD2A8006    3440           LD IY,[WORK_BUFFER]
    0BF5 FD7E04      3441           LD A,[IY+FRM]       ;A := FRM #
    0BF8 87          3442           ADD A,A              ;X2
    0BF9 4F          3443           LD C,A
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0BFA 0600        3444         LD B,0
    0BFC 110006      3445         LD DE,6              ;FRAME POINTERS START AT THIS OFFEST
    0BFF 19          3446         ADD HL,DE
    0C00 09          3447         ADD HL,BC            ;HL := POINTS TO FRAME_POINTER FOR FRAME
    0C01 5E          3448         LD E,[HL]
    0C02 23          3449         INC HL
    0C03 56          3450         LD D,[HL]            ;DE := ADDRESS OF FRAME NAMES
    0C04 2A8006      3451         LD HL,[WORK_BUFFER]  ;HL := BUFFER BASE ADDRESS
    0C07 01007C      3452         LD BC,OBJ_PTN+24     ;USE LOCATION FOR LAST GEN TO STORE NAMES
    0C0A 09          3453         ADD HL,BC
    0C0B E5          3454         PUSH HL              ;SAVE FOR LATER USE
    0C0C C5          3455         PUSH BC              ;SAVE OFFSET
    0C0D 010005      3456         LD BC,5
                     3457 ;       IF [D,LT,70H]        ;THEN NAMES ARE IN VRAM
    0C10 7A          3458         LD A,D
    0C11 FE70        3459         CP 70H
    0C13 3005        3460         JR NC,ELSE8
    0C15 CD1FE2      3461           CALL READ_VRAM        ;GET THE 4 NAMES
    0C18 1803        3462           JR END8
    0C1A             3463 ELSE8   ;ELSE                ;NAMES IN CPU MEMORY SPACE
    0C1A EB          3464           EX DE,HL
    0C1B EDB0        3465           LDIR
    0C1D             3466 END8    ;ENDIF
    0C1D FD2A8006    3467         LD IY,[WORK_BUFFER]  ;GET COLOR BYTE
    0C21 C1          3468         POP BC               ;OFFSET TO FIRST NAME
    0C22 FD09        3469         ADD IY,BC
    0C24 FD7E04      3470         LD A,[IY+4]          ;A := COLOR BYTE
    0C27 FD2A8006    3471         LD IY,[WORK_BUFFER]
    0C2B FD7702      3472         LD [IY+COLR],A       ;COLR := COLOR BYTE
    0C2E D1          3473 PM9     POP DE               ;DE := ADDRESS OF FIRST NAME IN BUFFER
    0C2F 2A8006      3474         LD HL,[WORK_BUFFER]  ;HL := BUFFER BASE ADDRESS
    0C32 010064      3475         LD BC,OBJ_PTN        ;START OF OBJECT'S PATTERN BUFFER
    0C35 09          3476         ADD HL,BC
                     3477 ;       DO B,4               ;GET 4 PATTERNS CORRESPONDING TO THE FOUR NAMES
    0C36 0604        3478         LD B,4
    0C38 1A          3479 DLP4    LD A,[DE]              ;GET NAME
    0C39 DDBE01      3480           CP [IX+1]            ;COMPARE TO NUMGEN
    0C3C D5          3481           PUSH DE              ;SAVE POINTER TO NAMES
                     3482 ;         IF [PSW,IS,CARRY]    ;THEN NAME < NUMGEN, THEREFORE
    0C3D 3016        3483           JR NC,ELSE9
    0C3F D9          3484             EXX                ;     PATTERN PART OF GRAPHICS TABLES
    0C40 87          3485             ADD A,A
    0C41 87          3486             ADD A,A
    0C42 87          3487             ADD A,A            ;A := 8*NAME
    0C43 6F          3488             LD L,A
    0C44 2600        3489             LD H,0
    0C46 09          3490             ADD HL,BC          ;HL := POINTER TO PATTERN
    0C47 E5          3491             PUSH HL
    0C48 D9          3492             EXX
    0C49 D1          3493             POP DE             ;DE :=   "          "
    0C4A EB          3494             EX DE,HL
    0C4B C5          3495             PUSH BC
    0C4C 010008      3496             LD BC,8            ;NUMBER OF BYTES TO MOVE
    0C4F EDB0        3497             LDIR
    0C51 C1          3498             POP BC
    0C52 EB          3499             EX DE,HL
    0C53 1830        3500             JR END9
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0C55             3501 ELSE9     ;ELSE                  ;NAME => NUMGEN, THEREFORE NOT PART OF ROMED GENS
    0C55 DD9601      3502             SUB [IX+1]             ;SUBTRACT NUMGEN FROM NAME
    0C58 D9          3503             EXX
    0C59 87          3504             ADD A,A
    0C5A 87          3505             ADD A,A
    0C5B 87          3506             ADD A,A                ;A := 8*NAME
    0C5C 6F          3507             LD L,A
    0C5D 2600        3508             LD H,0
    0C5F 19          3509             ADD HL,DE              ;HL := POINTER TO PATTERN
    0C60 E5          3510             PUSH HL
    0C61 D9          3511             EXX
    0C62 D1          3512             POP DE
    0C63             3513 PM10      ;IF [_D,LT,70H]          ;THEN PATTERN IN VRAM
    0C63 7A          3514             LD A,D
    0C64 FE70        3515             CP 70H
    0C66 3014        3516             JR NC,ELSE10
    0C68 C5          3517               PUSH BC
    0C69 E5          3518               PUSH HL              ;SAVE BUFFER ADDRESS
    0C6A D5          3519               PUSH DE              ;SAVE GENERATOR ADDRESS
    0C6B 010008      3520               LD BC,8              ;NUMBER BYTES TO MOVE
    0C6E CD1FE2      3521               CALL READ_VRAM
    0C71 010008      3522               LD BC,8              ;INCREMENT POINTERS BY 8
    0C74 E1          3523               POP HL
    0C75 09          3524               ADD HL,BC            ;GEN ADDR := GEN ADDR + 8
    0C76 EB          3525               EX DE,HL
    0C77 E1          3526               POP HL
    0C78 09          3527               ADD HL,BC            ;BUF ADDR := BUF ADDR + 8
    0C79 C1          3528               POP BC
    0C7A 1809        3529               JR END10
    0C7C             3530 ELSE10    ;ELSE                  ;PATTERN IN CPU RAM
    0C7C EB          3531               EX DE,HL
    0C7D C5          3532               PUSH BC
    0C7E 010008      3533               LD BC,8
    0C81 EDB0        3534               LDIR
    0C83 C1          3535               POP BC
    0C84 EB          3536               EX DE,HL
    0C85             3537 END10      ;ENDIF
    0C85             3538 END9     ;ENDIF
    0C85 D1          3539            POP DE
    0C86 13          3540            INC DE
                     3541 ;        ENDDO
    0C87 10AF        3542          DJNZ DLP4
                     3543
                     3544 ; COMBINE OBJECT PATTERN GENERATORS WITH BACKGROUND
    0C89 FD2A8006    3545 PM11     LD IY,[WORK_BUFFER]
    0C8D ED5B8006    3546          LD DE,[WORK_BUFFER]     ;DE = BUFFER BASE ADDR_
    0C91 21001C      3547          LD HL,BK_PTN
    0C94 19          3548          ADD HL,DE               ;HL POINTS TO START OF BACKGROUND GENERATORS
    0C95 FD4E00      3549          LD C,[IY+YDISP]         ;BC=
    0C98 0600        3550          LD B,0                  ;  DISPLACEMENT OF OBJECT BELOW PATTERN BOUNDARY
    0C9A 09          3551          ADD HL,BC               ;HL POINTS TO FIRST ROW IN BACKGROUND TO BE OVERLAYED
    0C9B E5          3552          PUSH HL
    0C9C DDE1        3553          POP IX                  ;IX := POINTER TO ROW IN PATTERN BUFFER
    0C9E 210064      3554          LD HL,OBJ_PTN
    0CA1 19          3555          ADD HL,DE               ;HL POINTS TO FIRST BYTE IN OBJECT'S PATTERN GENs
    0CA2 E5          3556          PUSH HL
    0CA3 3E10        3557          LD A,16                 ;USE A' AS LOOP COUNTER
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0CA5 08          3558         EX AF,AF'
    0CA6             3559 COMBINE_LOOP
    0CA6 E1          3560         POP HL                ;POINT TO OBJECT PATTERN BYTE
    0CA7 56          3561         LD D,[HL]             ;GET PATTERN BYTE
    0CA8 23          3562         INC HL
    0CA9 E5          3563         PUSH HL
    0CAA 01000F      3564         LD BC,15
    0CAD 09          3565         ADD HL,BC             ;POINT TO ADJACENT PATTERN BYTE
    0CAE 5E          3566         LD E,[HL]
    0CAF EB          3567         EX DE,HL              ;HL HAS 16 BIT ROW OF OBJECT'S PATTERN
    0CB0 FD4601      3568         LD B,[IY+XDISP]       ;SHIFT LEFT PATTERN BY THIS AMOUNT
    0CB3 AF          3569         XOR A                 ;CLEAR A
    0CB4 05          3570 SHFLP   DEC B
    0CB5 FA0CBC      3571         JP M,SHFEX
    0CB8 29          3572         ADD HL,HL
    0CB9 17          3573         RLA
    0CBA 18F8        3574         JR SHFLP
    0CBC 5F          3575 SHFEX   LD E,A                ;SAVE LEFT BYTE IN E
    0CBD CD0E2F      3576         CALL COM_PAT_COL
    0CC0 FD7E00      3577         LD A,[IY+YDISP]       ;INCREMENT YDISP
    0CC3 3C          3578         INC A
    0CC4 FD7700      3579         LD [IY+YDISP],A
                     3580 ;       IF [_A,EQ,8],OR,[_A,EQ,16]
    0CC7 FE08        3581         CP 8
    0CC9 2804        3582         JR Z,IF11
    0CCB FE10        3583         CP 16
    0CCD 2005        3584         JR NZ,END11
    0CCF 010010      3585 IF11      LD BC,16
    0CD2 DD09        3586           ADD IX,BC           ;BEGINNING OF NEXT ROW
    0CD4             3587 END11   ;ENDIF
    0CD4 DD23        3588         INC IX                ;POINT TO NEXT GEN BYTE
    0CD6 08          3589         EX AF,AF'             ;DECREMENT LOOP COUNTER
    0CD7 3D          3590         DEC A
    0CD8 2803        3591         JR Z,C_LP_EXIT
    0CDA 08          3592         EX AF,AF'
    0CDB 18C9        3593         JR COMBINE_LOOP
    0CDD             3594 C_LP_EXIT
    0CDD E1          3595         POP HL                ;POP POINTER OFF STACK
    0CDE FDCB037E    3596         BIT 7,[IY+FLAGS]      ;TEST FOR MODE 1
                     3597 ;       IF [PSW,IS,ZERO]      ;THEN UPDATE MODE 1 COLOR GENs
    0CE2 201F        3598         JR NZ,END12
    0CE4 2A8006      3599           LD HL,[WORK_BUFFER]
    0CE7 010084      3600           LD BC,BK_CLR        ;OFFSET FOR COLOR BUFFER
    0CEA 09          3601           ADD HL,BC           ;HL POINTS TO START OF COLOR BUFFER
    0CEB FD5602      3602           LD D,[IY+COLR]      ;GET OBJECT COLOR
    0CEE FDCB034E    3603           BIT 1,[IY+FLAGS]    ;COLOR0 = BACKGROUND OR TRANSPARENT ?
                     3604 ;         IF [PSW,IS,ZERO]    ;THEN USE BACKGROUND COLOR0
    0CF2 2004        3605           JR NZ,ELSE13
    0CF4 0E0F        3606             LD C,0FH          ;MASK FOR COLOR0 OF BACKGROUND
    0CF6 1802        3607             JR END13
    0CF8             3608 ELSE13    ;ELSE
    0CF8 0E00        3609             LD C,0            ;MASK REPLACE COLOR0 WITH TRANSPARENT
    0CFA             3610 END13     ;ENDIF
                     3611 ;         DO B,9              ;CHANGE ALL 9 COLOR BYTES
    0CFA 0609        3612           LD B,9
    0CFC 7E          3613 DLP5        LD A,[HL]         ;GET BACKGROUND COLOR GEN
    0CFD A1          3614             AND C             ;MASK OUT COLOR1
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0CFE B2          3615            OR D                    ;ADD OBJECT COLOR1
    0CFF 77          3616            LD (HL),A               ;UPDATE COLOR GENERATOR
    0D00 23          3617            INC HL                  ;POINT TO NEXT COLOR GEN
                     3618 ;        ENDDO
    0D01 10F9        3619          DJNZ DLP5
    0D03             3620 END12   ;ENDIF
                     3621
                     3622 ; DECIDE WHICH PART OF OBJECT'S PATTERN AND COLOR TABLES TO USE
    0D03 FD7E05      3623          LD A,(IY+F_GEN)           ;GET NAME OF FIRST GENERATOR IN OBJECT'S TABLE
    0D06 FDCB047E    3624          BIT 7,(IY+FRM)          ;TEST WHICH PART OF TABLE TO USE
                     3625 ;        IF (PSW,IS,NZERO)         ;THEN USE UPPER HALF OF TABLES
    0D0A 2802        3626          JR Z,END14
    0D0C C609        3627            ADD A,9
    0D0E             3628 END14   ;ENDIF
                     3629 ; CHANGE NAMES IN BACKGROUND BUFFER TO THOSE OF THE OBJECT'S PATTERNS
                     3630 ; THIS WILL THEN CONSTITUTE A NEW FRAME DISPLAYING THE OBJECT AT PATTERN PLANE
                     3631 ; LOCATION (YP_BK),(XP_BK)
    0D0E 4F          3632          LD C,A                    ;SAVE INDEX
    0D0F 2A8006      3633          LD HL,(WORK_BUFFER)       ;HL := BUFFER BASE
    0D12 110013      3634          LD DE,YP_BK+1             ;POINT TO FIRST BACKGROUND NAME
    0D15 19          3635          ADD HL,DE
                     3636 ;        DO B,9
    0D16 0609        3637          LD B,9
    0D18 77          3638 DLP6       LD (HL),A
    0D19 3C          3639            INC A
    0D1A 23          3640            INC HL
                     3641 ;        ENDDO
    0D1B 10FB        3642          DJNZ DLP6
                     3643 ; MOVE NEWLY FORMED GENERATORS TO OBJECT'S PATTERN AND COLOR GEN TABLES
    0D1D FDCB037E    3644 PM12     BIT 7,(IY+FLAGS)          ;TEST WHICH MODE
                     3645 ;        IF (PSW,IS,ZERO)          ;THEN MODE 1
    0D21 2073        3646          JR NZ,ELSE04
    0D23 59          3647            LD E,C
    0D24 1600        3648            LD D,0                    ;DE := INDEX INTO PATTERN GENERATOR TABLE
    0D26 2A8006      3649            LD HL,(WORK_BUFFER)
    0D29 01001C      3650            LD BC,BK_PTN
    0D2C 09          3651            ADD HL,BC                 ;HL POINTS TO FIRST GENERATOR
    0D2D FD210009    3652            LD IY,9                   ;9 ELEMENTS TO SEND
    0D31 3E03        3653            LD A,3                    ;CODE FOR PATTERN GENERATOR TABLE
    0D33 CD1FBE      3654            CALL PUT_VRAM
                     3655 ; SET UP POINTERS TO OBJECT'S PATTERN NAMES AND TO COLOR GEN BYTES
    0D36 FD2A8006    3656            LD IY,(WORK_BUFFER)
    0D3A 2A8006      3657            LD HL,(WORK_BUFFER)       ;HL := BUFFER BASE
    0D3D 010084      3658            LD BC,BK_CLR              ;OFFSET TO COLOR BUFFER
    0D40 09          3659            ADD HL,BC                 ;HL POINTS TO COLOR BUFFER
    0D41 DD2A8006    3660            LD IX,(WORK_BUFFER)       ;IX := BUFFER BASE
    0D45 010013      3661            LD BC,YP_BK+1             ;OFFSET TO FIRST NAME FOR OBJECT
    0D48 DD09        3662            ADD IX,BC                 ;IX POINTS TO FIRST NAME
    0D4A 0609        3663            LD B,9
    0D4C             3664 RPT1       ;REPEAT                   ;MOVE COLORS TO COLOR GENERATOR TABLE
    0D4C DD7E00      3665              LD A,(IX+0)               ;GET NAME
    0D4F DD23        3666              INC IX                    ;POINT TO NEXT NAME
    0D51 CB3F        3667              SRL A                     ;DIVIDE BY 8
    0D53 CB3F        3668              SRL A
    0D55 CB3F        3669              SRL A
    0D57 5F          3670              LD E,A
    0D58 1600        3671              LD D,0                    ;DE := OFFSET INTO COLOR GEN TABLE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    005A C5          3672              PUSH BC                 ;SAVE COUNTER
    005B 3E09        3673              LD A,9                  ;TEST WHETHER THIS PATTERN POSITION IS
    005D 90          3674              SUB B                   ;   ON THE SCREEN OR NOT (I.E. IS YP.BK
    005E 0600        3675              LD B,0                  ;   + (B-9)/3 BETWEEN 0 AND 23, AND IS
    0060 FE03        3676 DVLP         CP 3                    ;   XP.BAK + (B-9)/3 MOD 3 BETWEEN 0 AND 31)
    0062 3805        3677              JR C,DVEX
    0064 D603        3678              SUB 3
    0066 04          3679              INC B
    0067 18F7        3680              JR DVLP
    0069 FD8611      3681 DVEX         ADD A,[IY+XP_BK]        ;A := X_POS + (B-9)/3 MOD 3
    006C FE20        3682              CP 32
    006E 3017        3683              JR NC,NOTOS
    0070 78          3684              LD A,B
    0071 FD8612      3685              ADD A,[IY+YP_BK]        ;A := Y_POS + (B-9)/3
    0074 FE18        3686              CP 24                   ;IS THIS POSITION ON SCREEN ?
    0076 300F        3687              JR NC,NOTOS
    0078 DDE5        3688              PUSH IX                 ;      AND POINTERS
    007A E5          3689              PUSH HL
    007B 3E04        3690              LD A,4                  ;CODE FOR COLOR GENERATOR TABLE
    007D FD210001    3691              LD IY,1                 ;NUMBER OF ITEMS TO SEND
    0081 CD1FBE      3692              CALL PUT_VRAM
    0084 E1          3693              POP HL                  ;RESTORE POINTERS
    0085 DDE1        3694              POP IX
    0087 C1          3695 NOTOS        POP BC                  ;     AND COUNTER
    0088 23          3696              INC HL                  ;INCREMENT POINTERS
    0089 13          3697              INC DE
    008A 05          3698              DEC B                   ;DECREMENT COUNTER
    008B 78          3699            LD A,B
    008C FE00        3700            CP 0
    008E 208C        3701            JR NZ,RPT1
                     3702 ;          UNTIL [_B,EQ,0]
    0090 FD2A8006    3703          LD IY,[WORK_BUFFER]  ;RESTORE BUFFER ADDR
    0094 1854        3704          JR END04
    0096             3705 ELSE04
    0096             3706 PM13     ;ELSE
    0096 0600        3707            LD B,0
    0098             3708 RPT2       ;REPEAT
    0098 C5          3709              PUSH BC             ;SAVE COUNTER AND INDEX
    0099 79          3710              LD A,C              ;ADD 3xCOUNTER TO INDEX, WHERE NEXT GENs TO BE MOVED
    009A 80          3711              ADD A,B
    009B 80          3712              ADD A,B
    009C 80          3713              ADD A,B
    009D 4F          3714              LD C,A              ;C := INDEX TO TABLE LOCATION FOR NEXT 3 GENs
    009E 210000      3715              LD HL,0             ;CALCULATE OFFSETS FROM START OF PATTERN
    00A1 110018      3716              LD DE,24            ;  AND COLOR GENERATOR TABLES
    00A4 78          3717              LD A,B
    00A5 3D          3718 AD_LP        DEC A
    00A6 FA0DAC      3719              JP M,AD_EXIT
    00A9 19          3720              ADD HL,DE
    00AA 18F9        3721              JR AD_LP
                     3722                                  ;HL := OFFSET FROM START OF PTRN AND COLOR BUFFERS
    00AC FD2A8006    3723 AD_EXIT      LD IY,[WORK_BUFFER]
    00B0 FD7E12      3724              LD A,[IY+YP_BK]     ;GET Y_PAT_POS TO SEE WHICH THIRD OF TABLES TO USE
    00B3 80          3725              ADD A,B             ;EACH GROUP OF THREE IN NEXT PATTERN PLANE ROW
                     3726 ;            IF [_A,LT,24]       ;'A' MUST CONTAIN VALID Y_PAT_POS [0-23]
    00B4 FE18        3727              CP 24
    00B6 302B        3728              JR NC,END15
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    00B8 CB3F       3729              SRL A               ;THIS NUMBER / 8 INDICATES WHICH 1/3 OF
    00BA CB3F       3730              SRL A               ;    TABLES TO USE
    00BC CB3F       3731              SRL A
    00BE 57         3732              LD D,A
    00BF 59         3733              LD E,C              ;DE := INDEX INTO PATTERN AND COLOR TABLES
    00C0 D5         3734              PUSH DE             ;SAVE INDEX
    00C1 01001C     3735              LD BC,BK_PTN        ;FORM POINTER TO GENERATORS IN HL
    00C4 09         3736              ADD HL,BC
    00C5 ED4B8006   3737              LD BC,[WORK_BUFFER] ;GET BUFFER BASE ADDR
    00C9 09         3738              ADD HL,BC
    00CA E5         3739              PUSH HL             ;SAVE THIS POINTER
    00CB FD210003   3740              LD IY,3             ;NUMBER OF ELEMENTS TO MOVE
    00CF 3E03       3741              LD A,3              ;PATTERN GENERATOR TABLE CODE
    00D1 CD1FBE     3742              CALL PUT_VRAM
    00D4 E1         3743              POP HL              ;GET POINTER BACK
    00D5 110068     3744              LD DE,BK_CLR-BK_PTN ;OFFSET BETWEEN BUFFERS
    00D8 19         3745              ADD HL,DE           ;HL POINTS TO START OF NEXT 3 COLOR GENERATORS
    00D9 D1         3746              POP DE              ;GET INDEX INTO GEN TABLES
    00DA FD210003   3747              LD IY,3
    00DE 3E04       3748              LD A,4              ;CODE FOR COLOR GENERATOR TABLE
    00E0 CD1FBE     3749              CALL PUT_VRAM
    00E3            3750 END15       ;ENDIF
    00E3 C1         3751             POP BC              ;RESTORE COUNTER AND INDEX
    00E4 04         3752             INC B
    00E5 78         3753          LD A,B
    00E6 FE03       3754          CP 3
    00E8 20AE       3755          JR NZ,RPT2
                    3756 ;        UNTIL [_B,EQ,3]    ;REPEAT 3 TIMES
    00EA            3757 END04   ;ENDIF
                    3758
                    3759 ; RESTORE OLD_SCREEN IF IT'S Y_PAT_POS AND X_PAT_POS DIFFERS FROM THE
                    3760 ; Y_PAT_POS AND X_PAT_POS FOR THE OBJECT
    00EA FD2A8006   3761 PM14    LD IY,[WORK_BUFFER]
    00EE FD4606     3762         LD B,[IY+XP_OS]         ;TEST FOR VALID OLD_SCREEN DATA
                    3763 ;       IF [_B,NE,80H]          ;THEN THERE IS VALID DATA
    00F1 78         3764         LD A,B
    00F2 FE80       3765         CP 80H
    00F4 2821       3766         JR Z,END16
    00F6 FD4E07     3767           LD C,[IY+YP_OS]       ;TEST IF OS POSITION SAME AS CURRENT POSITION
    00F9 FD6611     3768           LD H,[IY+XP_BK]
    00FC FD6E12     3769           LD L,[IY+YP_BK]
    00FF B7         3770           OR A                  ;CLEAR THE CARRY
    0E00 ED42       3771           SBC HL,BC             ;IS THERE ANY DIFFERENCE?
                    3772 ;         IF [PSW,IS,NZERO]     ;THEN POSTION HAS CHANGED
    0E02 2813       3773           JR Z,END17
    0E04 2A8006     3774             LD HL,[WORK_BUFFER]   ;GET BUFFER BASE
    0E07 110008     3775             LD DE,YP_OS+1
    0E0A 19         3776             ADD HL,DE             ;POINT TO OLD_SCREEN NAMES
    0E0B FD5E06     3777             LD E,[IY+XP_OS]
    0E0E FD5607     3778             LD D,[IY+YP_OS]       ;DE := X AND Y PAT_POS
    0E11 010303     3779             LD BC,0303H           ;BC := X AND Y EXTENT
    0E14 CD0808     3780             CALL PUTFRAME
    0E17            3781 END17      ;ENDIF
    0E17            3782 END16    ;ENDIF
                    3783 ; PLACE OBJECT ON SCREEN
    0E17 FD2A8006   3784         LD IY,[WORK_BUFFER]
    0E1B 2A8006     3785         LD HL,[WORK_BUFFER]     ;HL := BUFFER BASE ADDR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0E1E 110013      3786         LD DE,YP_BK+1          ;POINT TO NAMES FOR OBJECT
    0E21 19          3787         ADD HL,DE
    0E22 FD5E11      3788         LD E,[IY+XP_BK]
    0E25 FD5612      3789         LD D,[IY+YP_BK]        ;DE := X AND Y PAT_POS
    0E28 010303      3790         LD BC,0303H            ;BC := X AND Y EXTENT
    0E2B CD0808      3791         CALL PUTFRAME
                     3792 ****************** END OF PUT_MOBILE ***********************************************
    0E2E C9          3793         RET
                     3794
                     3795 ; REGS A, H AND L CONTAIN 24 BIT PATTERN TO BE COMBINED WITH BACKGROUND GENERATORS
                     3796 ; IX POINTS TO THE FIRST OF THREE GENERATOR BYTES TO BE COMBINED WITH A, H AND L
    0E2F             3797 COM_PAT_COL
    0E2F FDCB0346    3798         BIT 0,[IY+FLAGS]       ;'OR' GENs OR REPLACE
                     3799 ;       IF [PSW,IS,ZERO]       ;THEN 'OR'
    0E33 2016        3800         JR NZ,ELSE18
    0E35 DDB600      3801           OR [IX+0]              ;'OR' LEFT BYTE WITH BACKGROUND
    0E38 DD7700      3802           LD [IX+0],A            ;AND SUBSTITUTE FOR THAT GENERATOR BYTE
    0E3B 7C          3803           LD A,H                 ;NOW DO MIDDLE BYTE
    0E3C DDB608      3804           OR [IX+8]
    0E3F DD7708      3805           LD [IX+8],A
    0E42 7D          3806           LD A,L                 ;RIGHT HAND BYTE
    0E43 DDB610      3807           OR [IX+16]
    0E46 DD7710      3808           LD [IX+16],A
    0E49 1814        3809           JR END18
    0E4B             3810 ELSE18  ;ELSE                  ;REPLACE BACKGROUND WITH NON-ZERO BYTES
    0E4B B7          3811           OR A                   ;IS BYTE NON-ZERO
                     3812 ;         IF [PSW,IS,NZERO]
    0E4C 2803        3813           JR Z,END19
    0E4E DD7700      3814             LD [IX+0],A          ;YES, THEN REPLACE BACKGROUND WITH OBJECT
    0E51             3815 END19     ;ENDIF
    0E51 7C          3816           LD A,H                 ;SAME FOR MIDDLE BYTE
    0E52 B7          3817           OR A
                     3818 ;         IF [PSW,IS,NZERO]
    0E53 2803        3819           JR Z,END20
    0E55 DD7708      3820             LD [IX+8],A
    0E58             3821 END20     ;ENDIF
    0E58 7D          3822           LD A,L                 ;SAME FOR RIGHT HAND BYTE
    0E59 B7          3823           OR A
                     3824 ;         IF [PSW,IS,NZERO]
    0E5A 2803        3825           JR Z,END21
    0E5C DD7710      3826             LD [IX+16],A
    0E5F             3827 END21     ;ENDIF
    0E5F             3828 END18   ;ENDIF
    0E5F FDCB037E    3829 PM15    BIT 7,[IY+FLAGS]       ;FIND OUT WHICH GRAPHICS MODE USED
                     3830 ;       IF [PSW,IS,NZERO]      ;THEN MODE 2 [MODE 1 COLORS DONE AFTER COMBINELOOP]
    0E63 283C        3831         JR Z,END22
    0E65 DDE5        3832           PUSH IX              ;SAVE BACKGROUND POINTER
    0E67 010068      3833           LD BC,BK_CLR-BK_PTN  ;CHANGE IX TO POINT TO COLOR GENERATORS
    0E6A DD09        3834           ADD IX,BC
    0E6C FD4602      3835           LD B,[IY+COLR]       ;GET OBJECT COLOR
    0E6F FDCB034E    3836           BIT 1,[IY+FLAGS]     ;COLOR0 = BACKGROUND OR TRANSPARENT ?
                     3837 ;         IF [PSW,IS,ZERO]     ;THEN USE BACKGROUND COLOR0
    0E73 2004        3838           JR NZ,ELSE23
    0E75 0E0F        3839             LD C,0FH             ;MASK FOR COLOR0 OF BACKGROUND
    0E77 1802        3840             JR END23
    0E79             3841 ELSE23    ;ELSE
    0E79 0E00        3842             LD C,0               ;MASK REPLACE COLOR0 WITH TRANSPARENT
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0E7B             3843 END23     ;ENDIF
    0E7B 7B          3844           LD A,E                  ;GET FIRST OBJECT'S PATTERN BYTE
    0E7C B7          3845           OR A                    ;ARE THERE ANY '1' BITS ?
                     3846 ;         IF [PSW,IS,NZERO]
    0E7D 2808        3847           JR Z,END24
    0E7F DD7E00      3848             LD A,[IX+0]             ;GET BACKGROUND COLOR GEN
    0E82 A1          3849             AND C                   ;MASK OUT COLOR1
    0E83 B0          3850             OR B                    ;ADD OBJECT COLOR1
    0E84 DD7700      3851             LD [IX+0],A             ;UPDATE COLOR GENERATOR
    0E87             3852 END24     ;ENDIF
    0E87 7C          3853           LD A,H                  ;SAME FOR MIDDLE BYTE
    0E88 B7          3854           OR A
                     3855 ;         IF [PSW,IS,NZERO]
    0E89 2808        3856           JR Z,END25
    0E8B DD7E08      3857             LD A,[IX+8]
    0E8E A1          3858             AND C
    0E8F B0          3859             OR B
    0E90 DD7708      3860             LD [IX+8],A
    0E93             3861 END25     ;ENDIF
    0E93 7D          3862           LD A,L                  ;RIGHT HAND BYTE
    0E94 B7          3863           OR A
                     3864 ;         IF [PSW,IS,NZERO]
    0E95 2808        3865           JR Z,END26
    0E97 DD7E10      3866             LD A,[IX+16]
    0E9A A1          3867             AND C
    0E9B B0          3868             OR B
    0E9C DD7710      3869             LD [IX+16],A
    0E9F             3870 END26     ;ENDIF
    0E9F DDE1        3871           POP IX                  ;RESTORE BACKGROUND POINTER
    0EA1             3872 END22   ;ENDIF
    0EA1 C9          3873         RET
                     3874 ;       END             ;PUT_MOBILE
                     3875  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     3877 ;       .IDENT PUTCOMP
                     3878 ;       .ZOP
                     3879 ;       .EPOP
                     3880 ;       .IF1 ,.INSERT B:SPZ80.ASM
                     3881 ;       .COMMENT )
                     3882 ;                                                    4/15/82
                     3883 ;                                                    10:40:00
                     3884 ;****************** PUT_COMPLEX ***************************************************
                     3885 ;
                     3886 ;DESCRIPTION:   THE POSITION AND FRAME NUMBER OF EACH OF A COMPLEX OBJECT'S
                     3887 ;               COMPONENT OBJECTS IS UPDATED.  THEN PUT_OBJECT IS CALLED FOR
                     3888 ;               EACH OF THE COMPONENT OBJECTS.
                     3889 ;
                     3890 ;INPUT:         IX = ADDRESS OF OBJECT TO BE PROCESSED
                     3891 ;               HL = ADDRESS OF OBJECT'S GRAPHICS TABLES IN ROM
                     3892 ;                B = SELECTOR FOR METHODE OF COMBINING OBJECT GENERATORS
                     3893 ;                    WITH BACKGROUND GENERATORS
                     3894 ;
                     3895 ;                        1 = OBJECT PATTERN GENS ORed WITH BACKGROUND PATTERN GENS
                     3896 ;                            COLOR1 OF BACKGROUND CHANGED TO MOBILE OBJECT'S COLOR
                     3897 ;                            IF CORRESPONDING PATTERN BYTE NOT ZERO
                     3898 ;
                     3899 ;                        2 = REPLACE BACKGROUND PATTERN GENS WITH OBJECT PATTERN GENS
                     3900 ;                            TREAT COLOR SAME AS #1
                     3901 ;
                     3902 ;                        3 = SAME AS #1 EXCEPT COLOR0 CHANGED TO TRANSPARENT
                     3903 ;
                     3904 ;                        4 = SAME AS #2 EXCEPT COLOR0 CHANGED TO TRANSPARENT
                     3905 ;
                     3906 ;                 C = OBJECT TYPE, AND NUMBER OF COMPONENTS
                     3907 ;
                     3908 ;*******************************************************************************
                     3909 ;)
                     3910 ;       EXT     PUTOBJ
                     3911         GLB     PUTCOMPLEX
                     3912 ;
0EA2                 3913 PUTCOMPLEX
                     3914 ; UPDATE THE FRAME NUMBER AND THE X AND Y LOCATION IN EACH OF THE COMPONTEN
                     3915 ; OBJECT'S STATUS AREAS
0EA2 C5              3916         PUSH BC                 ;SAVE SELECTOR AND COMPONENT COUNT [COMP_CNT]
0EA3 D9              3917         EXX                     ;USE PRIMED REGS FOR X_LOC AND Y_LOC
0EA4 DD6603          3918         LD H,[IX+3]             ;HIGH BYTE OF STATUS
0EA7 DD6E02          3919         LD L,[IX+2]             ;LOW BYTE OF STATUS
0EAA 7E              3920         LD A,[HL]               ;A := FRAME
0EAB 23              3921         INC HL
0EAC 4E              3922         LD C,[HL]               ;BC' := X_LOCATION
0EAD 23              3923         INC HL
0EAE 46              3924         LD B,[HL]
0EAF 23              3925         INC HL
0EB0 5E              3926         LD E,[HL]               ;DE' := Y_LOCATION
0EB1 23              3927         INC HL
0EB2 56              3928         LD D,[HL]
0EB3 D9              3929         EXX
0EB4 87              3930         ADD A,A                 ;FRAME := 4*FRAME
0EB5 87              3931         ADD A,A
0EB6 5F              3932         LD E,A                  ;FORM POINTER TO FRAME AND OFFSET POINTERS
0EB7 1600            3933         LD D,0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0EB9 23          3934         INC HL                  ;POINT TO FIRST OF FRA_OFFSET_PNTR PAIRS
    0EBA 19          3935         ADD HL,DE               ;POINT TO FRAME POINTER
    0EBB 4E          3936         LD C,[HL]
    0EBC 23          3937         INC HL
    0EBD 46          3938         LD B,[HL]               ;BC := FRAME POINTER [POINTER TO LIST OF FRAME #'S]
    0EBE 23          3939         INC HL
    0EBF 5E          3940         LD E,[HL]
    0EC0 23          3941         INC HL
    0EC1 56          3942         LD D,[HL]               ;DE := OFFSET POINTER [PNTR TO LIST OF OFFSETS]
    0EC2 60          3943         LD H,B
    0EC3 69          3944         LD L,C                  ;HL := FRAME POINTER
                     3945 ;
                     3946 ; DE' = Y_LOC, BC' = X_LOC, HL = PNTR TO FRAME LIST, DE = PNTR TO OFFSET LIST
                     3947 ; IX = ADDR OF OBJ, [SP] = COMP_CNT & SELECTOR
                     3948 ; FOR N=0 TO COMP_CNT-1: COMP_OBJ[N] FRAME := FRAME#[N] FROM FRAME LIST
                     3949 ;                        COMP_OBJ[N] X_LOCATION := CMPLX_OBJ X_LOCATION + X_OFFSET[N]
                     3950 ;                        COMP_OBJ[N] Y_LOCATION := CMPLX_OBJ Y_LOCATION + Y_OFFSET[N]
    0EC4 C1          3951         POP BC
    0EC5 79          3952         LD A,C                  ;GET COMPONENT COUNT INTO B
    0EC6 48          3953         LD C,B                  ;SAVE SELECTOR IN C
    0EC7 CB3F        3954         SRL A                   ;GET COUNT INTO LOW NIBBLE
    0EC9 CB3F        3955         SRL A
    0ECB CB3F        3956         SRL A
    0ECD CB3F        3957         SRL A
    0ECF 47          3958         LD  B,A                 ;B := component count
    0ED0 C5          3959         PUSH BC                 ;SAVE COUNTER AND SELECTOR ON STACK
    0ED1 DDE5        3960         PUSH IX                 ;SAVE ADDR OF OBJ
    0ED3 E5          3961 LP1     PUSH HL                 ;SAVE PNTR TO FRAME LIST
    0ED4 D5          3962         PUSH DE                 ;SAVE PNTR TO OFFSET LIST
    0ED5 DD6E04      3963         LD L,[IX+4]
    0ED8 DD6605      3964         LD H,[IX+5]             ;HL := ADDR OF COMPONENT OBJ
    0EDB DD23        3965         INC IX                  ;Point to next object pointer
    0EDD DD23        3966         INC IX
    0EDF 23          3967         INC HL
    0EE0 23          3968         INC HL                  ;HL POINTS TO STATUS POINTER
    0EE1 5E          3969         LD E,[HL]
    0EE2 23          3970         INC HL
    0EE3 56          3971         LD D,[HL]               ;DE := ADDR OF STATUS FOR COMPONENT OBJECT
    0EE4 D5          3972         PUSH DE
    0EE5 FDE1        3973         POP IY                  ;IY := ADDR OF STATUS FOR COMPONENT OBJECT
    0EE7 D1          3974         POP DE                  ;DE := PNTR TO OFFSET LIST
    0EE8 E1          3975         POP HL                  ;HL := PNTR TO FRAME LIST
    0EE9 7E          3976         LD A,[HL]               ;GET FRAME NUMBER
    0EEA FDCB007E    3977         BIT 7,[IY+0]            ;Preserve bit 7 of frame
    0EEE 2802        3978         JR  Z,TBL0              ;used by mobile objects to indicate which
                     3979                                 ;VRAM tables in use.
    0EF0 CBFF        3980         SET 7,A
    0EF2             3981 TBL0:
    0EF2 FD7700      3982         LD [IY+0],A             ;MOVE TO COMPONENTS STATUS AREA
    0EF5 23          3983         INC HL                  ;POINT TO NEXT FRAME NUMBER
    0EF6 1A          3984         LD A,[DE]               ;GET X_OFFSET
    0EF7 D9          3985         EXX
    0EF8 6F          3986         LD L,A
    0EF9 2600        3987         LD H,0                  ;HL' := X_OFFSET
    0EFB 09          3988         ADD HL,BC               ;HL' := X_OFFSET + X_LOCATION
    0EFC FD7501      3989         LD [IY+1],L
    0EFF FD7402      3990         LD [IY+2],H             ;COMPONENT'S X_LOCATION := X_OFFSET + X_LOCATION
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0F02 D9          3991         EXX
    0F03 13          3992         INC DE                ;POINT TO Y_OFFSET
    0F04 1A          3993         LD A,(DE)
    0F05 D9          3994         EXX
    0F06 6F          3995         LD L,A
    0F07 2600        3996         LD H,0                ;HL := Y_OFFSET
    0F09 19          3997         ADD HL,DE             ;HL := Y_OFFSET + Y_LOCATION
    0F0A FD7503      3998         LD (IY+3),L
    0F0D FD7404      3999         LD (IY+4),H           ;COMPONENT'S Y_LOCATION := Y_OFFSET + Y_LOCATION
    0F10 D9          4000         EXX
    0F11 13          4001         INC DE                ;POINT TO NEXT OFFSET PAIR
    0F12 10BF        4002         DJNZ LP1
                     4003 ;
                     4004 ; CALL PUT_OBJECT FOR EACH OF THE COMPONENT OBJECTS, PASS SELECTOR IN B
    0F14 FDE1        4005         POP IY                ;GET OBJECT ADDRESS BACK
    0F16 010004      4006         LD BC,4
    0F19 FD09        4007         ADD IY,BC             ;IY POINTS TO POINTER TO FIRST COMPONENT OBJECT
    0F1B D1          4008         POP DE                ;DE := COUNTER AND SELECTOR
    0F1C FD6E00      4009 LP2     LD L,(IY+0)
    0F1F FD6601      4010         LD H,(IY+1)           ;HL := ADDRESS OF COMPONENT OBJECT
    0F22 FD23        4011         INC IY
    0F24 FD23        4012         INC IY                ;IY POINTS TO NEXT COMPONENT OBJECT POINTER
    0F26 E5          4013         PUSH HL
    0F27 DDE1        4014         POP IX                ;IX := ADDRESS OF COMPONENT OBJECT
    0F29 FDE5        4015         PUSH IY               ;SAVE POINTER
    0F2B D5          4016         PUSH DE               ;SAVE COUNTER AND SELECTOR
    0F2C 43          4017         LD B,E                ;B := SELECTOR
    0F2D CD1FFA      4018         CALL PUTOBJ
    0F30 D1          4019         POP DE                ;GET COUNTER AND SELECTOR
    0F31 FDE1        4020         POP IY                ;GET ADDRESS OF NEXT COMPONENT OBJECT POINTER
    0F33 15          4021         DEC D
    0F34 20E6        4022         JR NZ,LP2
    0F36 C9          4023         RET
                     4024 PROG
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

                    4026
                    4027
                    4028 ;            Ken   Lagace and Rob Jepson 3/82
                    4029
                    4030                  GLB           TIMER_TABLE_BASE      ;IS THIS NECESSARY ??
                    4031                  GLB           NEXT_TIMER_DATA_BYTE  ; OR THIS???
                    4032
                    4033                  ;EXT          PARAM_     ;Parameter passing routine needed
                    4034                                           ;for setting up pascal interfaces
                    4035
                    4036                            ;Global routine labels are the routines to call
                    4037                            ;from pascal
                    4038                  GLB           INIT_TIMER_
                    4039                  GLB           INIT_TIMERQ
                    4040                  GLB           FREE_SIGNAL_
                    4041                  GLB           FREE_SIGNALQ
                    4042                  GLB           REQUEST_SIGNAL_
                    4043                  GLB           REQUEST_SIGNALQ
                    4044                  GLB           TEST_SIGNAL_
                    4045                  GLB           TEST_SIGNALQ
                    4046                  GLB           TIME_MGR_
                    4047                  GLB           TIME_MGRQ
         <0007>     4048 DONE             EQU           7
         <0006>     4049 REPEAT           EQU           6
         <0005>     4050 FREE             EQU           5
         <0004>     4051 EOT              EQU           4
         <0003>     4052 LONG             EQU           3
                    4053 ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    4054 ;                                                                         ;
                    4055 ;                 ------------ MODE BIT -------------                     ;
                    4056 ;      TO CHECK BIT =====  BIT ?  JR Z = JUMP IF BIT IS 0 !               ;
                    4057 ;                                                                         ;
                    4058 ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    4059 ;        ;        ;        ;        ;;        ;        ;        ;        ;
                    4060 ;  DONE  ; REPEAT ;  FREE  ;  E O T ;;  LONG  ;        ;        ;        ;
                    4061 ;   7    ;   6    ;   5    ;    4   ;;   3    ;   2    ;   1    ;   0    ;
                    4062 ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    4063
                    4064
                    4065             PROG
                    4066
 0F37               4067 TIME_MGRQ
 0F37               4068 TIME_MGR_
 0F37 2A73D3        4069             LD            HL,[TIMER_TABLE_BASE];Current timer addr.
 0F3A               4070 NEXT_TIMERO
 0F3A CB6E          4071             BIT           FREE,[HL]       ;Free?
 0F3C CC0F49        4072             CALL          Z,DCR_TIMER     ;If not, decr.
 0F3F CB66          4073             BIT           EOT,[HL]        ;End of table?
 0F41 2005          4074             JR            NZ,SCRAM        ; If it is, we're done.
 0F43 23            4075             INC           HL              ;Otherwise get next timer
 0F44 23            4076             INC           HL              ; and start over.
 0F45 23            4077             INC           HL
 0F46 18F2          4078             JR            NEXT_TIMERO
 0F48               4079 SCRAM
 0F48 C9            4080             RET
                    4081
                    4082
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4083
                     4084 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;:
                     4085
   0F49              4086 DCR_TIMER
   0F49 E5           4087              PUSH         HL              ;Save current timer.
   0F4A CB5E         4088              BIT          LONG,[HL]       ;Long?
   0F4C 282B         4089              JR           Z,DCR_S_MODE_TBL;Short,non-repeating
   0F4E CB76         4090              BIT          REPEAT,[HL]     ;Repeat?
   0F50 2000         4091              JR           NZ,DCR_L_RPT_TBL;Long, repeating
   0F52              4092 DCR_L_MODE_TBL                            ;Long non-repeating
   0F52 23           4093              INC          HL
   0F53 5E           4094              LD           E,[HL]          ;Move counter to DE
   0F54 23           4095              INC          HL
   0F55 56           4096              LD           D,[HL]
   0F56 1B           4097              DEC          DE              ;Decrement.
   0F57 7B           4098              LD           A,E
   0F58 B2           4099              OR           D               ;Check if 0.
   0F59 2034         4100              JR           NZ,SAVE_2_BYTES ;If not, save'm.
   0F5B E1           4101              POP          HL              ;Otherwise, get mode byte
   0F5C E5           4102              PUSH         HL              ; and set it's done bit.
   0F5D 182C         4103              JR           SET_DONE_BIT
   0F5F              4104 DCR_L_RPT_TBL
   0F5F 23           4105              INC          HL              ;Long-repeating timer.
   0F60 5E           4106              LD           E,[HL]          ;Load addr. into DE.
   0F61 23           4107              INC          HL
   0F62 56           4108              LD           D,[HL]
   0F63 EB           4109              EX           DE,HL           ;Exchange and
   0F64 5E           4110              LD           E,[HL]          ; load counter into DE.
   0F65 23           4111              INC          HL
   0F66 56           4112              LD           D,[HL]
   0F67 1B           4113              DEC          DE              ;Decrement.
   0F68 7B           4114              LD           A,E
   0F69 B2           4115              OR           D               ;Check for 0.
   0F6A 2023         4116              JR           NZ,SAVE_2_BYTES ;Save if not.
   0F6C 23           4117              INC          HL              ;Otherwise, reload
   0F6D 5E           4118              LD           E,[HL]          ; original counter #.
   0F6E 23           4119              INC          HL
   0F6F 56           4120              LD           D,[HL]          ;Jockey all over to
   0F70 2B           4121              DEC          HL              ; perform said task!
   0F71 2B           4122              DEC          HL
   0F72 72           4123              LD           [HL],D
   0F73 2B           4124              DEC          HL
   0F74 73           4125              LD           [HL],E
   0F75 E1           4126              POP          HL
   0F76 E5           4127              PUSH         HL
   0F77 1812         4128              JR           SET_DONE_BIT    ;Then set done bit.
   0F79              4129 DCR_S_MODE_TBL
   0F79 23           4130              INC          HL
   0F7A 35           4131              DEC          [HL]
   0F7B 2010         4132              JR           NZ,TIMER_EXIT
   0F7D E1           4133              POP          HL
   0F7E E5           4134              PUSH         HL
   0F7F CB76         4135              BIT          REPEAT,[HL]     ;Repeat?
   0F81 2808         4136              JR           Z,SET_DONE_BIT  ;If not, leave.
   0F83 23           4137              INC          HL              ;Otherwise, jockey
   0F84 23           4138              INC          HL              ; around again and
   0F85 7E           4139              LD           A,[HL]          ; reload original #.
```

```
CATION OBJECT CODE LINE     SOURCE LINE

  0F86 2B          4140                 DEC             HL
  0F87 77          4141                 LD              [HL],A
  0F88 2B          4142                 DEC             HL
  0F89 E1          4143                 POP             HL
  0F8A E5          4144                 PUSH            HL
  0F8B             4145 SET_DONE_BIT
  0F8B CBFE        4146                 SET             7,[HL]
  0F8D             4147 TIMER_EXIT
  0F8D E1          4148                 POP             HL
  0F8E C9          4149                 RET
                   4150
                   4151
                   4152
  0F8F             4153 SAVE_2_BYTES
  0F8F 72          4154                 LD              [HL],D
  0F90 2B          4155                 DEC             HL
  0F91 73          4156                 LD              [HL],E
  0F92 18F9        4157                 JR              TIMER_EXIT
                   4158
                   4159 ;Procedure Init Timer
                   4160 ;HL has address of Timer Table
                   4161 ;DE has address of Timer Data Table
                   4162
                   4163 ;          COMN
                   4164 ;INIT_TIME_DATA:
                   4165 ;TEMP1:
                   4166 ;          DEFS 2
                   4167 ;TEMP2:
                   4168 ;          DEFS 2
                   4169
                   4170            PROG
  0F94             4171 INIT_TIME_PAR:
  0F94 00020002    4172            DEFW 2,2,2
  0F98 0002
                   4173
  0F9A             4174 INIT_TIMERQ:
  0F9A 010F94      4175            LD   BC,INIT_TIME_PAR
  0F9D 1173BA      4176            LD   DE,INIT_TIME_DATA
  0FA0 CD0098      4177            CALL PARAM
  0FA3 2A73BA      4178            LD   HL,[TEMP1]
  0FA6 ED5B73BC    4179            LD   DE,[TEMP2]
  0FAA             4180 INIT_TIMER_:
  0FAA 2273D3      4181            LD   [TIMER_TABLE_BASE],HL          ;Store given base address for timer table
  0FAD 3630        4182            LD   [HL],30H                       ;Set first byte in timer table to free and last timer
  0FAF EB          4183            EX   DE,HL
  0FB0 2273D5      4184            LD   [NEXT_TIMER_DATA_BYTE],HL      ;Store given base address for data block
  0FB3 C9          4185            RET
                   4186
                   4187 ;Procedure Free Signal
                   4188 ;Acc has signal number to be freed
                   4189 ;No output is generated
                   4190 ;          COMN
                   4191 ;SIGNAL_NUM:
                   4192 ;          DEFS 1
                   4193
                   4194
                   4195            PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4196
    0FB4             4197 FREE_SIG_PAR:
    0FB4 00010001    4198           DEFW 1,1
                     4199
    0FB8             4200 FREE_SIGNALQ:
    0FB8 010FB4      4201           LD   BC,FREE_SIG_PAR
    0FBB 11738E      4202           LD   DE,SIGNAL_NUM
    0FBE CD0098      4203           CALL PARAM_
    0FC1 3A738E      4204           LD   A,[SIGNAL_NUM]
                     4205
    0FC4             4206 FREE_SIGNAL_:
    0FC4 4F          4207           LD   C,A                        ;Put Free Code into C register
    0FC5 2A73D3      4208           LD   HL,[TIMER_TABLE_BASE]      ;Get Timer Base Address
    0FC8 47          4209           LD   B,A                        ;Timer offset already available
    0FC9 110003      4210           LD   DE,3                       ;Setup offset to subsequent timers
    0FCC B7          4211           OR   A                          ;See if the first timer is requested
    0FCD 2808        4212           JR   Z,FREE_MATCH               ;If so we have a match
                     4213
    0FCF             4214 FREE1:                                    ;Loop to find requested signal
    0FCF CB66        4215           BIT  EOT,[HL]                   ;Check if end of table
    0FD1 206A        4216           JR   NZ,FREE_EXIT               ;If so go no further
    0FD3 19          4217           ADD  HL,DE                      ;If not then offset to next timer
    0FD4 0D          4218           DEC  C                          ;Decrement signal requested
    0FD5 20F8        4219           JR   NZ,FREE1                   ;If requested signal not zero then go try the next
                     4220
    0FD7             4221 FREE_MATCH:                               ;Here when the requested signal matches current signal
    0FD7 CB6E        4222           BIT  FREE,[HL]                  ;Check if already free
    0FD9 2062        4223           JR   NZ,FREE_SET.               ;If so then just reset LONG
    0FDB CBEE        4224           SET  FREE,[HL]                  ;Set current timer to free
    0FDD CB76        4225           BIT  REPEAT,[HL]                ;Check for repeating timer
    0FDF 285C        4226           JR   Z,FREE_SET                 ;If not then go free it
    0FE1 CB5E        4227           BIT  LONG,[HL]                  ;Check also for long timer
    0FE3 2858        4228           JR   Z,FREE_SET                 ;If not long then go free it
                     4229 ;         CALL FREE_COUNTER_               ;Release timer data before freeing timer
                     4230
                     4231 ;
                     4232 ;              FREE   (DELETE) COUNTER
                     4233 ;                   KERNL 3/82
                     4234 ;
                     4235 ;       NEEDS   "TO-DELETE":COUNTER ADDR.IN DE
                     4236
    0FE5             4237 FREE_COUNTER_                  ;HL should contain timer to delete addr.
    0FE5 23          4238                INC             HL      ;Get next after mode byte
    0FE6 5E          4239                LD              E,[HL]          ; into DE.
    0FE7 23          4240                INC             HL
    0FE8 56          4241                LD              D,[HL]
    0FE9 D5          4242                PUSH            DE              ;Save them for later.
    0FEA 2A73D3      4243                LD              HL,[TIMER_TABLE_BASE]
    0FED E5          4244                PUSH            HL              ;Save beginning of table.
    0FEE             4245 NEXT
    0FEE CB66        4246                BIT             EOT,[HL]        ;End of table?
    0FF0 202E        4247                JR              NZ,MOVE_IT
    0FF2 CB6E        4248                BIT             FREE,[HL]       ;Free?
    0FF4 2023        4249                JR              NZ,GET_NEXT     ;If so we don't want it.
    0FF6 7E          4250                LD              A,[HL]
    0FF7 E648        4251                AND             48H             ;Repeating and long?
    0FF9 FE48        4252                CP              48H
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    OFFB 201C        4253                JR              NZ,GET_NEXT     ;If NOT we don't want it.
    OFFD 23          4254                INC             HL
    OFFE 23          4255                INC             HL
    OFFF 7E          4256                LD              A,[HL]
    1000 BA          4257                CP              D
    1001 3816        4258                JR              C,GET_NEXT      ;If so we don't want it.
    1003 2008        4259                JR              NZ,SUBTRACT_4   ;However, if larger, change it.
    1005 2B          4260                DEC             HL
    1006 7E          4261                LD              A,[HL]
    1007 BB          4262                CP              E               ;Smaller?
    1008 380F        4263                JR              C,GET_NEXT      ;If so we don't want it.
    100A 2831        4264                JR              Z,EXIT          ;Error if equal!
    100C 23          4265                INC             HL              ;Set up HL for SUBTRACT_4
    1000             4266 SUBTRACT_4
    1000 56          4267                LD              D,[HL]
    100E 2B          4268                DEC             HL
    100F 5E          4269                LD              E,[HL]
    1010 1B          4270                DEC             DE              ;Reduce this addr. by 4.
    1011 1B          4271                DEC             DE
    1012 1B          4272                DEC             DE
    1013 1B          4273                DEC             DE
    1014 73          4274                LD              [HL],E          ;Replace reduced addr.
    1015 23          4275                INC             HL              ; back where we got it.
    1016 72          4276                LD              [HL],D
    1017 1800        4277                JR              GET_NEXT
    1019             4278 GET_NEXT
    1019 E1          4279                POP             HL              ;Now we can get next timer.
    101A 23          4280                INC             HL
    101B 23          4281                INC             HL
    101C 23          4282                INC             HL
    101D E5          4283                PUSH            HL
    101E 18CE        4284                JR              NEXT
    1020             4285 MOVE_IT
    1020 0600        4286                LD              B,0
    1022 B7          4287                OR              A               ;CLEAR CARRY
    1023 E1          4288                POP             HL
    1024 D1          4289                POP             DE              ;Get addr. of timer to delete.
    1025 E5          4290                PUSH            HL
    1026 2A7305      4291                LD              HL,[NEXT_TIMER_DATA_BYTE] ;Find # of bytes
    1029 ED52        4292                SBC             HL,DE           ;to move by subtraction.
    102B 4D          4293                LD              C,L             ;Save in counter reg.
    102C 68          4294                LD              L,E             ;Copy into HL.
    102D 62          4295                LD              H,D
    102E 23          4296                INC             HL              ;Find source addr.
    102F 23          4297                INC             HL
    1030 23          4298                INC             HL
    1031 23          4299                INC             HL
    1032 EDB0        4300                LDIR                            ;Move it!
    1034 010008      4301                LD              BC,8            ;Adjust Next available byte by -4 from LDIR dest.
    1037 ED42        4302                SBC             HL,BC           ;(or -8 from source of LDIR! [saves instrs.]).
    1039 227305      4303                LD              [NEXT_TIMER_DATA_BYTE],HL
    103C E1          4304                POP             HL
    103D             4305 EXIT
                     4306 ;              RET
                     4307
                     4308
    103D             4309 FREE_SET:
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4310 ;          RES  LONG,[HL]                  ;Reset repeat bit just in case
 103D                4311 FREE_EXIT:                                 ;Return
 103D C9             4312            RET
                     4313
                     4314
                     4315 ;Procedure Request Signal
                     4316 ;HL pair has length of timer
                     4317 ;Acc has zero for repeating timer any other value for a non_repeating type
                     4318 ;Signal number is returned in the Accumulator
                     4319 ;          COMM
                     4320 ;REPEAT_SIG_CODE:
                     4321 ;          DEFS 1
                     4322 ;TIMER_LENGTH:
                     4323 ;          DEFS 2
                     4324
                     4325            PROG
                     4326
 103E                4327 REQUEST_SIG_PARAM:
 103E 00020001       4328            DEFW 2,1,2
 1042 0002
                     4329
 1044                4330 REQUEST_SIGNALQ:
 1044 01103E         4331            LD   BC,REQUEST_SIG_PARAM
 1047 11738F         4332            LD   DE,REPEAT_SIG_CODE
 104A CD0098         4333            CALL PARAM_
 104D 2A73C0         4334            LD   HL,[TIMER_LENGTH]
 1050 3A738F         4335            LD   A,[REPEAT_SIG_CODE]
                     4336
 1053                4337 REQUEST_SIGNAL_:
 1053 4F             4338            LD   C,A                        ;Put Repeat Code into C register
 1054 EB             4339            EX   DE,HL                      ;Put length of timer into DE
 1055 2A7303         4340            LD   HL,[TIMER_TABLE_BASE]      ;Get Timer Base Address
 1058 AF             4341            XOR  A                          ;Init offset to First Table value
 1059 47             4342            LD   B,A
 105A                4343 TIMER1:
 105A CB6E           4344            BIT  FREE,[HL]                  ;See if current timer free
 105C 283E           4345            JR   Z,NEXT_TIMER1              ;If not go get the next timer
 105E E5             4346            PUSH HL
                     4347 ;          PUSH AF
 105F 7E             4348            LD   A,[HL]
 1060 E610           4349            AND  10H
 1062 F620           4350            OR   20H
 1064 77             4351            LD   [HL],A
 1065 AF             4352            XOR  A
                     4353 ;          POP  AF
 1066 B2             4354            OR   D
 1067 2008           4355            JR   NZ,LONG_TIMER
                     4356 ;          RES  FREE,[HL]                  ;Reset Free bit
                     4357 ;          OR   D                          ;Check for zero
                     4358 ;          JR   NZ,LONG_TIMER              ;If non_zero then its a long timer
                     4359 ;          RES  REPEAT,[HL]                ;Set for a NON_Repeating timer
                     4360 ;          RES  LONG,[HL]
                     4361 ;          LD   A,C                        ;Check for a short repeating timer
 1069 B1             4362            OR   C
 106A 2802           4363            JR   Z,NOT_A_REPEAT_TIMER       ;Don't reset repeat bit in mode byte if non_repeating
 106C CBF6           4364            SET  REPEAT,[HL]                ;Set repeat bit
 106E                4365 NOT_A_REPEAT_TIMER:
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   106E 23          4366            INC  HL                          ;Go to next table location
   106F 73          4367            LD   (HL),E                      ;Store timer length
   1070 23          4368            INC  HL
   1071 73          4369            LD   (HL),E                      ;Store timer length again in case of repeat
   1072 1842        4370            JR   INIT_TIMER_EXIT              ;All done so let's exit
   1074             4371 LONG_TIMER:
   1074 CBDE        4372            SET  LONG,(HL)                   ;Set long timer bit
   1076 79          4373            LD   A,C                         ;Check for a long repeat timer
   1077 B7          4374            OR   A
   1078 281B        4375            JR   Z,NOT_A_LONG_REPEAT         ;If zero then go to section for non_repeating timer
   107A D5          4376            PUSH DE                          ;Store timer length temporarily
   107B EB          4377            EX   DE,HL                       ;Swap regisers
   107C 2A7305      4378            LD   HL,(NEXT_TIMER_DATA_BYTE)   ;To get free space in long timer table
   107F EB          4379            EX   DE,HL                       ;Then swap back
   1080 CBF6        4380            SET  REPEAT,(HL)                 ;Set mode byte to repeating
   1082 23          4381            INC  HL
   1083 73          4382            LD   (HL),E                      ;Store low byte of timer address into the value word
   1084 23          4383            INC  HL
   1085 72          4384            LD   (HL),D                      ;Store high byte of timer address
   1086 EB          4385            EX   DE,HL                       ;Move address of data area into HL
   1087 D1          4386            POP  DE                          ;Get back the length of timer
   1088 73          4387            LD   (HL),E                      ;Store that in the data table
   1089 23          4388            INC  HL
   108A 72          4389            LD   (HL),D
   108B 23          4390            INC  HL
   108C 73          4391            LD   (HL),E                      ;Store it again
   108D 23          4392            INC  HL
   108E 72          4393            LD   (HL),D
   108F 23          4394            INC  HL
   1090 227305      4395            LD   (NEXT_TIMER_DATA_BYTE),HL   ;Store the next available data area for future use
   1093 1821        4396            JR   INIT_TIMER_EXIT
   1095             4397 NOT_A_LONG_REPEAT:
   1095 23          4398            INC  HL
   1096             4399 TIMER2:
   1096 73          4400            LD   (HL),E                      ;Store it again
   1097 23          4401            INC  HL
   1098 72          4402            LD   (HL),D
   1099 23          4403            INC  HL
   109A 181A        4404            JR   INIT_TIMER_EXIT
                    4405
   109C             4406 NEXT_TIMER1:
   109C CB66        4407            BIT  EOT,(HL)
   109E 2006        4408            JR   NZ,MAKE_NEW_TIMER
   10A0 23          4409            INC  HL                          ;Go to next mode byte
   10A1 23          4410            INC  HL                          ;
   10A2 23          4411            INC  HL                          ;
   10A3 04          4412            INC  B                           ;Count to next offset
   10A4 1884        4413            JR   TIMER1                      ;Go back up to init. tomer
                    4414
   10A6             4415 MAKE_NEW_TIMER:              ;Maximum of 255 signals allowed
   10A6 D5          4416            PUSH DE                          ;Save DE for a work register
   10A7 E5          4417            PUSH HL                          ;Save current timer address
   10A8 23          4418            INC  HL                          ;Go to next available memory location in the Timer Table
   10A9 23          4419            INC  HL
   10AA 23          4420            INC  HL
   10AB 04          4421            INC  B                           ;Increment the Signal count
   10AC 3630        4422            LD   (HL),30H                    ;Set to free and last timer
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    10AE EB          4423            EX   DE,HL                      ;Save momentarily
    10AF E1          4424            POP  HL                         ;Get back original timer
    10B0 CBA6        4425            RES  EOT,(HL)                   ;Reset previous last timer
    10B2 EB          4426            EX   DE,HL                      ;Get back current last timer
    10B3 D1          4427            POP  DE                         ;Restore DE register
    10B4 18A4        4428            JR   TIMER1                     ;Go back up and initialize counter for use
                     4429
    10B6             4430 INIT_TIMER_EXIT:
    10B6 E1          4431            POP  HL
    10B7 CBAE        4432            RES  FREE,(HL)
    10B9 78          4433            LD   A,B            ;Put the offset into the Accumulator for the user of routine
    10BA C9          4434            RET
                     4435
                     4436
                     4437 ;Procedure Test Signal
                     4438 ;Acc has the Signal number to be tested
                     4439 ;A value of True(1) or False(0) is returned in the Accumulator for the
                     4440 ;               Signal given.
                     4441 ;          COMM
                     4442 ;TEST_SIG_NUM:
                     4443 ;          DEFS 1
                     4444
                     4445            PROG
    10BB             4446 TEST_SIG_PARAM:
    10BB 00010001    4447            DEFW 1,1
                     4448
    10BF             4449 TEST_SIGNALQ
    10BF 01BB10      4450            LD   BC,TEST_SIG_PARAM
    10C2 1173C2      4451            LD   DE,TEST_SIG_NUM
    10C5 CD0098      4452            CALL PARAM
    10C8 3A73C2      4453            LD   A,(TEST_SIG_NUM)
                     4454
    10CB             4455 TEST_SIGNAL_
    10CB 4F          4456            LD   C,A                        ;Put Signal Code into C register
    10CC 2A73D3      4457            LD   HL,(TIMER_TABLE_BASE)      ;Get Timer Base Address
    10CF 47          4458            LD   B,A                        ;Save Signal
    10D0 110003      4459            LD   DE,3                       ;Set up offset for next timer
    10D3 B7          4460            OR   A                          ;See if first timer is a match
    10D4 2808        4461            JR   Z,SIGNAL_MATCH             ;If so go check it
                     4462
    10D6             4463 TEST1:                                     ;Loop to match timer table to desired timer
    10D6 CB66        4464            BIT  EOT,(HL)                   ;Check for end of table
    10D8 200C        4465            JR   NZ,SIGNAL_FALSE            ;If so then return a not done
    10DA 19          4466            ADD  HL,DE                      ;Now index to next timer
    10DB 0D          4467            DEC  C                          ;Decrement to the timer desired
    10DC 20F8        4468            JR   NZ,TEST1                   ;If not a timer desired timer then go back
                     4469
    10DE             4470 SIGNAL_MATCH:                              ;Here with a timer match
    10DE CB6E        4471            BIT  FREE,(HL)
    10E0 2004        4472            JR   NZ,SIGNAL_FALSE
    10E2 CB7E        4473            BIT  DONE,(HL)                  ;Check for timer done
    10E4 2003        4474            JR   NZ,SIGNAL_TRUE             ;If so then go return a True
    10E6             4475 SIGNAL_FALSE:                              ;Here to return a false for either
                     4476                                            ;a not done or non_existent timer
    10E6 AF          4477            XOR  A                          ;Put a false in Acc
    10E7 180A        4478            JR   TEST_EXIT                  ;Go to the exit
                     4479
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    10E9             4480 SIGNAL_TRUE:                    ;Here when timer is finished
    10E9 CB76        4481             BIT  REPEAT,[HL]    ;Check for repeating timer
    10EB 2002        4482             JR   NZ,SIGNAL_TRUE1;If so then just return True
    10ED CBEE        4483             SET  FREE,[HL]      ;Free current timer since not repeating
    10EF             4484 SIGNAL_TRUE1:
                     4485
    10EF CBBE        4486             RES  DONE,[HL]      ;***** Start add 4/30/82****
                     4487                                 ;Reset current timer to not done
    10F1 3E01        4488             LD   A,1            ;***** End add 4/30/82****
    10F3             4489 TEST_EXIT:                      ;Put a True in the Acc
    10F3 B7          4490             OR   A              ;Return
    10F4 C9          4491             RET
                     4492
                     4493             DATA
                     4494
    7303             4495 TIMER_TABLE_BASE:     DEFS 2
    7305             4496 NEXT_TIMER_DATA_BYTE: DEFS 2
                     4497
                     4498  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     4500 ;CONTROLLER SOFTWARE
               <0000> 4501 FIRE            EQU             0         ;BITS IN STATUS WORD TO CHECK
               <0001> 4502 JOY             EQU             1         ;WHETHER DEVICE IS ACTIVE
               <0002> 4503 SPIN            EQU             2
               <0003> 4504 ARM             EQU             3
               <0004> 4505 KBD             EQU             4
               <0002> 4506 PLYR_0          EQU             2         ;OFFSET TO CONTROLLER MEMORY
               <0007> 4507 PLYR_1          EQU             7
               <0007> 4508 SEG_0           EQU             07H       ;BIT MASK IN PLAYER STATUS
               <0018> 4509 SEG_1           EQU             18H       ;FOR SEGMENT 0 OR 1 DEVICES
               <0000> 4510 FIRE_OLD        EQU             0         ;OFFSETS TO DEBOUNCE STATUS
               <0001> 4511 FIRE_STATE      EQU             1         ;BUFFER: (OLD,STATE)
               <0002> 4512 JOY_OLD         EQU             2
               <0003> 4513 JOY_STATE       EQU             3
               <0004> 4514 SPIN_OLD        EQU             4
               <0005> 4515 SPIN_STATE      EQU             5
               <0006> 4516 ARM_OLD         EQU             6
               <0007> 4517 ARM_STATE       EQU             7
               <0008> 4518 KBD_OLD         EQU             8
               <0009> 4519 KBD_STATE       EQU             9
               <000F> 4520 KBD_MASK        EQU             0FH       ;MASK FOR INPUT DATA BYTE
               <0040> 4521 FIRE_MASK       EQU             40H
               <0040> 4522 ARM_MASK        EQU             40H
               <000F> 4523 JOY_MASK        EQU             0FH
               <0080> 4524 SPNR_MASK       EQU             10110000B
               <0005> 4525 NUM_DEV         EQU             5         ;NUMBER OF POSSIBLE DEVICES
                     4526 ;STACK            EQU              73FFH
                     4527 ;MODE_0_PORT      EQU              0BEH      ;D8
                     4528 ;MODE_1_PORT      EQU              0BFH      ;D9
               <000F> 4529 KBD_NULL        EQU             0FH
               <00FF> 4530 CTRL_1_PORT     EQU             0FFH
               <00FC> 4531 CTRL_0_PORT     EQU             0FCH
               <00C0> 4532 STRB_RST_PORT   EQU             0C0H            ;STROBE RESET PORT
               <0080> 4533 STRB_SET_PORT   EQU             080H            ;STROBE SET PORT
               <0000> 4534 CONTROLLER_0    EQU             0
               <0001> 4535 CONTROLLER_1    EQU             1
               <0000> 4536 STROBE_RESET    EQU             0
               <0001> 4537 STROBE_SET      EQU             1
                     4538 ************************ MACRO *********************************
                     4539
                     4540
                     4541 DELAY_10   MACRO
                     4542            CALL DELAY
                     4543            MEND
                     4544
                     4545 *******************************DATA*****************************
                     4546 *
                     4547 *  DECODER TABLE FOR THE KEYBOARD
                     4548 *
     10F5            4549 DEC_KBD_TBL
     10F5 0F         4550            DEFB KBD_NULL       ; NULL ENTRY
     10F6 06         4551            DEFB 6              ; '6'
     10F7 01         4552            DEFB 1              ; '1'
     10F8 03         4553            DEFB 3              ; '3'
     10F9 09         4554            DEFB 9              ; '9'
     10FA 00         4555            DEFB 0              ; '0'
     10FB 0A         4556            DEFB 10             ; '*'
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

   10FC 0F          4557            DEFB KBD_NULL          ; NULL ENTRY
   10FD 02          4558            DEFB 2                 ; '2'
   10FE 08          4559            DEFB 11                ; RESET
   10FF 07          4560            DEFB 7                 ; '7'
   1100 0F          4561            DEFB KBD_NULL          ; NULL ENTRY
   1101 05          4562            DEFB 5                 ; '5'
   1102 04          4563            DEFB 4                 ; '4'
   1103 08          4564            DEFB 8                 ; '8'
   1104 0F          4565            DEFB KBD_NULL          ; NULL ENTRY
                    4566
                    4567 ******************************SUBROUTINES**********************
                    4568
                    4569                 GLB             CONTROLLER_INIT
   1105             4570 CONTROLLER_INIT                  ;INITIALIZE CONTROLLER TO STROBE RESET
   1105 D3C0        4571            OUT  [STRB_RST_PORT],A
   1107 AF          4572            XOR  A
   1108 DD2A8008    4573            LD   IX,[CONTROLLER_MAP]
   110C DD23        4574            INC  IX
   110E DD23        4575            INC  IX
   1110 FD2173D7    4576            LD   IY,DBNCE_BUFF
   1114 060A        4577            LD   B,NUM_DEV*2
                    4578 *  CLEAR CONTROLLER MEMORY AND DEBOUNCE STATUS BUFFER
   1116             4579 CINIT1
   1116 DD7700      4580            LD   [IX+0],A
   1119 DD23        4581            INC  IX
   111B FD7700      4582            LD   [IY+0],A
   111E FD23        4583            INC  IY
   1120 FD7700      4584            LD   [IY+0],A
   1123 FD23        4585            INC  IY
   1125 05          4586            DEC  B
   1126 20EE        4587            JR   NZ,CINIT1
                    4588 * CLEAR REMAINING VARIABLES
   1128 3273EB      4589            LD   [SPIN_SW0_CT],A
   1128 3273EC      4590            LD   [SPIN_SW1_CT],A
   112E 3273EE      4591            LD   [S0_C0],A
   1131 3273EF      4592            LD   [S0_C1],A
   1134 3273F0      4593            LD   [S1_C0],A
   1137 3273F1      4594            LD   [S1_C1],A
   113A C9          4595            RET
                    4596
                    4597
                    4598
   1138 00          4599 DELAY      NOP                   ;DELAY AFTER STROBE, BEFORE READ
   113C C9          4600            RET
                    4601 *
                    4602 *  CONTROLLER READ ROUTINE
                    4603 *               INPUT:
                    4604 *                    H - CONTROLLER NUMBER
                    4605 *               OUTPUT:
                    4606 *                    A - RAW DATA
                    4607 *
                    4608
   113D             4609 CONT_READ
   113D 7C          4610            LD   A,H
   113E FE00        4611            CP   CONTROLLER_0          ;IF CONTROLLER<>0
   1140 2004        4612            JR   NZ,CONT_READ1         ;THEN READ PLAYER 1
   1142 DBFC        4613            IN   A,[CTRL_0_PORT]       ;ELSE READ PLAYER 0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1144 1802        4614           JR   CONT_READX
    1146             4615 CONT_READ1
    1146 DBFF        4616           IN   A,[CTRL_1_PORT]
    1148             4617 CONT_READX
    1148 2F          4618           CPL                          ;COMPLEMENT DATA SO 0=NO DATA
    1149 C9          4619           RET                                  ;1= DATA
                     4620
                     4621
                     4622 *
                     4623 *  CONTROLLER SCANNER ROUTINE
                     4624 *
                     4625                GLB            CONT_SCAN
    114A             4626 CONT_SCAN
    114A DBFC        4627           IN   A,[CTRL_0_PORT]         ;READ SEGMENT 0, BOTH PLAYERS
    114C 2F          4628           CPL
    114D 3273EE      4629           LD   [S0_C0],A
    1150 DBFF        4630           IN   A,[CTRL_1_PORT]
    1152 2F          4631           CPL
    1153 3273EF      4632           LD   [S0_C1],A
    1156 D380        4633           OUT  [STRB_SET_PORT],A       ;STROBE SEGMENT 1
    1158             4634           DELAY 10                     ;WAIT 10 MICROSECS
    1158 CD1138        +            CALL DELAY
    115B DBFC        4635           IN   A,[CTRL_0_PORT]         ;READ SEGMENT 1, BOTH PLAYERS
    115D 2F          4636           CPL
    115E 3273F0      4637           LD   [S1_C0],A
    1161 DBFF        4638           IN   A,[CTRL_1_PORT]
    1163 2F          4639           CPL
    1164 3273F1      4640           LD   [S1_C1],A
    1167 D3C0        4641           OUT  [STRB_RST_PORT],A       ;RESET TO SEGMENT 0
    1169 C9          4642           RET
                     4643
                     4644
                     4645 *
                     4646 *              UPDATE SPINNER SWITCH ROUTINE                *
                     4647 *
                     4648 *
                     4649
                     4650
                     4651
                     4652                GLB            UPDATE_SPINNER_
    116A             4653 UPDATE_SPINNER_
    116A DBFC        4654           IN   A,[CTRL_0_PORT]         ;GET DATA
    116C 2173EB      4655           LD   HL,SPIN_SW0_CT          ;ADDRESS OF SPINNER 0 COUNT
    116F CB67        4656           BIT  4,A                     ;IF INT BIT SET
    1171 2008        4657           JR   NZ,UPDATE_S1            ;THEN SPINNER 1
                     4658 *  CHECK DIRECTION                     ; ELSE SPINNER 0
    1173 CB6F        4659           BIT  5,A                     ; IF BIT 5 IS  SET
    1175 2003        4660           JR   NZ,UPDATE_R0            ; THEN GOING RIGHT
                     4661                                        ; ELSE LEFT
    1177 35          4662           DEC  [HL]                    ; DECREMENT SPINNER COUNTER
    1178 1801        4663           JR   UPDATE_S1               ; GO CHECK SPINNER 1
                     4664 ***  RIGHT SPINNER SWITCH
    117A             4665 UPDATE_R0
    117A 34          4666           INC  [HL]                    ;RIGHT, INCREMENT COUNTER
                     4667 * CHECK SPINNER 1
    117B DBFF        4668 UPDATE_S1 IN   A,[CTRL_1_PORT]         ;LOOK AT SPINNER 1 DATA
    117D CB67        4669           BIT  4,A                     ;IF INT BIT  SET
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   117F 2009        4670           JR   NZ,UPDATE_SPINX          ;THEN NOT SPINNER 1
   1181 23          4671           INC  HL                       ;ELSE SPINNER 1, BUMP HL
                    4672 *  CHECK DIRECTION
   1182 CB6F        4673           BIT  5,A                      ; IF BIT 5 IS SET
   1184 2003        4674           JR   NZ,UPDATE_R1             ; THEN GOING RIGHT
                    4675                                         ; ELSE LEFT
   1186 35          4676           DEC  [HL]                     ; DECREMENT SPINNER COUNTER
   1187 1801        4677           JR   UPDATE_SPINX
                    4678 ***  RIGHT SPINNER SWITCH
   1189             4679 UPDATE_R1
   1189 34          4680           INC  [HL]                     ;RIGHT, INCREMENT COUNTER
   118A             4681 UPDATE_SPINX
   118A C9          4682           RET
                    4683
                    4684
                    4685
                    4686 **************  DECODER ROUTINE  *************************
                    4687 *  THIS ROUTINE RETURNS DECODED RAW, UNDEBOUNCED DATA           *
                    4688 *  AND MAY OR MAY NOT BE REQUIRED BY O/S                *
                    4689 *
                    4690 *          INPUT:
                    4691 *              H - CONTROLLER NUMBER
                    4692 *              L - SEGMENT NUMBER
                    4693 *          OUTPUT:
                    4694 *                                  SEGMENT 0       SEGMENT 1
                    4695 *              H - BYTE 1            FIRE            ARM
                    4696 *              L - BYTE 2            JOYSTK          KBD
                    4697 *              E - BYTE 3            SPINNER
                    4698 *
                    4699 *
                    4700               GLB                   DECODER_
   1188             4701 DECODER_
   1188 7D          4702           LD   A,L
   118C FE01        4703           CP   STROBE_SET               ;IF L=1 THEN DECODE SEGMENT 1
   118E 281A        4704           JR   Z,DEC_SEG1
                    4705 *
                    4706 *  SEGMENT 0 (FIRE BUTTON, JOYSTICK)
                    4707 *  RETURN H=FIRE BUTTON, L=JOYSTICK, E=SPINNER
                    4708 *
                    4709 * DO SPINNER FIRST
   1190 0173EB      4710           LD   BC,SPIN_SW0_CT
   1193 7C          4711           LD   A,H
   1194 FE00        4712           CP   CONTROLLER_0             ;IF PLAYER=0 THEN GO DECODE
   1196 2801        4713           JR   Z,DEC_PLYR
   1198 03          4714           INC  BC                       ;ELSE INCREMENT BC TO SPINNER 1
   1199 0A          4715 DEC_PLYR  LD   A,[BC]                   ;GET SPINNER SWITCH COUNT
   119A 5F          4716           LD   E,A                      ;RETURN IT IN E
   119B AF          4717           XOR  A
   119C 02          4718           LD   [BC],A                   ;CLEAR OUT SPINNER SWITCH COUNT
                    4719
   119D CD113D      4720           CALL CONT_READ                ;GET OTHER DEVICE DATA FOR PLAYER
   11A0 57          4721           LD   D,A                      ;SAVE IT
   11A1 E60F        4722           AND  JOY_MASK                 ;MASK OUT JOYSTICK DATA
   11A3 6F          4723           LD   L,A                      ;RETURN IT IN L
                    4724
   11A4 7A          4725           LD   A,D                      ;RESTORE DATA
   11A5 E640        4726           AND  FIRE_MASK                ;MASK OUT FIRE BUTTON DATA
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    11A7 67          4727           LD   H,A                    ;RETURN IT IN H
    11A8 1816        4728           JR   DECODERX
                     4729 *
                     4730 * SEGMENT 1 (ARM BUTTON, KEYBOARD)
                     4731 *  RETURN H=ARM BUTTON, L=KEYBOARD
                     4732 *
    11AA             4733 DEC_SEG1
    11AA D380        4734           OUT  [STRB_SET_PORT],A      ;STROBE SEGMENT 1
    11AC CD113D      4735           CALL CONT_READ              ;READ SEGMENT 1 PLAYER DATA
    11AF 57          4736           LD   D,A                    ;SAVE IT
    11B0 D3C0        4737           OUT  [STRB_RST_PORT],A      ;RESET BACK TO SEGMENT 0
    11B2 E60F        4738           AND  KBD_MASK               ;MASK OUT KBD DATA
    11B4 2110F5      4739           LD   HL,DEC_KBD_TBL         ;GET DECODER TABLE ADDRESS
    11B7 0600        4740           LD   B,0
    11B9 4F          4741           LD   C,A
    11BA 09          4742           ADD  HL,BC                  ;COMPUTE OFFSET
    11BB 6E          4743           LD   L,[HL]                 ;RETURN KBD DATA IN L
                     4744
    11BC 7A          4745           LD   A,D                    ;RESTORE DATA
    11BD E640        4746           AND  ARM_MASK               ;MASK OUT ARM BUTTON DATA
    11BF 67          4747           LD   H,A                    ;RETURN IT IN H
    11C0             4748 DECODERX
    11C0 C9          4749           RET
                     4750
                     4751
                     4752
                     4753
                     4754 *  POLLING ROUTINE FOR ALL DEVICES IN CONTROLLER *
                     4755 *
                     4756 *
                     4757
                     4758                GLB            POLLER_
    11C1             4759 POLLER_
    11C1 CD114A      4760           CALL CONT_SCAN              ;GO SCAN ALL THE DATA FIRST
    11C4 FD2173D7    4761           LD   IY,DBNCE_BUFF          ;DEBOUNCE BUFFER POINTER
    11C8 DD2A8008    4762           LD   IX,[CONTROLLER_MAP]    ;CONTROLLER MEMORY POINTER
    11CC DDE5        4763           PUSH IX
    11CE DD7E00      4764           LD   A,[IX+0]          ;GET PLAYER 0 STATUS
    11D1 CB7F        4765           BIT  7,A               ;IF PLAYER 0 NOT ACTIVE
    11D3 281E        4766           JR   Z,CHK_PLYR_1      ;THEN CHECK PLAYER 1
                     4767 *  PLAYER 0 IS ACTIVE            ;ELSE
    11D5 47          4768           LD   B,A               ;SAVE STATUS
    11D6 110002      4769           LD   DE,PLYR_0         ;COMPUTE ADDRESS OF PLAYER_0
    11D9 DD19        4770           ADD  IX,DE             ;CONTROLLER_MEMORY
    11DB E607        4771           AND  SEG_0             ;IF SEGMENT 0 IS NOT ACTIVE
    11DD 2809        4772           JR   Z,CHK_SEG_01      ;THEN CHECK SEGMENT 1
                     4773 *  SEGMENT 0 ACTIVE              ;ELSE
    11DF 3A73EE      4774           LD   A,[S0_C0]
    11E2 2173E8      4775           LD   HL,SPIN_SW0_CT
    11E5 CD1220      4776           CALL DECODE_0          ;DECODE DATA FOR SEGMENT 0
    11E8             4777 CHK_SEG_01
    11E8 78          4778           LD   A,B               ;RESTORE PLAYER_0 STATUS
    11E9 E618        4779           AND  SEG_1             ;IF SEGMENT 1 IS NOT ACTIVE
    11EB 2806        4780           JR   Z,CHK_PLYR_1      ;THEN CHECK PLAYER 1
                     4781 *  SEGMENT 1 IS ACTIVE           ;ELSE
    11ED 3A73F0      4782           LD   A,[S1_C0]
    11F0 CD123F      4783           CALL DECODE_1          ;DECODE DATA FOR SEGMENT 1
```

```
CATION OBJECT CODE LINE     SOURCE LINE

  11F3            4784 CHK_PLYR_1
  11F3 DDE1       4785          POP  IX
  11F5 DD7E01     4786          LD   A,[IX+1]        ;GET PLAYER 1 STATUS
  11F8 CB7F       4787          BIT  7,A             ;IF PLAYER 1 IS NOT ACTIVE
  11FA 2823       4788          JR   Z,POLLER_X       ;THEN EXIT, ALL DONE
                  4789 *  PLAYER 1 IS ACTIVE
  11FC 47         4790          LD   B,A             ;SAVE PLAYER 1 STATUS
  11FD 11000A     4791          LD   DE,2*NUM_DEV    ;COMPUTE ADDRESS OF DEBOUNCE BUFFER
  1200 FD19       4792          ADD  IY,DE           ;FOR PLAYER 1
  1202 110007     4793          LD   DE,PLYR_1       ;COMPUTE ADDRESS OF CONTROLLER_MEMORY
  1205 DD19       4794          ADD  IX,DE           ;FOR PLAYER 1
  1207 E607       4795          AND  SEG_0           ;IF SEGMENT 0 IS NOT ACTIVE
  1209 2809       4796          JR   Z,CHK_SEG_11    ;THEN CHECK SEGMENT 1
                  4797 *  SEGMENT 0 IS ACTIVE       ;ELSE
  120B 3A73EF     4798          LD   A,[S0_C1]
  120E 2173EC     4799          LD   HL,SPIN_SW1_CT
  1211 CD1220     4800          CALL DECODE_0        ;DECODE DATA FOR SEGMENT 0
  1214            4801 CHK_SEG_11
  1214 78         4802          LD   A,B             ;RESTORE STATUS FOR PLAYER 1
  1215 E618       4803          AND  SEG_1           ;IF SEGMENT 1 IS NOT ACTIVE
  1217 2806       4804          JR   Z,POLLER_X       ;THEN EXIT, ALL DONE
                  4805 *  SEGMENT 1 IS ACTIVE       ;ELSE
  1219 3A73F1     4806          LD   A,[S1_C1]
  121C CD123F     4807          CALL DECODE_1        ;DECODE DATA FOR SEGMENT 1
  121F            4808 POLLER_X
  121F C9         4809          RET
                  4810
                  4811 *  DECODER ROUTINE FOR SEGMENT 0
                  4812 *
                  4813 *          INPUT:
                  4814 *               A - DATA
                  4815 *               B - DEVICE STATUS BYTE FOR CURRENT PLAYER
                  4816 *              HL - ADDRESS OF SPINNER DATA
                  4817 *              IX - POINTER TO CONTROLLER MEMORY
                  4818 *              IY - POINTER TO DEBOUNCE STATUS BUFFER
                  4819 *
  1220            4820 DECODE_0
  1220 4F         4821          LD   C,A             ;SAVE DATA
  1221 CB48       4822          BIT  JOY,B           ;IF JOYSTICK NOT ACTIVE
  1223 2804       4823          JR   Z,DEC_FIRE      ;THEN CHECK FIRE BUTTON
                  4824 *  JOYSTICK ACTIVE
  1225 CD1289     4825          CALL JOY_DBNCE       ;DEBOUNCE JOYSTICK DATA
  1228 79         4826          LD   A,C
  1229            4827 DEC_FIRE
  1229 CB40       4828          BIT  FIRE,B          ;IF FIRE BUTTON NOT ACTIVE
  122B 2804       4829          JR   Z,DEC_SPNR      ;THEN CHECK SPINNER
                  4830 *  FIRE BUTTON ACTIVE        ;ELSE
  122D CD1289     4831          CALL FIRE_DBNCE      ;DEBOUNCE FIRE BUTTON
  1230 79         4832          LD   A,C
  1231            4833 DEC_SPNR
  1231 CB50       4834          BIT  SPIN,B          ;IF SPINNER NOT ACTIVE
  1233 2809       4835          JR   Z,DECODE_0X     ;THEN EXIT DECODER
                  4836 *  SPINNER ACTIVE
  1235 7E         4837          LD   A,[HL]          ;SAVE SPINNER COUNT
  1236 DD8602     4838          ADD  A,[IX+SPIN]
  1239 DD7702     4839          LD   [IX+SPIN],A     ;IN CONTROLLER MEMORY
  123C AF         4840          XOR  A
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1230 77          4841           LD   [HL],A          ;CLEAR COUNTER
    123E             4842 DECODE_0X
    123E C9          4843           RET
                     4844
                     4845
                     4846 *  DECODER ROUTINE FOR SEGMENT 1
                     4847 *
                     4848 *          INPUT:
                     4849 *               A - DATA
                     4850 *               B - DEVICE STATUS BYTE FOR CURRENT PLAYER
                     4851 *              IX - POINTER TO CONTROLLER MEMORY
                     4852 *              IY - POINTER TO DEBOUNCE STATUS BUFFER
                     4853 *
    123F             4854 DECODE_1
    123F 4F          4855           LD   C,A             ;SAVE DATA
    1240 CB58        4856           BIT  ARM,B           ;IF ARM BUTTON NOT ACTIVE
    1242 2804        4857           JR   Z,DEC_KBD       ;THEN CHECK KEYBOARD
                     4858 *  ARM BUTTON ACTIVE           ;ELSE
    1244 CD12E9      4859           CALL ARM_DBNCE       ;DEBOUNCE ARM BUTTON
    1247 79          4860           LD   A,C
    1248             4861 DEC_KBD
    1248 CB60        4862           BIT  KBD,B           ;IF KEYBOARD NOT ACTIVE
    124A 2803        4863           JR   Z,DECODE_1X     ;THEN EXIT DECODER
                     4864 * KBD ACTIVE
    124C CD1250      4865           CALL KBD_DBNCE       ;DEBOUNCE KEYBOARD
    124F             4866 DECODE_1X
    124F C9          4867           RET
                     4868
                     4869
                     4870
                     4871 *  KEYBOARD DEBOUNCE ROUTINE  *
                     4872 *
                     4873 *          INPUT:
                     4874 *               A - RAW DATA
                     4875 *              IX - CONTROLLER MEMORY POINTER
                     4876 *              IY - DEBOUNCE STATE BUFFER
                     4877 *
    1250             4878 KBD_DBNCE
    1250 C5          4879           PUSH BC
    1251 D5          4880           PUSH DE
    1252 E5          4881           PUSH HL
    1253 E60F        4882           AND  KBD_MASK        ;MASK OUT VALID DATA
    1255 5F          4883           LD   E,A             ;SAVE IT
    1256 FD4608      4884           LD   B,[IY+KBD_OLD]  ;GET OLD DATA AND CURREN STATE
    1259 FD7E09      4885           LD   A,[IY+KBD_STATE]
    125C FE00        4886           CP   0               ;IF STATE <> 0
    125E 201A        4887           JR   NZ,KBD_ST1      ;THEN MUST BE STATE 1
                     4888 *  STATE = 0                   ;ELSE
    1260 7B          4889           LD   A,E             ;GET CURRENT DATA
    1261 B8          4890           CP   B               ;IF OLD=NEW
    1262 2805        4891           JR   Z,KBD_REG       ;THEN SAW DATA TWICE IN SEQUENCE
    1264 FD7308      4892           LD   [IY+KBD_OLD],E  ;ELSE FIRST TIME, SAVE CURRENT DATA
    1267 181C        4893           JR   KBD_EXIT
                     4894 *  SAW DATA TWICE IN SEQUENCE
    1269             4895 KBD_REG
    1269 3E01        4896           LD   A,1             ;SET STATE=1
    1268 FD7709      4897           LD   [IY+KBD_STATE],A
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

                    4898 *  DECODE KEYBOARD DATA
   126E 2110F5      4899            LD   HL,DEC_KBD_TBL  ;DECODE TABLE ADDRESS
   1271 1600        4900            LD   D,0            ;D/E RAW DATA
   1273 19          4901            ADD  HL,DE          ;COMPUTE ADDRESS INTO TABLE
   1274 7E          4902            LD   A,[HL]         ;DO TABLE LOOKUP
   1275 DD7704      4903            LD   [IX+KBD],A     ;SAVE IN CONTROLLER MEMORY @KBD
   1278 1808        4904            JR   KBD_EXIT
                    4905 *  STATE = 1
   127A             4906 KBD_ST1
   127A 7B          4907            LD   A,E            ;GET CURRENT DATA
   127B B8          4908            CP   B              ;IF OLD=NEW
   127C 2807        4909            JR   Z,KBD_EXIT     ;NO CHANGE IN STATE
   127E FD7308      4910            LD   [IY+KBD_OLD],E ;ELSE SAVE CURRENT DATA
   1281 AF          4911            XOR  A              ;SET STATE=0
   1282 FD7709      4912            LD   [IY+KBD_STATE],A
   1285             4913 KBD_EXIT
   1285 E1          4914            POP  HL
   1286 D1          4915            POP  DE
   1287 C1          4916            POP  BC
   1288 C9          4917            RET
                    4918
                    4919 *  FIRE BUTTON DEBOUNCE ROUTINE  *
                    4920 *
                    4921 *          INPUT:
                    4922 *              A - RAW DATA
                    4923 *             IX - CONTROLLER MEMORY POINTER
                    4924 *             IY - DEBOUNCE STATE BUFFER
                    4925 *
   1289             4926 FIRE_DBNCE
   1289 C5          4927            PUSH BC
   128A D5          4928            PUSH DE
   128B E640        4929            AND  FIRE_MASK      ;MASK OUT VALID DATA
   128D 5F          4930            LD   E,A            ;SAVE IT
   128E FD4600      4931            LD   B,[IY+FIRE_OLD]  ;GET OLD DATA AND CURRENT STATE
   1291 FD7E01      4932            LD   A,[IY+FIRE_STATE]
   1294 FE00        4933            CP   0              ;IF STATE <> 0
   1296 2013        4934            JR   NZ,FIRE_ST1    ;THEN MUST BE STATE 1
                    4935 *  STATE = 0                   ;ELSE
   1298 7B          4936            LD   A,E            ;GET CURRENT DATA
   1299 B8          4937            CP   B              ;IF OLD=NEW
   129A 2805        4938            JR   Z,FIRE_REG     ;THEN SAW DATA TWICE IN SEQUENCE
   129C FD7300      4939            LD   [IY+FIRE_OLD],E  ;ELSE FIRST TIME, SAVE CURRENT DATA
   129F 1815        4940            JR   FIRE_EXIT
                    4941 *  SAW DATA TWICE IN SEQUENCE
   12A1             4942 FIRE_REG
   12A1 3E01        4943            LD   A,1            ;SET STATE=1
   12A3 FD7701      4944            LD   [IY+FIRE_STATE],A
   12A6 DD7300      4945            LD   [IX+FIRE],E    ;SAVE IN CONTROLLER MEMORY @FIRE
   12A9 1808        4946            JR   FIRE_EXIT
                    4947 *  STATE = 1
   12AB             4948 FIRE_ST1
   12AB 7B          4949            LD   A,E            ;GET CURRENT DATA
   12AC B8          4950            CP   B              ;IF OLD=NEW
   12AD 2807        4951            JR   Z,FIRE_EXIT    ;NO CHANGE IN STATE
   12AF FD7300      4952            LD   [IY+FIRE_OLD],E  ;ELSE SAVE CURRENT DATA
   12B2 AF          4953            XOR  A              ;SET STATE=0
   12B3 FD7701      4954            LD   [IY+FIRE_STATE],A
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    12B6            4955 FIRE_EXIT
    12B6 D1         4956            POP  DE
    12B7 C1         4957            POP  BC
    12B8 C9         4958            RET
                    4959
                    4960 *  JOYSTICK DEBOUNCE ROUTINE  *
                    4961 *
                    4962 *          INPUT:
                    4963 *               A - RAW DATA
                    4964 *              IX - CONTROLLER MEMORY POINTER
                    4965 *              IY - DEBOUNCE STATE BUFFER
                    4966 *
    12B9            4967 JOY_DBNCE
    12B9 C5         4968            PUSH BC
    12BA D5         4969            PUSH DE
    12BB E60F       4970            AND  JOY_MASK        ;MASK OUT VALID DATA
    12BD 5F         4971            LD   E,A             ;SAVE IT
    12BE FD4602     4972            LD   B,[IY+JOY_OLD]  ;GET OLD DATA AND CURRENT STATE
    12C1 FD7E03     4973            LD   A,[IY+JOY_STATE]
    12C4 FE00       4974            CP   0               ;IF STATE <> 0
    12C6 2013       4975            JR   NZ,JOY_ST1      ;THEN MUST BE STATE 1
                    4976 *  STATE = 0                    ;ELSE
    12C8 7B         4977            LD   A,E             ;GET CURRENT DATA
    12C9 B8         4978            CP   B               ;IF OLD=NEW
    12CA 2805       4979            JR   Z,JOY_REG       ;THEN SAW DATA TWICE IN SEQUENCE
    12CC FD7302     4980            LD   [IY+JOY_OLD],E  ;ELSE FIRST TIME, SAVE CURRENT DATA
    12CF 1815       4981            JR   JOY_EXIT
                    4982 *  SAW DATA TWICE IN SEQUENCE
    12D1            4983 JOY_REG
    12D1 3E01       4984            LD   A,1             ;SET STATE=1
    12D3 FD7703     4985            LD   [IY+JOY_STATE],A
    12D6 DD7301     4986            LD   [IX+JOY],E      ;SAVE IN CONTROLLER MEMORY @JOY
    12D9 1808       4987            JR   JOY_EXIT
                    4988 *  STATE = 1
    12DB            4989 JOY_ST1
    12DB 7B         4990            LD   A,E             ;GET CURRENT DATA
    12DC B8         4991            CP   B               ;IF OLD=NEW
    12DD 2807       4992            JR   Z,JOY_EXIT      ;NO CHANGE IN STATE
    12DF FD7302     4993            LD   [IY+JOY_OLD],E  ;ELSE SAVE CURRENT DATA
    12E2 AF         4994            XOR  A               ;SET STATE=0
    12E3 FD7703     4995            LD   [IY+JOY_STATE],A
    12E6            4996 JOY_EXIT
    12E6 D1         4997            POP  DE
    12E7 C1         4998            POP  BC
    12E8 C9         4999            RET
                    5000
                    5001 *  ARM BUTTON DEBOUNCE ROUTINE  *
                    5002 *
                    5003 *          INPUT:
                    5004 *               A - RAW DATA
                    5005 *              IX - CONTROLLER MEMORY POINTER
                    5006 *              IY - DEBOUNCE STATE BUFFER
                    5007 *
    12E9            5008 ARM_DBNCE
    12E9 C5         5009            PUSH BC
    12EA D5         5010            PUSH DE
    12EB E640       5011            AND  ARM_MASK        ;MASK OUT VALID DATA
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

   12ED 5F          5012            LD   E,A              ;SAVE IT
   12EE FD4606      5013            LD   B,[IY+ARM_OLD]   ;GET OLD DATA AND CURRENT STATE
   12F1 FD7E07      5014            LD   A,[IY+ARM_STATE]
   12F4 FE00        5015            CP   0                ;IF STATE <> 0
   12F6 2013        5016            JR   NZ,ARM_ST1       ;THEN MUST BE STATE 1
                    5017 *  STATE = 0                     ;ELSE
   12F8 7B          5018            LD   A,E              ;GET CURRENT DATA
   12F9 B8          5019            CP   B                ;IF OLD=NEW
   12FA 2805        5020            JR   Z,ARM_REG        ;THEN SAW DATA TWICE IN SEQUENCE
   12FC FD7306      5021            LD   [IY+ARM_OLD],E   ;ELSE FIRST TIME, SAVE CURRENT DATA
   12FF 1815        5022            JR   ARM_EXIT
                    5023 *  SAW DATA TWICE IN SEQUENCE
    1301            5024 ARM_REG
    1301 3E01       5025            LD   A,1              ;SET STATE=1
    1303 FD7707     5026            LD   [IY+ARM_STATE],A
    1306 DD7303     5027            LD   [IX+ARM],E       ;SAVE IN CONTROLLER MEMORY @ARM
    1309 1808       5028            JR   ARM_EXIT
                    5029 *  STATE = 1
    1308            5030 ARM_ST1
    1308 7B         5031            LD   A,E              ;GET CURRENT DATA
    130C B8         5032            CP   B                ;IF OLD=NEW
    130D 2807       5033            JR   Z,ARM_EXIT       ;NO CHANGE IN STATE
    130F FD7306     5034            LD   [IY+ARM_OLD],E   ;ELSE SAVE CURRENT DATA
    1312 AF         5035            XOR  A                ;SET STATE=0
    1313 FD7707     5036            LD   [IY+ARM_STATE],A
    1316            5037 ARM_EXIT
    1316 D1         5038            POP  DE
    1317 C1         5039            POP  BC
    1318 C9         5040            RET
                    5041
                    5042
                    5043 ;               EXT             CONTROLLER_MAP
                    5044 * THIS IS AN EXTERNAL POINTER ( DEFINED IN THE CARTRIDGE) TO THE
                    5045 * CARTRIDGE PROGRAMMER'S CONTROLLER MAP AREA.
                    5046
                    5047 * THE CARTRIDGE PROGRAMMER IS RESPONSIBLE FOR MAINTAINING THIS AREA.
                    5048
                    5049                DATA
    73D7            5050 DBNCE_BUFF     DEFS            NUM_DEV*4
    73EB            5051 SPIN_SW0_CT    DEFS            1
    73EC            5052 SPIN_SW1_CT    DEFS            1
    73ED            5053 STROBE_FLG     DEFS            1
    73EE            5054 S0_C0          DEFS            1
    73EF            5055 S0_C1          DEFS            1
    73F0            5056 S1_C0          DEFS            1
    73F1            5057 S1_C1          DEFS            1
                    5058  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    5060
                    5061 ************************** EXTERNAL SYMBOLS ***************************
                    5062
                    5063 * EXTERNAL ROUTINES FROM OS
                    5064
                    5065                ;EXT INIT_TABLE
                    5066                ;EXT PUT_VRAM
                    5067                ;EXT GAME_NAME
                    5068                ;EXT WRITE_REGISTER
                    5069                ;EXT READ_REGISTER
                    5070                ;EXT WRITE_VRAM
                    5071                ;EXT START_GAME
                    5072
                    5073 ************************** DEFINITIONS ****************************
                    5074
          <00BE>    5075 MODE_0_PORT    EQU 0BEH
          <00BF>    5076 MODE_1_PORT    EQU 0BFH
                    5077
                    5078 ***************************** EXPORTS *****************************
                    5079
                    5080                GLB ASCII_TBL                   ;POINTER TO UPPERCASE ASCII GENERATORS
                    5081                GLB NUMBER_TBL                  ;POINTER TO 0-9 GENERATORS
                    5082                GLB DISPLAY_LOGO                ;DISPLAY COLECOVISION LOGO
                    5083                GLB LOAD_ASCII_                 ;LOAD PATTERN GEN TABLE WITH FULL ASCII SET
                    5084                GLB FILL_VRAM_                  ;FILL VRAM WITH A VALUE
                    5085                GLB MODE_1_                     ;SET UP MODE 1 GRAPHICS
                    5086
                    5087 ***************************  DESCRIPTION *****************************
                    5088 *
                    5089 *  DISPLAY_LOGO  DISPLAYS THE COLECO LOGO SCREEN WITH COLECOVISION
                    5090 *                ON A BLACK BACKGROUND.  THE GAME TITLE, MANUFACTURER,
                    5091 *                AND COPYRIGHT YEAR ARE OBTAINED FROM THE CARTRIDGE
                    5092 *                AND OVERLAYED ONTO THE LOGO SCREEN.  THE LOGO IS THEN
                    5093 *                DISPLAYED FOR 10 SECONDS AFTER WHICH TIME A JUMP TO
                    5094 *                THE GAME START ADDRESS IS EXECUTED.
                    5095 *
                    5096 *                IF NO CARTRIDGE IS PRESENT A DEFAULT MESSAGE IS
                    5097 *                DISPLAYED, INSTRUCTING THE OPERATOR TO:
                    5098 *
                    5099 *                      "TURN GAME OFF"
                    5100 *                      "BEFORE INSERTING CARTRIDGE"
                    5101 *                      "OR EXPANSION MODULE."
                    5102 *                      "[COPYRIGHT SYMBOL] 1982 COLECO"
                    5103 *                THIS MESSAGE IS DISPLAYED FOR 60 SECONDS, THE SCREEN
                    5104 *                IS THEN BLANKED AND FINALLY A SOFT HALT (JP $) IS
                    5105 *                EXECUTED LOCKING UP THE PROGRAM UNTIL THE UNIT IS
                    5106 *                RESET.
                    5107 *
                    5108 *                DISPLAY_LOGO EXITS WITH THE VDP IN MODE 1, THE SCREEN
                    5109 *                BLANKED, AND THE ASCII CHARACTER SET IN VRAM.
                    5110 *                THE MEMORY MAP IS AS FOLLOWS:
                    5111 *
                    5112 *                          VDP MEMORY MAP
                    5113 *                3800H-3FFFH  SPRITE GENERATOR TABLE
                    5114 *                2000H-37FFH  PATTERN COLOR TABLE
                    5115 *                1B00H-1B7FH  SPRITE ATTRIBUTE TABLE
                    5116 *                1800H-1AFFH  PATTERN NAME TABLE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    5117 *               0000H-17FFH  PATTERN GENERATOR TABLE
                    5118 *
                    5119 ********************************************************************************
                    5120
                    5121 ***************************** DISPLAY LOGO *****************************
                    5122
                    5123                 PROG
                    5124
                    5125 *  FILL VRAM WITH 0'S
    1319 210000     5126 DISPLAY_LOGO    LD HL,0
    131C 114000     5127                 LD DE,16384
    131F 3E00       5128                 LD A,0
    1321 CD18D4     5129                 CALL FILL_VRAM_
                    5130
                    5131 *  SET UP VDP WITH MODE_1
    1324 CD18E9     5132                 CALL MODE_1_
                    5133
                    5134 ******************* WRITE OUT PATTERN GEN TABLE *****************
                    5135
                    5136 *  WRITE OUT ASCII GENERATOR TABLES
    1327 CD1927     5137                 CALL LOAD_ASCII_
                    5138
                    5139 *  WRITE OUT GRAPHICS GENERATORS
    132A 2118A3     5140                 LD HL,OBJ_TABLE          ;POINT TO TABLE OF PTN GEN NUMBERS
    132D 110060     5141                 LD DE,60H                ;ITEM LOCATION IN VRAM PATTERN_GEN_TABLE
    1330 E5         5142 WRITE_LOOP      PUSH HL                  ;SAVE LOCATION OF CURRENT CONSTRUCTION
    1331 D5         5143                 PUSH DE                  ;SAVE VRAM ITEM #
                    5144 *  CALCULATE GENERATOR LOCATION
    1332 7E         5145                 LD A,[HL]
    1333 FEFF       5146                 CP 0FFH                  ;HAVE WE PROCESSED ALL GENERATORS?
    1335 281B       5147                 JR Z,DONE_LOGO           ; YES.  WE'RE ALL DONE
    1337 47         5148                 LD B,A                   ; NO. B=NUMBER FROM OBJ_TABLES
    1338 04         5149                 INC B
    1339 2114C3     5150                 LD HL,LOGO_GEN           ;POINT TO ROM GENERATOR TABLE
    133C 110008     5151                 LD DE,8                  ;WE'RE GOING TO ADD 8 FOR EVERY
    133F 1015       5152 ADDR_ADJ        DJNZ ADD_8               ;GENERATOR INTO THE ROM GEN TBL
                    5153
                    5154 *  DONE ADJUSTING ROM GENERATOR ADDRESS
                    5155
    1341 D1         5156                 POP DE                   ;RESTORE ITEM # IN VRAM
    1342 D5         5157                 PUSH DE                  ;SAVE IT
    1343 FD210001   5158                 LD IY,1                  ;NUMBER OF GENERATORS
    1347 3E03       5159                 LD A,3                   ;PATTERN_GEN_TABLE CODE
    1349 CD1FBE     5160                 CALL PUT_VRAM            ;HL=ROM ADDRESS
    134C D1         5161                 POP DE                   ;RESTORE ITEM # IN VRAM
    134D E1         5162                 POP HL                   ;RESTORE CONSTRUCTION ADDRESS
    134E 13         5163                 INC DE                   ;SET UP FOR NEXT ITEM
    134F 23         5164                 INC HL
    1350 18DE       5165                 JR WRITE_LOOP            ;KEEP GOING UNTIL DONE
                    5166
    1352 D1         5167 DONE_LOGO       POP DE                   ;GOT TO POP FOR EVERY PUSH
    1353 E1         5168                 POP HL
    1354 1803       5169                 JR WRITE_NAMES           ;HOP AROUND ADD_8
                    5170
    1356 19         5171 ADD_8           ADD HL,DE                ;POINT TO NEXT GENERATOR
    1357 18E6       5172                 JR ADDR_ADJ
                    5173
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    5174 ******************** WRITE OUT PATTERN_NAME_TABLE ****************
                    5175
                    5176 *  WRITE OUT PATTERN_NAME_TABLE
  1359 21144D       5177 WRITE_NAMES     LD HL,LOGO_NAMES          ;WRITE OUT TOP HALF
  135C 110085       5178                 LD DE,133                 ;OF COLCOVISION
  135F FD210016     5179                 LD IY,22
  1363 3E02         5180                 LD A,2
  1365 CD1FBE       5181                 CALL PUT_VRAM
                    5182
  1368 211463       5183                 LD HL,LOGO_NAMES+22       ;WRITE OUT BOTTOM HALF
  1368 1100A5       5184                 LD DE,165                 ;OF COLECOVISION
  136E FD210016     5185                 LD IY,22
  1372 3E02         5186                 LD A,2
  1374 CD1FBE       5187                 CALL PUT_VRAM
                    5188
                    5189
  1377 2114C1       5190                 LD HL,TRADEMARK           ;WRITE OUT TM
  137A 110098       5191                 LD DE,155
  137D FD210002     5192                 LD IY,2
  1381 3E02         5193                 LD A,2
  1383 CD1FBE       5194                 CALL PUT_VRAM
                    5195
                    5196 * SET UP DEFAULT COPYRIGHT MESSAGE
  1386 2114B4       5197                 LD HL,LOGO_NAMES+103      ;WRITE OUT c 1982 COLECO
  1389 1102AA       5198                 LD DE,682
  138C FD210000     5199                 LD IY,13
  1390 3E02         5200                 LD A,2
  1392 CD1FBE       5201                 CALL PUT_VRAM
                    5202
                    5203 ******************** WRITE OUT COLOR_NAME_TABLE ****************
                    5204
  1395 21143B       5205                 LD HL,LOGO_COLORS
  1398 110000       5206                 LD DE,0
  1398 3E04         5207                 LD A,4
  139D FD210012     5208                 LD IY,18
  13A1 CD1FBE       5209                 CALL PUT_VRAM
                    5210
                    5211 ***************************** ENABLE DISPLAY **********************
                    5212
                    5213 * ENABLE DISPLAY
  13A4 0601         5214                 LD B,1
  13A6 0EC0         5215                 LD C,11000000B
  13A8 CD1FD9       5216                 CALL WRITE_REGISTER
                    5217
                    5218 * CARTRIDGE TEST
  13AB 218000       5219                 LD HL,8000H               ;IF A CARTRIDGE IS PRESENT
  13AE 7E           5220                 LD A,[HL]                 ;LOCATION 8000H WILL = 0AAH
  13AF FEAA         5221                 CP 0AAH                   ;AND 8001H WILL = 55H
  13B1 204C         5222                 JR NZ,NO_CARTRIDGE        ;    NOT PRESENT
  13B3 23           5223                 INC HL
  13B4 7E           5224                 LD A,[HL]
  13B5 FE55         5225                 CP 55H
  13B7 2046         5226                 JR NZ,NO_CARTRIDGE
                    5227                                           ;    NOT PRESENT
                    5228 * CARTRIDGE PRESENT
                    5229 * DISPLAY GAME TITLE
  13B9 218024       5230                 LD HL,GAME_NAME
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

   13BC CD1946      5231                CALL PARSE          ;GET LENGTH OF STRING
   13BF 118024      5232                LD DE,GAME_NAME     ;STARTING LOCATION OF 1ST STRING
   13C2 210201      5233                LD HL,513           ;LOCATION (0_767) TO START PRINTING
   13C5 CD1951      5234                CALL CENTER_PRT     ;PRINT IT
                    5235
                    5236 * DISPLAY COMPANY NAME
   13C8 218024      5237                LD HL,GAME_NAME     ;GET PAST 1ST STRING
   13CB CD1946      5238                CALL PARSE
   13CE 23          5239                INC HL
   13CF 54          5240                LD D,H              ;STARTING LOCATION OF 2ND STRING
   13D0 5D          5241                LD E,L              ;SAVED IN DE
   13D1 CD1946      5242                CALL PARSE          ;GET LENGTH OF STRING
   13D4 2101C1      5243                LD HL,449           ;LOCATION (0_767) TO START PRINTING
   13D7 CD1951      5244                CALL CENTER_PRT     ;PRINT IT
                    5245
                    5246 * CHANGE DATE
   13DA 218024      5247                LD HL,GAME_NAME     ;USE PARSE TO ADVANCE HL TO
   13DD CD1946      5248                CALL PARSE          ;COPYRIGHT YEAR
   13E0 23          5249                INC HL
   13E1 CD1946      5250                CALL PARSE
   13E4 23          5251                INC HL
   13E5 1102AC      5252                LD  DE,684          ;SCREEN LOCATION
   13E8 FD210004    5253                LD IY,4             ;# OF DIGITS
   13EC 3E02        5254                LD A,2
   13EE CD1FBE      5255                CALL PUT_VRAM
                    5256
                    5257 * DISPLAY 10 SECONDS
   13F1 CD1968      5258                CALL DELAY_10
                    5259
                    5260 * TURN OFF DISPLAY
   13F4 0601        5261                LD B,1
   13F6 0E80        5262                LD C,10000000B
   13F8 CD1FD9      5263                CALL WRITE_REGISTER
                    5264
                    5265 * EXIT LOGO
   13FB 2A800A      5266                LD HL,[START_GAME]          ;A CARTRIDGE WAS PRESENT
   13FE E9          5267                JP [HL]                     ;SO JUMP TO IT
                    5268
                    5269 *  WRITE OUT PATTERN_NAME_TABLE
   13FF 211479      5270 NO_CARTRIDGE   LD HL,LOGO_NAMES+44         ;NO CARTRIDGE PRESENT
   1402 1101AA      5271                LD DE,426                   ;DISPLAY DEFAULT
   1405 FD210000    5272                LD IY,13                    ;MESSAGE
   1409 3E02        5273                LD A,2
   140B CD1FBE      5274                CALL PUT_VRAM
                    5275
   140E 211486      5276                LD HL,LOGO_NAMES+57         ;"TURN GAME OFF"
   1411 1101E4      5277                LD DE,484                   ;"TO INSERT CARTRIDGE"
   1414 FD21001A    5278                LD IY,26                    ;"OR EXPANSION MODULE"
   1418 3E02        5279                LD A,2
   141A CD1FBE      5280                CALL PUT_VRAM
                    5281
   141D 2114A0      5282                LD HL,LOGO_NAMES+83
   1420 110227      5283                LD DE,551
   1423 FD210014    5284                LD IY,20
   1427 3E02        5285                LD A,2
   1429 CD1FBE      5286                CALL PUT_VRAM
                    5287
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   142C 218A00       5288                LD HL,8A00H                   ;DISPLAY 60 SECONDS
   142F CD1968       5289                CALL TIMER_1
                     5290 * TURN OFF DISPLAY
   1432 0601         5291                LD B,1                        ;BLANK SCREEN
   1434 0E80         5292                LD C,10000000B
   1436 CD1FD9       5293                CALL WRITE_REGISTER
   1439 18FE         5294                JR $                          ;SOFT HALT
                     5295
                     5296 ********************************************************************
                     5297 *                                                                  *
                     5298 *                        DATA TABLES                               *
                     5299 *                                                                  *
                     5300 ********************************************************************
                     5301
                     5302 ************************ COLOR_NAME_TABLE ************************
                     5303
   1438 000000F0F0   5304 LOGO_COLORS    HEX 00,00,00,F0,F0,F0,F0,F0,F0,F0,F0,F0
   1440 F0F0F0F0F0
   1445 F0F0
   1447 D080908030   5305                HEX D0,80,90,80,30,40
   144C 40
                     5306
                     5307 ************************ PATTERN_NAME_TABLE ************************
                     5308
         <144D>      5309 LOGO_NAMES     EQU $
                     5310
   144D 6061686970   5311                HEX 60,61,68,69,70,71,78,79,80,81,88,89
   1452 7178798081
   1457 8889
   1459 64656C7475   5312                HEX 64,65,6C,74,75,7C,84,85,8C,8D
   145E 7C84858C8D
                     5313
   1463 62636A6872   5314                HEX 62,63,6A,6B,72,73,7A,7B,82,83,8A
   1468 737A7B8283
   146D 8A
   146E 8B666767D76  5315                HEX 8B,66,67,6D,76,77,7D,86,87,8E,8F
   1473 777D86878E
   1478 8F
                     5316
   1479 5455524E20   5317                DEFB "TURN GAME OFF"
   147E 47414D4520
   1483 4F4646
   1486 4245464F52   5318                DEFB "BEFORE INSERTING CARTRIDGE"
   1488 4520494E53
   1490 455254494E
   1495 4720434152
   149A 5452494447
   149F 45
   14A0 4F52204558   5319                DEFB "OR EXPANSION MODULE."
   14A5 50414E5349
   14AA 4F4E204D4F
   14AF 44554C452E
   14B4 1D20313938   5320                DEFB 1DH," 1982 COLECO"
   14B9 3220434F4C
   14BE 45434F
                     5321
   14C1 1E1F         5322 TRADEMARK      HEX  1E,1F
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

                    5323
                    5324
                    5325
                    5326
                    5327 ********************* PATTERN GENERATOR TABLES *********************
                    5328
          <14C3>    5329 LOGO_GEN        EQU $
                    5330
                    5331 *GR0
  14C3 0000000000   5332                 HEX            00,00,00,00,00,00,00,00
  14C8 000000
                    5333 *GR1
  14CB 3F7FFFFFF3   5334                 HEX            3F,7F,FF,FF,F3,F3,F0,F0
  14D0 F3F0F0
                    5335 *GR2
  14D3 0080C0C0C0   5336                 HEX            00,80,C0,C0,C0,C0,00,00
  14D8 C00000
                    5337 *GR3
  14DB 3F7FFFFFF3   5338                 HEX            3F,7F,FF,FF,F3,F3,F3,F3
  14E0 F3F3F3
                    5339 *GR4
  14E3 0080C0C0C0   5340                 HEX            00,80,C0,C0,C0,C0,C0,C0
  14E8 C0C0C0
                    5341 *GR5
  14EB F0F0F0F0F0   5342                 HEX            F0,F0,F0,F0,F0,F0,F0,F0
  14F0 F0F0F0
                    5343 *GR6
  14F3 FFFFFFF0F0   5344                 HEX            FF,FF,FF,F0,F0,FF,FF,FF
  14F8 FFFFFF
                    5345 *GR7
  14FB C0C0C00000   5346                 HEX            C0,C0,C0,00,00,00,00,00
  1500 000000
                    5347 *GR8
  1503 F1F1F17B7B   5348                 HEX            F1,F1,F1,7B,7B,7B,3F,3F
  1508 7B3F3F
                    5349 *GR9
  1508 E0E0E0C0C0   5350                 HEX            E0,E0,E0,C0,C0,C0,80,80
  1510 C08080
                    5351 *GR10
  1513 1F3F7F7978   5352                 HEX            1F,3F,7F,79,78,7F,7F,3F
  1518 7F7F3F
                    5353 *GR11
  151B 80C0E0E000   5354                 HEX            80,C0,E0,E0,00,80,C0,E0
  1520 80C0E0
                    5355 *GR12
  1523 F3F3FBFBFB   5356                 HEX            F3,F3,FB,FB,FB,FF,FF,FF
  1528 FFFFFF
                    5357 *GR13
  152B C0C0C0C0C0   5358                 HEX            C0,C0,C0,C0,C0,C0,C0,C0
  1530 C0C0C0
                    5359 *GR14
  1533 F3F3FFFF7F   5360                 HEX            F3,F3,FF,FF,7F,3F,00,00
  1538 3F0000
                    5361 *GR15
  153B C0C0C0C080   5362                 HEX            C0,C0,C0,C0,80,00,00,00
  1540 000000
                    5363 *GR16
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1543 F0F0FFFFFF  5364                HEX             F0,F0,FF,FF,FF,FF,00,00
    1548 FF0000
                     5365 *GR17
    154B 0000C0C0C0  5366                HEX             00,00,C0,C0,C0,C0,00,00
    1550 C00000
                     5367 *GR18
    1553 3F1F1F1F0E  5368                HEX             3F,1F,1F,1F,0E,0E,00,00
    1558 0E0000
                     5369 *GR19
    155B 8000000000  5370                HEX             80,00,00,00,00,00,00,00
    1560 000000
                     5371 *GR20
    1563 F0F0F0F0F0  5372                HEX             F0,F0,F0,F0,F0,F0,00,00
    1568 F00000
                     5373 *GR21
    1568 1F01797F3F  5374                HEX             1F,01,79,7F,3F,1F,00,00
    1570 1F0000
                     5375 *GR22
    1573 E0E0E0E0C0  5376                HEX             E0,E0,E0,E0,C0,80,00,00
    1578 800000
                     5377 *GR23
    157B FFF7F7F7F3  5378                HEX             FF,F7,F7,F7,F3,F3,00,00
    1580 F30000
                     5379 *GR24
    1583 C0C0C0C0C0  5380                HEX             C0,C0,C0,C0,C0,C0,00,00
    1588 C00000
                     5381
          <1588>     5382 ASC_TABLE       EQU $
                     5383 * c=1D
    1588 7E81BDA1A1  5384                HEX             7E,81,BD,A1,A1,BD,81,7E      ;cOPYRIGHT
    1590 BD817E
                     5385 * t=1E
    1593 1F04040400  5386                HEX             1F,4,4,4,0,0,0,0             ;tRADE
    1598 000000
                     5387 * m=1F
    159B 446C545400  5388                HEX             44,6C,54,54,0,0,0,0          ;mARK
    15A0 000000
                     5389 *   =20
    15A3 0000000000  5390 SPACE          DEFB            0,0,0,0,0,0,0,0
    15A8 000000
                     5391 * !=21
    15AB 2020202020  5392                DEFB            20H,20H,20H,20H,20H,0,20H,00H
    15B0 002000
                     5393 * "=22
    15B3 5050500000  5394                DEFB            50H,50H,50H,0,0,0,0,0
    15B8 000000
                     5395 * #=23
    15BB 5050F850F8  5396                DEFB            50H,50H,0F8H,50H,0F8H,50H,50H,0
    15C0 505000
                     5397 * $=24
    15C3 2078A07028  5398                DEFB            20H,78H,0A0H,70H,28H,0F0H,020H,0
    15C8 F02000
                     5399 * %=25
    15CB C0C8102040  5400                DEFB            0C0H,0C8H,10H,20H,40H,98H,18H,0
    15D0 981800
                     5401 * &=26
    15D3 40A0A040A8  5402                DEFB            40H,0A0H,0A0H,40H,0A8H,90H,68H,0
```

```
CATION OBJECT CODE LINE     SOURCE LINE

  15D8 906800
                    5403 * '=27
  15DB 2020200000   5404                DEFB          20H,20H,20H,0,0,0,0,0
  15E0 000000
                    5405 * (=28
  15E3 2040808080   5406                DEFB          20H,40H,80H,80H,80H,40H,20H,0
  15E8 402000
                    5407 * )=29
  15EB 2010080808   5408                DEFB          20H,10H,08H,08H,08H,10H,20H,0
  15F0 102000
                    5409 * *=2A
  15F3 20A8702070   5410                DEFB          20H,0A8H,70H,20H,70H,0A8H,20H,0
  15F8 A82000
                    5411 * +=2B
  15FB 002020F820   5412                DEFB          0,20H,20H,0F8H,20H,20H,0,0
  1600 200000
                    5413 * ,=2C
  1603 0000000020   5414                DEFB          0,0,0,0,20H,20H,40H,0
  1608 204000
                    5415 * -=2D
  160B 000000F800   5416                DEFB          0,0,0,0F8H,0,0,0,0
  1610 000000
                    5417 * .=2E
  1613 0000000000   5418                DEFB          0,0,0,0,0,0,20H,0
  1618 002000
                    5419 * /=2F
  161B 0008102040   5420                DEFB          0,8,10H,20H,40H,80H,0,0
  1620 800000
                    5421
        <1623>      5422 NUMBER_TBL     EQU $
                    5423 * 0=30
  1623 708898A8C8   5424                DEFB          70H,88H,98H,0A8H,0C8H,088H,70H,0
  1628 887000
                    5425 * 1=31
  162B 2060202020   5426                DEFB          20H,60H,20H,20H,20H,20H,70H,0
  1630 207000
                    5427 * 2=32
  1633 7088083040   5428                DEFB          70H,88H,08,30H,40H,80H,0F8H,0
  1638 80F800
                    5429 * 3=33
  163B F808103008   5430                DEFB          0F8H,08,10H,30H,08,88H,70H,0
  1640 887000
                    5431 * 4=34
  1643 10305090F8   5432                DEFB          10H,30H,50H,90H,0F8H,10H,10H,0
  1648 101000
                    5433 * 5=35
  164B F880F00808   5434                DEFB          0F8H,80H,0F0H,08H,08H,88H,70H,0
  1650 887000
                    5435 * 6=36
  1653 384080F088   5436                DEFB          38H,40H,80H,0F0H,88H,88H,70H,0
  1658 887000
                    5437 * 7=37
  165B F808102040   5438                DEFB          0F8H,08H,10H,20H,40H,40H,40H,0
  1660 404000
                    5439 * 8=38
  1663 7088887088   5440                DEFB          70H,88H,88H,70H,88H,88H,70H,0
  1668 887000
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5441 * 9=39
1668 7088887808      5442                DEFB        70H,88H,88H,78H,08H,10H,0E0H,0
1670 10E000
                     5443 * :=3A
1673 0000200020      5444                DEFB        0,0,20H,0,20H,0,0,0
1678 000000
                     5445 * ;=3B
167B 0000200020      5446                DEFB        0,0,20H,0,20H,20H,40H,0
1680 204000
                     5447 * <=3C
1683 1020408040      5448                DEFB        10H,20H,40H,80H,40H,20H,10H,0
1688 201000
                     5449 * ==3D
1688 0000F800F8      5450                DEFB        0,0,0F8H,0,0F8H,0,0,0
1690 000000
                     5451 * >=3E
1693 4020100810      5452                DEFB        40H,20H,10H,08H,10H,20H,40H,0
1698 204000
                     5453 * ?=3F
169B 7088102020      5454                DEFB        70H,88H,10H,20H,20H,0,20H,0
16A0 002000
                     5455 * @=40
16A3 7088A88880      5456                DEFB        70H,88H,0A8H,088H,080H,080H,78H,0
16A8 807800
                     5457
          <16AB>     5458 ASCII_TBL      EQU $
                     5459 * A=41
16AB 20508888F8      5460                DEFB        20H,50H,88H,88H,0F8H,88H,88H,0
16B0 888800
                     5461 * B=42
16B3 F08888F088      5462                DEFB        0F0H,88H,88H,0F0H,88H,88H,0F0H,0
16B8 88F000
                     5463 * C=43
16BB 7088808080      5464                DEFB        70H,88H,80H,80H,80H,88H,70H,0
16C0 887000
                     5465 * D=44
16C3 F088888888      5466                DEFB        0F0H,88H,88H,88H,88H,88H,0F0H,0
16C8 88F000
                     5467 * E=45
16CB F88080F080      5468                DEFB        0F8H,80H,80H,0F0H,080H,80H,0F8H,0
16D0 80F800
                     5469 * F=46
16D3 F88080F080      5470                DEFB        0F8H,80H,80H,0F0H,80H,80H,80H,0
16D8 808000
                     5471 * G=47
16DB 7880808098      5472                DEFB        78H,80H,80H,80H,98H,88H,78H,0
16E0 887800
                     5473 * H=48
16E3 888888F888      5474                DEFB        88H,88H,88H,0F8H,88H,88H,88H,0
16E8 888800
                     5475 * I=49
16EB 7020202020      5476                DEFB        70H,20H,20H,20H,20H,20H,70H,0
16F0 207000
                     5477 * J=4A
16F3 0808080808      5478                DEFB        8,8,8,8,8,88H,70H,0
16F8 887000
                     5479 * K=4B
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   16FB 8890A0C0A0  5480                DEFB        88H,90H,0A0H,0C0H,0A0H,90H,88H,0
   1700 908800
                    5481 * L=4C
   1703 8080808080  5482                DEFB        80H,80H,80H,80H,80H,80H,0F8H,0
   1708 80F800
                    5483 * M=4D
   170B 88D8A8A888  5484                DEFB        88H,0D8H,0A8H,0A8H,88H,88H,88H,0
   1710 888800
                    5485 * N=4E
   1713 8888C8A898  5486                DEFB        88H,88H,0C8H,0A8H,98H,88H,88H,0
   1718 888800
                    5487 * O=4F
   171B 7088888888  5488                DEFB        70H,88H,88H,88H,88H,88H,70H,0
   1720 887000
                    5489 * P=50
   1723 F08888F080  5490                DEFB        0F0H,88H,88H,0F0H,80H,80H,80H,00
   1728 808000
                    5491 * Q=51
   172B 70888888A8  5492                DEFB        70H,88H,88H,88H,0A8H,90H,68H,0
   1730 906800
                    5493 * R=52
   1733 F08888F0A0  5494                DEFB        0F0H,88H,88H,0F0H,0A0H,90H,88H,0
   1738 908800
                    5495 * S=53
   1738 7088807008  5496                DEFB        70H,88H,80H,70H,08H,88H,70H,0
   1740 887000
                    5497 * T=54
   1743 F820202020  5498                DEFB        0F8H,20H,20H,20H,20H,20H,20H,0
   1748 202000
                    5499 * U=55
   1748 8888888888  5500                DEFB        88H,88H,88H,88H,88H,88H,70H,0
   1750 887000
                    5501 * V=56
   1753 8888888888  5502                DEFB        88H,88H,88H,88H,88H,50H,20H,0
   1758 502000
                    5503 * W=57
   175B 888888A8A8  5504                DEFB        88H,88H,88H,0A8H,0A8H,0D8H,88H,0
   1760 D88800
                    5505 * X=58
   1763 8888502050  5506                DEFB        88H,88H,50H,20H,50H,88H,88H,0
   1768 888800
                    5507 * Y=59
   1768 8888502020  5508                DEFB        88H,88H,50H,20H,20H,20H,20H,0
   1770 202000
                    5509 * Z=5A
   1773 F808102040  5510                DEFB        0F8H,08H,10H,20H,40H,80H,0F8H,0
   1778 80F800
                    5511 * [=5B
   177B F8C0C0C0C0  5512                DEFB        0F8H,0C0H,0C0H,0C0H,0C0H,0C0H,0F8H,0
   1780 C0F800
                    5513 * \=5C
   1783 0080402010  5514                DEFB        0,80H,40H,20H,10H,08H,0,0
   1788 080000
                    5515 * ]=5D
   1788 F818181818  5516                DEFB        0F8H,18H,18H,18H,18H,18H,0F8H,0
   1790 18F800
                    5517 * ^=5E
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1793 0000205088  5518                DEFB        0,0,20H,50H,88H,0,0,0
    1798 000000
                     5519 * _=5F
    1798 0000000000  5520                DEFB        0,0,0,0,0,0,0,0F8H
    17A0 0000F8
                     5521 * `=60
    17A3 4020100000  5522                DEFB        40H,20H,10H,0,0,0,0,0
    17A8 000000
                     5523 * a=61
    17AB 00007088F8  5524                DEFB        0,0,70H,88H,0F8H,88H,88H,0
    17B0 888800
                     5525 * b=62
    17B3 0000F04870  5526                DEFB        0,0,0F0H,48H,70H,48H,0F0H,0
    17B8 48F000
                     5527 * c=63
    17BB 0000788080  5528                DEFB        0,0,78H,80H,80H,80H,78H,0
    17C0 807800
                     5529 * d=64
    17C3 0000F04848  5530                DEFB        0,0,0F0H,048H,048H,048H,0F0H,0
    17C8 48F000
                     5531 * e=65
    17CB 0000F080E0  5532                DEFB        0,0,0F0H,080H,0E0H,80H,0F0H,0
    17D0 80F000
                     5533 * f=66
    17D3 0000F080E0  5534                DEFB        0,0,0F0H,080H,0E0H,080H,80H,0
    17D8 808000
                     5535 * g=67
    17DB 0000788088  5536                DEFB        0,0,78H,80H,0B8H,88H,70H,0
    17E0 887000
                     5537 * h=68
    17E3 00008888F8  5538                DEFB        0,0,88H,88H,0F8H,88H,88H,0
    17E8 888800
                     5539 * i=69
    17EB 0000F82020  5540                DEFB        0,0,0F8H,20H,20H,20H,0F8H,0
    17F0 20F800
                     5541 * j=6A
    17F3 0000702020  5542                DEFB        0,0,70H,20H,20H,0A0H,0E0H,0
    17F8 A0E000
                     5543 * k=6B
    17FB 000090A0C0  5544                DEFB        0,0,90H,0A0H,0C0H,0A0H,90H,0
    1800 A09000
                     5545 * l=6C
    1803 0000808080  5546                DEFB        0,0,80H,80H,80H,80H,0F8H,0
    1808 80F800
                     5547 * m=6D
    1808 000088D8A8  5548                DEFB        0,0,88H,0D8H,0A8H,88H,88H,0
    1810 888800
                     5549 * n=6E
    1813 000088C8A8  5550                DEFB        0,0,88H,0C8H,0A8H,98H,88H,0
    1818 988800
                     5551 * o=6F
    181B 0000F88888  5552                DEFB        0,0,0F8H,088H,88H,88H,0F8H,0
    1820 88F800
                     5553 * p=70
    1823 0000F088F0  5554                DEFB        0,0,0F0H,88H,0F0H,80H,80H,0
    1828 808000
                     5555 * q=71
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    182B 0000F888A8  5556                DEFB            0,0,0F8H,88H,0A8H,90H,0E0H,0
    1830 90E000
                     5557 * r=72
    1833 0000F888F8  5558                DEFB            0,0,0F8H,88H,0F8H,0A0H,90H,0
    1838 A09000
                     5559 * s=73
    183B 0000788070  5560                DEFB            0,0,78H,80H,70H,08H,0F0H,0
    1840 08F000
                     5561 * t=74
    1843 0000F82020  5562                DEFB            0,0,0F8H,20H,20H,20H,20H,0
    1848 202000
                     5563 * u=75
    184B 0000888888  5564                DEFB            0,0,88H,88H,88H,88H,70H,00
    1850 887000
                     5565 * v=76
    1853 0000888890  5566                DEFB            0,0,88H,88H,90H,0A0H,40H,0
    1858 A04000
                     5567 * w=77
    185B 00008888A8  5568                DEFB            0,0,88H,88H,0A8H,0D8H,88H,00
    1860 D88800
                     5569 * x=78
    1863 0000886020  5570                DEFB            0,0,88H,60H,20H,60H,88H,0
    1868 608800
                     5571 * y=79
    1868 0000885020  5572                DEFB            0,0,88H,50H,20H,20H,20H,0
    1870 202000
                     5573 * z=7A
    1873 0000F81020  5574                DEFB            0,0,0F8H,10H,20H,40H,0F8H,0
    1878 40F800
                     5575 * {=7B
    187B 384020C020  5576                DEFB            38H,40H,20H,0C0H,20H,40H,38H,0
    1880 403800
                     5577 * >=7C
    1883 4020100810  5578                DEFB            40H,20H,10H,08H,10H,20H,40H,0
    1888 204000
                     5579 * }=7D
    1888 E010201820  5580                DEFB            0E0H,10H,20H,18H,20H,10H,0E0H,0
    1890 10E000
                     5581 * ~=7E
    1893 40A8100000  5582                DEFB            40H,0A8H,10H,0,0,0,0,0
    1898 000000
                     5583 *  ■=7F
    189B A850A850A8  5584                DEFB            0A8H,50H,0A8H,50H,0A8H,50H,0A8H,0
    18A0 50A800
                     5585
          <18A3>     5586 OBJ_TABLE       EQU $
    18A3 01020E0F08  5587                DEFB 1,2,14,15,8,9,18,19
    18A8 091213
    18AB 03040E0F05  5588                DEFB 3,4,14,15,5,20,0,0
    18B0 140000
    18B3 050010110A  5589                DEFB 5,0,16,17,10,11,21,22
    18B8 0B1516
    18BB 0607101105  5590                DEFB 6,7,16,17,5,20,0,0
    18C0 140000
    18C3 01020E0F03  5591                DEFB 1,2,14,15,3,4,14,15
    18C8 040E0F
    18CB 03040E0F0C  5592                DEFB 3,4,14,15,12,13,23,24
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    18D0 001718
    18D3 FF          5593                HEX  FF                          ;END OF TABLE INDICATOR
                     5594
                     5595 ************************************************************************
                     5596 *                                                                      *
                     5597 *                          SUBROUTINES                                 *
                     5598 *                                                                      *
                     5599 ************************************************************************
                     5600
                     5601 ************************** OS SUBROUTINES ************************
                     5602
                     5603 * FILL_VRAM_  WRITES TO VRAM ADDRESS POINTED TO BY HL THE VALUE IN A
                     5604 *             DE TIMES.
                     5605
                     5606 *          VRAM STARTING ADDRESS IN HL
                     5607 *          NO OF BYTES IN DE
                     5608 *          VALUE TO BE WRITTEN IN A
                     5609
    18D4 4F          5610 FILL_VRAM_  LD   C,A
    18D5 7D          5611             LD   A,L
    18D6 D3BF        5612             OUT  [MODE_1_PORT],A
    18D8 7C          5613             LD   A,H
    18D9 F640        5614             OR   40H
    18DB D3BF        5615             OUT  [MODE_1_PORT],A
    18DD 79          5616 FILL        LD   A,C
    18DE D3BE        5617             OUT  [MODE_0_PORT],A
    18E0 1B          5618             DEC  DE
    18E1 7A          5619             LD   A,D
    18E2 B3          5620             OR   E
    18E3 20F8        5621             JR   NZ,FILL
    18E5 CD1FDC      5622             CALL READ_REGISTER
    18E8 C9          5623             RET
                     5624
                     5625
                     5626 * MODE_1_  SETS UP GRAPHICS MODE 1 WITH VRAM ADDRESSES AS IN THE TABLE
                     5627 *          BELOW AND EXITS WITH THE VIDEO BLANKED AND A BLACK BACKGROUND.
                     5628
                     5629 *                              VDP MEMORY MAP
                     5630 *                   3800H-3FFFH  SPRITE GENERATOR TABLE
                     5631 *                   2000H-37FFH  PATTERN COLOR TABLE
                     5632 *                   1B00H-1B7FH  SPRITE ATTRIBUTE TABLE
                     5633 *                   1800H-1AFFH  PATTERN NAME TABLE
                     5634 *                   0000H-17FFH  PATTERN GENERATOR TABLE
                     5635
    18E9 0600        5636 MODE_1_             LD B,0
    18EB 0E00        5637                     LD C,0
    18ED CD1FD9      5638                     CALL WRITE_REGISTER
                     5639
    18F0 0601        5640                     LD B,1
    18F2 0E80        5641                     LD C,10000000B
    18F4 CD1FD9      5642                     CALL WRITE_REGISTER
                     5643
                     5644 * SET UP TABLE ADDRESSES IN VRAM
                     5645
                     5646 * PATTERN NAME TABLE
    18F7 3E02        5647                     LD A,2
    18F9 211800      5648                     LD HL,1800H
```

```
ATION OBJECT CODE LINE     SOURCE LINE

 18FC CD1FB8      5649                CALL INIT_TABLE
                  5650
                  5651 * PATTERN COLOR TABLE
 18FF 3E04        5652                LD A,4
 1901 212000      5653                LD HL,2000H
 1904 CD1FB8      5654                CALL INIT_TABLE
                  5655
                  5656 * PATTERN GENERATOR TABLE
 1907 3E03        5657                LD A,3
 1909 210000      5658                LD HL,0
 190C CD1FB8      5659                CALL INIT_TABLE
                  5660
                  5661 * SPRITE ATTRIBUTE TABLE
 190F 3E00        5662                LD A,0
 1911 211B00      5663                LD HL,1B00H
 1914 CD1FB8      5664                CALL INIT_TABLE
                  5665
                  5666 * SPRITE GENERATOR TABLE
 1917 3E01        5667                LD A,1
 1919 213800      5668                LD HL,3800H
 191C CD1FB8      5669                CALL INIT_TABLE
                  5670
                  5671 *  SET UP BLACK BACKGROUND
 191F 0607        5672                LD B,7
 1921 0E00        5673                LD C,0
 1923 CD1FD9      5674                CALL WRITE_REGISTER
 1926 C9          5675                RET
                  5676
                  5677 *  LOAD_ASCII_ WRITES OUT ASCII CHARACTER GENERATORS TO THE PATTERN
                  5678 *              GENERATOR TABLE.  INIT_TABLE MUST BE USED TO SET UP
                  5679 *              THE TABLE ADDRESS.
                  5680
 1927 211588      5681 LOAD_ASCII_    LD HL,ASC_TABLE             ;LOCATION OF GENERATORS
 192A 11001D      5682                LD DE,1DH                   ;OFFSET TO PLACE ASC_TABLE
 192D FD210060    5683                LD IY,96                    ;IN THE CORRECT LOCATION
 1931 3E03        5684                LD A,3
 1933 CD1FBE      5685                CALL PUT_VRAM
                  5686
                  5687 * WRITE OUT A BLANK PATTERN FOR ASCII_0
 1936 2115A3      5688                LD HL,SPACE
 1939 110000      5689                LD DE,0
 193C FD210001    5690                LD IY,1
 1940 3E03        5691                LD A,3
 1942 CD1FBE      5692                CALL PUT_VRAM
 1945 C9          5693                RET
                  5694
                  5695 ********************** LOCAL SUBROUTINES **********************
                  5696
 1946 010000      5697 PARSE          LD BC,0             ;FROM HL INCREMENT BC UNTIL
 1949 7E          5698 P_LOOP         LD A,[HL]           ;[HL] = "/"
 194A FE2F        5699                CP "/"
 194C C8          5700                RET Z
 194D 23          5701                INC HL
 194E 03          5702                INC BC
 194F 18F8        5703                JR P_LOOP
                  5704
 1951 C5          5705 CENTER_PRT     PUSH BC             ;BC=LEN$
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1952 FDE1       5706                POP IY              ;IY= #ITEMS TO BE TRANSFERED IN PUT_VRAM
    1954 3E20       5707                LD A,32             ;DE= LOCATION OF START OF STRING
    1956 99         5708                SBC A,C             ; A=32-C
    1957 1F         5709                RRA                 ; DIV 2
    1958 0600       5710                LD B,0
    195A 4F         5711                LD C,A
    195B 09         5712                ADD HL,BC
    195C 44         5713                LD B,H
    1950 4D         5714                LD C,L
    195E 62         5715                LD H,D
    195F 6B         5716                LD L,E
    1960 50         5717                LD D,B
    1961 59         5718                LD E,C
    1962 3E02       5719                LD A,2
    1964 CD1FBE     5720                CALL PUT_VRAM
    1967 C9         5721                RET
                    5722
    1968 211700     5723 DELAY_10       LD HL,1700H
    1968 1100FF     5724 TIMER_1        LD DE,255
    196E 1B         5725 TIMER_2        DEC DE
    196F 7A         5726                LD A,D
    1970 B3         5727                OR E
    1971 20FB       5728                JR NZ,TIMER_2
    1973 2B         5729                DEC HL
    1974 7C         5730                LD A,H
    1975 B5         5731                OR L
    1976 20F3       5732                JR NZ,TIMER_1
    1978 C9         5733                RET
                    5734  PROG
```

```
OCATION OBJECT CODE LINE     SOURCE LINE

                    5736
                    5737 ************************** EXTERNAL SYMBOLS **************************
                    5738
                    5739 **********************************************************************
                    5740
                    5741 * EXTERNAL ROUTINES FROM OS
                    5742
                    5743                ;EXT INIT_TABLE
                    5744                ;EXT PUT_VRAM
                    5745                ;EXT WRITE_REGISTER
                    5746                ;EXT WRITE_VRAM
                    5747                ;EXT VRAM_ADDR_TABLE
                    5748                ;EXT SPRITENAMETBL
                    5749                ;EXT SPRITEGENTBL
                    5750                ;EXT PATTRNNAMETBL
                    5751                ;EXT PATTRNGENTBL
                    5752                ;EXT COLORTABLE
                    5753                ;EXT LOAD_ASCII
                    5754                ;EXT FILL_VRAM
                    5755                ;EXT MODE_1
                    5756
                    5757 ************************** DEFINITIONS **************************
                    5758
                    5759 ***************************** EXPORTS *****************************
                    5760
                    5761                 GLB GAME_OPT_
                    5762
                    5763 ********************* DISPLAY GAME OPTION SCREEN *********************
                    5764
                    5765 *  GAME_OPT_  DISPLAYS THE GAME OPTION SCREEN WITH WHITE LETTERS ON A
                    5766 *             BLUE BACKGROUND.  VDP IS LEFT IN MODE 1 WITH THE VRAM
                    5767 *             MEMORY MAP AS FOLLOWS.
                    5768 *
                    5769 ;                          VDP MEMORY MAP
                    5770 ;               3800H-3FFFH  SPRITE GENERATOR TABLE
                    5771 ;               2000H-37FFH  PATTERN COLOR TABLE
                    5772 ;               1B00H-1B7FH  SPRITE ATTRIBUTE TABLE
                    5773 ;               1800H-1AFFH  PATTERN NAME TABLE
                    5774 ;               0000H-17FFH  PATTERN GENERATOR TABLE
                    5775
                    5776                 PROG
1979 210000         5777 GAME_OPT_       LD HL,0                        ;ZERO VRAM
197C 114000         5778                 LD DE,16384
197F 3E00           5779                 LD A,0
1981 CD1F82         5780                 CALL FILL_VRAM
                    5781
                    5782 *  SET UP VDP WITH MODE_1
1984 CD1F85         5783                 CALL MODE_1
                    5784
                    5785 *  SET UP BACKGROUND COLOR
1987 060F           5786                 LD B,0FH
1989 0E04           5787                 LD C,4
198B CD1FD9         5788                 CALL WRITE_REGISTER
                    5789
                    5790 ******************* WRITE OUT PATTERN GEN TABLE ******************
                    5791
198E CD1F7F         5792                 CALL LOAD_ASCII
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5793
                     5794 ****************** WRITE OUT PATTERN_NAME_TABLE ****************
                     5795
                     5796 *  WRITE OUT PATTERN_NAME_TABLE
 1991 211A7C         5797                LD HL,LINE_1
 1994 110025         5798                LD DE,37
 1997 FD210016       5799                LD IY,22
 1998 3E02           5800                LD A,2
 199D CD1FBE         5801                CALL PUT_VRAM
                     5802
 19A0 211A92         5803                LD HL,LINE_2
 19A3 110065         5804                LD DE,101
 19A6 FD210017       5805                LD IY,23
 19AA 3E02           5806                LD A,2
 19AC CD1FBE         5807                CALL PUT_VRAM
                     5808
 19AF 1100C5         5809                LD DE,197
 19B2 CD1ACA         5810                CALL WRITE_L3
                     5811
 19B5 110105         5812                LD DE,261
 19B8 CD1ACA         5813                CALL WRITE_L3
                     5814
 19BB 110145         5815                LD DE,325
 19BE CD1ACA         5816                CALL WRITE_L3
                     5817
 19C1 110185         5818                LD DE,389
 19C4 CD1ACA         5819                CALL WRITE_L3
                     5820
 19C7 1101E5         5821                LD DE,485
 19CA CD1ACA         5822                CALL WRITE_L3
                     5823
 19CD 110225         5824                LD DE,549
 19D0 CD1ACA         5825                CALL WRITE_L3
                     5826
 19D3 110265         5827                LD DE,613
 19D6 CD1ACA         5828                CALL WRITE_L3
                     5829
 19D9 1102A5         5830                LD DE,677
 19DC CD1ACA         5831                CALL WRITE_L3
                     5832
 19DF 110105         5833                LD DE,261
 19E2 CD1AD7         5834                CALL WRITE_L4
                     5835
 19E5 110145         5836                LD DE,325
 19E8 CD1ADC         5837                CALL WRITE_L5
                     5838
 19EB 110185         5839                LD DE,389
 19EE CD1AE1         5840                CALL WRITE_L6
                     5841
 19F1 211AC2         5842                LD HL,LINE_7
 19F4 1101E5         5843                LD DE,485
 19F7 CD1AE4         5844                CALL WRITE_CHAR
                     5845
 19FA 211AC3         5846                LD HL,LINE_8
 19FD 110225         5847                LD DE,549
 1A00 CD1AE4         5848                CALL WRITE_CHAR
                     5849
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1A03 211AC4      5850                 LD HL,LINE_9
    1A06 110265      5851                 LD DE,613
    1A09 CD1AE4      5852                 CALL WRITE_CHAR
                     5853
    1A0C 211AC5      5854                 LD HL,LINE_10
    1A0F 1102A5      5855                 LD DE,677
    1A12 CD1AE4      5856                 CALL WRITE_CHAR
                     5857
    1A15 11010F      5858                 LD DE,271
    1A18 CD1AD7      5859                 CALL WRITE_L4
                     5860
    1A1B 11014F      5861                 LD DE,335
    1A1E CD1ADC      5862                 CALL WRITE_L5
                     5863
    1A21 11018F      5864                 LD DE,399
    1A24 CD1AE1      5865                 CALL WRITE_L6
                     5866
    1A27 1101F1      5867                 LD DE,497
    1A2A CD1AEE      5868                 CALL WRITE_L11
                     5869
    1A2D 110231      5870                 LD DE,561
    1A30 CD1AEE      5871                 CALL WRITE_L11
                     5872
    1A33 110271      5873                 LD DE,625
    1A36 CD1AEE      5874                 CALL WRITE_L11
                     5875
    1A39 1102B1      5876                 LD DE,689
    1A3C CD1AEE      5877                 CALL WRITE_L11
                     5878
    1A3F 11022F      5879                 LD DE,559
    1A42 CD1AD7      5880                 CALL WRITE_L4
                     5881
    1A45 11026F      5882                 LD DE,623
    1A48 CD1ADC      5883                 CALL WRITE_L5
                     5884
    1A4B 1102AF      5885                 LD DE,687
    1A4E CD1AE1      5886                 CALL WRITE_L6
                     5887
    1A51 1101FB      5888                 LD DE,507
    1A54 CD1AFB      5889                 CALL WRITE_L12
                     5890
    1A57 11023B      5891                 LD DE,571
    1A5A CD1AFB      5892                 CALL WRITE_L12
                     5893
    1A5D 11027B      5894                 LD DE,635
    1A60 CD1AFB      5895                 CALL WRITE_L12
                     5896
    1A63 1102BB      5897                 LD DE,699
    1A66 CD1AFB      5898                 CALL WRITE_L12
                     5899
                     5900 ****************** WRITE OUT COLOR_NAME_TABLE ****************
                     5901
    1A69 2A73FA      5902                 LD HL,[COLORTABLE]
    1A6C 110020      5903                 LD DE,32
    1A6F 3EF4        5904                 LD A,0F4H
    1A71 CD1F82      5905                 CALL FILL_VRAM
                     5906
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5907 ************************** ENABLE DISPLAY **********************
                     5908
                     5909 * ENABLE DISPLAY
    1A74 0601        5910                 LD B,1
    1A76 0EC0        5911                 LD C,11000000B
    1A78 CD1FD9      5912                 CALL WRITE_REGISTER
    1A7B C9          5913                 RET
                     5914
                     5915 ***************************************************************************
                     5916 *                                                                         *
                     5917 *                            DATA TABLES                                  *
                     5918 *                                                                         *
                     5919 ***************************************************************************
                     5920
                     5921 ********************** PATTERN_NAME_TABLE **********************
    1A7C 544F205345  5922 LINE_1 DEFB "TO SELECT GAME OPTION,"
    1A81 4C45435420
    1A86 47414D4520
    1A88 4F5054494F
    1A90 4E2C
    1A92 5052455353  5923 LINE_2 DEFB "PRESS BUTTON ON KEYPAD."
    1A97 2042555454
    1A9C 4F4E204F4E
    1AA1 2048455950
    1AA6 41442E
    1AA9 3120302053  5924 LINE_3 DEFB "1 = SKILL 1/ONE PLAYER"
    1AAE 4B494C4C20
    1AB3 312F4F4E45
    1AB8 20504C4159
    1ABD 4552
    1ABF 32          5925 LINE_4 DEFB "2"
    1AC0 33          5926 LINE_5 DEFB "3"
    1AC1 34          5927 LINE_6 DEFB "4"
    1AC2 35          5928 LINE_7 DEFB "5"
    1AC3 36          5929 LINE_8 DEFB "6"
    1AC4 37          5930 LINE_9 DEFB "7"
    1AC5 38          5931 LINE_10 DEFB "8"
    1AC6 54574F      5932 LINE_11 DEFB "TWO"
    1AC9 53          5933 LINE_12 DEFB "S"
                     5934
                     5935 ***************************************************************************
                     5936 *                                                                         *
                     5937 *                         LOCAL SUBROUTINES                               *
                     5938 *                                                                         *
                     5939 ***************************************************************************
                     5940
    1ACA 211AA9      5941 WRITE_L3        LD HL,LINE_3
    1ACD FD210016    5942                 LD IY,22
    1AD1 3E02        5943                 LD A,2
    1AD3 CD1FBE      5944                 CALL PUT_VRAM
    1AD6 C9          5945                 RET
                     5946
    1AD7 211ABF      5947 WRITE_L4        LD HL,LINE_4
    1ADA 1808        5948                 JR WRITE_CHAR
                     5949
    1ADC 211AC0      5950 WRITE_L5        LD HL,LINE_5
    1ADF 1803        5951                 JR WRITE_CHAR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     5952
    1AE1 211AC1      5953 WRITE_L6        LD HL,LINE_6
    1AE4 FD210001    5954 WRITE_CHAR      LD IY,1
    1AE8 3E02        5955                 LD A,2
    1AEA CD1FBE      5956                 CALL PUT_VRAM
    1AED C9          5957                 RET
                     5958
    1AEE 211AC6      5959 WRITE_L11       LD HL,LINE_11
    1AF1 FD210003    5960                 LD IY,3
    1AF5 3E02        5961                 LD A,2
    1AF7 CD1FBE      5962                 CALL PUT_VRAM
    1AFA C9          5963                 RET
                     5964
    1AFB 211AC9      5965 WRITE_L12       LD HL,LINE_12
    1AFE FD210001    5966                 LD IY,1
    1B02 3E02        5967                 LD A,2
    1B04 CD1FBE      5968                 CALL PUT_VRAM
    1B07 C9          5969                 RET
                     5970
                     5971  PROG
```

```
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
 1B08 00020001       5987 INIT_TABLE_P    DEFW            2,1,2
 1B0C 0002
                     5988 * THIS IS THE PARAMETER DESCRIPTOR FOR INIT_TABLEQ
                     5989
 1B0E                5990 INIT_TABLEQ
 1B0E 011B08         5991                 LD              BC,INIT_TABLE_P
 1B11 1173BA         5992                 LD              DE,PARAM_AREA
 1B14 CD0098         5993                 CALL            PARAM_
 1B17 3A73BA         5994                 LD              A,[PARAM_AREA]
 1B1A 2A73BB         5995                 LD              HL,[PARAM_AREA+1]
                     5996
 1B1D                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
 1B1D 4F             6007           LD    C,A
 1B1E 0600           6008           LD    B,0              ;USE TABLE CODE AS INDEX
 1B20 DD2173F2       6009           LD    IX,VRAM_ADDR_TABLE
 1B24 DD09           6010           ADD   IX,BC
 1B26 DD09           6011           ADD   IX,BC
 1B28 DD7500         6012           LD    [IX+0],L         ;SAVE VRAM ADDRESS IN TABLE
 1B2B DD7401         6013           LD    [IX+1],H
                     6014                                  ;CHECK VDP GRAPHICS MODE
 1B2E 3A73C3         6015           LD    A,[VDP_MODE_WORD]
 1B31 CB4F           6016           BIT   1,A
 1B33 2827           6017           JR    Z,INIT_TABLE80
                     6018           ;**   GRAPHICS MODE 2
 1B35 79             6019           LD    A,C              ;GET TABLE CODE
 1B36 FE03           6020           CP    3                ;CHECK IF GENERATOR TABLE
 1B38 2806           6021           JR    Z,CASE_OF_GEN
 1B3A FE04           6022           CP    4                ;CHECK IF COLOR TABLE
 1B3C 2810           6023           JR    Z,CASE_OF_COLOR
 1B3E 181C           6024           JR    INIT_TABLE80
 1B40                6025 CASE_OF_GEN
 1B40 0604           6026           LD    B,4              ;VDP REGISTER TO WRITE
 1B42 7D             6027           LD    A,L              ;CHECK FOR WHICH 8K BOUNDARY IN 16K VRAM
 1B43 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 ED5B738B    6082                 LD              DE,[PARAM_AREA+1]
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1B9C FD2A73BF    6083                LD              IY,[PARAM_AREA+5]
    1BA0 2A73BD      6084                LD              HL,[PARAM_AREA+3]
                     6085
                     6086
    1BA3             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
    1BA3 CD1BAA      6097            CALL SET_COUNT
    1BA6 CD1D3E      6098            CALL VRAM_READ
    1BA9 C9          6099            RET
                     6100
                     6101
    1BAA             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
    1BAA FD2273FE    6113            LD   [SAVED_COUNT],IY               ;SAVE COUNT
    1BAE DD2173F2    6114            LD   IX,VRAM_ADDR_TABLE             ;POINTER TO SAVED VRAM ADDRESSES
    1BB2 4F          6115            LD   C,A             ;SAVE TABLE CODE
    1BB3 0600        6116            LD   B,0             ;BC USED AS INDEX
    1BB5 FE04        6117            CP   4                              ;CHECK FOR CASE OF COLOR TABLE
    1BB7 2007        6118            JR   NZ,SET_COUNT10
                     6119            ;**  COLOR TABLE, CHECK IF MODE 1
    1BB9 3A73C3      6120            LD   A,[VDP_MODE_WORD]              ;CHECK FOR VDP GRAPHICS MODE
    1BBC CB4F        6121            BIT  1,A
    1BBE 282C        6122            JR   Z,SET_COUNT20
                     6123            ;**  NOT MODE 1, ADJUST ITEM COUNT AND INDEX
    1BC0             6124 SET_COUNT10
    1BC0 FD211BFF    6125            LD   IY,SHIFT_CT     ;GET ITEM COUNT CONVERSION FACTORS
    1BC4 FD09        6126            ADD  IY,BC
    1BC6 FD7E00      6127            LD   A,[IY+0]        ;SHIFT COUNT FOR MULTIPLICATION
    1BC9 FE00        6128            CP   0
    1BCB 281F        6129            JR   Z,SET_COUNT20
    1BCD             6130 ADJUST_INDEX                   ;MULTIPLY ITEM_INDEX TO GET BYTE_INDEX
    1BCD CB23        6131            SLA  E
    1BCF CB12        6132            RL   D
    1BD1 3D          6133            DEC  A
    1BD2 20F9        6134            JR   NZ,ADJUST_INDEX
    1BD4             6135 END_ADJ_INDEX
    1BD4 C5          6136            PUSH BC              ;SAVE TABLE_CODE/INDEX
    1BD5 ED4B73FE    6137            LD   BC,[SAVED_COUNT]
    1BD9 FD7E00      6138            LD   A,[IY+0]        ;SHIFT COUNT FOR MULTIPLICATION
    1BDC 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 ED5B738B     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]
  1C68             6255 INIT_SPR10
  1C68 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
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     6352
                     6353 * The video drivers provide a standard protocol for low-level comunication
                     6354 * with the 9918/28 VDP. There are four basic driver routines which between
                     6355 * them allow the programmer to write a value to a VDP register, read the
                     6356 * VDP status register, write a RAM or ROM buffer to a specified address
                     6357 * in VRAM, and read a RAM buffer from a specified address in VRAM.
                     6358 *
                     6359 * The four routines outlined above are:
                     6360 *
                     6361 *   Procedure Reg_Write
                     6362 *
                     6363 *     Reg_Write takes a VDP register number (0..7) in the B register
                     6364 *     and a byte value to be written to it in the C register. It writes
                     6365 *     the value to the given VDP register and returns.
                     6366 *
                     6367 *     If the specified register is one of the VDP mode control registers,
                     6368 *     ie. 0 or 1, then Reg_Write also writes the given value to the
                     6369 *     corresponding half of the VDP_Mode_Word in RAM. All mode dependant
                     6370 *     decisions made by the operating system are made by referencing the
                     6371 *     contents of this word. Thus it is important for the cartridge
                     6372 *     programmer to maintain it should he/she choose not to use
                     6373 *     Reg_Write in accessing the VDP registers.
                     6374 *
                     6375 *     In addition to the BC pair, Reg_Write also makes use of AF.
                     6376 *
                     6377 *   Function Reg_Read
                     6378 *
                     6379 *     Reg_Read reads the VDP status register and returns its contents
                     6380 *     in the A register.
                     6381 *
                     6382 *     It uses no other registers.
                     6383 *
                     6384 *     NOTE ****** While this routine has no side effects with respect to
                     6385 *                 the CPU, it should be used with caution since reads
                     6386 *                 to the status register have the effect of resetting the
                     6387 *                 VDP interrupt flag and may cause field interrupts to be
                     6388 *                 missed.
                     6389 *
                     6390 *   Procedure Vram_Write
                     6391 *
                     6392 *     Vram_Write takes a pointer to the beginning of a data buffer in the
                     6393 *     HL pair, the VRAM destination address in the DE pair, and a byte
                     6394 *     count in the BC pair.
                     6395 *
                     6396 *     It writes the specified number of bytes from the buffer to VRAM
                     6397 *     starting at the destination address.
                     6398 *
                     6399 *     The AF,BC,DE, and HL register pairs are all affected.
                     6400 *
                     6401 *     NOTE ****** This procedure is not re-entrant.
                     6402 *
                     6403 *   Procedure Vram_Read
                     6404 *
                     6405 *     Vram_Read takes a pointer to the beginning of a data buffer in the
                     6406 *     HL pair, the VRAM source address in the DE pair, and a byte count
                     6407 *     count in the BC pair.
                     6408 *
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    6409 *      It reads the specified number of bytes into the buffer from VRAM
                    6410 *      starting at the source address.
                    6411 *
                    6412 *      The AF,BC,DE, and HL register pairs are all affected.
                    6413 *
                    6414 *      NOTE ****** This procedure is not re-entrant.
                    6415 *
                    6416 *
                    6417 *
                    6418 * For each of the routines listed above, there is an additional entry
                    6419 * point which allows the routine to be called using the standard pascal
                    6420 * 64000 parameter passing protocol and passing the parameters through
                    6421 * a common data area into the registers. It should be noted that use of
                    6422 * these routines in this fashion may cause problems in an interrupt
                    6423 * driven environment. They should therefor be used with care. If the
                    6424 * name of a given routine is Name, the name of the additional entry point
                    6425 * is NameQ for the actual routine and NameP when called through the
                    6426 * jump table in OS ROM. "Q" entry points destroy all registers.
                    6427 ********************************************************************************
                    6428
                    6429 ************************** DICTIONARY *********************************
                    6430
          <00BE>    6431 DATA_PORT       EQU             0BEH
          <00BF>    6432 CTRL_PORT       EQU             0BFH
                    6433                  GLB             DATA_PORT,CTRL_PORT
                    6434 * DATA_PORT AND CTRL_PORT ARE THE PORTS THROUGH WHICH THE CPU
                    6435 * COMMUNICATES WITH THE 9928.
                    6436
                    6437
                    6438 ;                 COMN
                    6439 ;PARAM_AREA       DEFS            6
                    6440 * THIS THE PARAMETER PASSING AREA THAT IS HELD IN COMMON WITH ALL OTHER
                    6441 * PASCAL ENTRY POINTS TO OS ROUTINES.
                    6442
                    6443 ********************************************************************************
                    6444
                    6445 ****************** PROCEDURES AND FUNCTIONS ***************************
                    6446                   PROG
                    6447
                    6448 ;EXT              PARAM_
                    6449 * PARAM_ IS THE PASCAL PARAMETER PASSING ROUTINE.
                    6450
                    6451 * PROCEDURE  REG_WRITE (REGISTER:BYTE;VALUE:BYTE)
                    6452
                    6453 * REGISTER IS PASSED IN THE B REGISTER.
                    6454 * VALUE IS PASSED IN THE C REGISTER.
                    6455 * DESTROYS: A
                    6456 ************ NOTE
                    6457 *
                    6458 *                 **** THIS ROUTINE MAY SIDE EFFECT THE ****
                    6459 *                 **** VDP MODE WORD                    ****
                    6460
                    6461                  ;EXT              VDP_MODE_WORD
                    6462
                    6463                   GLB REG_WRITE_P
 1CB6 00020001      6464 REG_WRITE_P       DEFW              2,1,1
 1CBA 0001
```

```
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 1173BA       6471                    LD              DE,PARAM_AREA
  1CC2 CD0098       6472                    CALL            PARAM_
  1CC5 2A73BA       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 1173BA  6579                 LD              DE,PARAM_AREA
     1D30 CD0098  6580                 CALL            PARAM_
     1D33 2A73BA  6581                 LD              HL,[PARAM_AREA]
     1D36 ED5B73BC 6582                LD              DE,[PARAM_AREA+2]
     1D3A ED4B73BE 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 D3BF    6590                 OUT             [CTRL_PORT],A
                  6591
                  6592 * HIGH BYTE OF SRCE =; CTRL_PORT
     1D41 7A      6593                 LD              A,D
     1D42 D3BF    6594                 OUT             [CTRL_PORT],A
                  6595
                  6596 * DATA =; DATA_PORT
     1D44 C5      6597                 PUSH            BC
     1D45 D1      6598                 POP             DE
     1D46 0EBE    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 C21D49  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 DBBF    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 CD1BA3      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 RFLCT_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
                    6858
```

```
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
```

```
OCATION 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 CD1EA8      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 D9          6971                EXX
   1E44 E5          6972                PUSH            HL
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1E45 D9          6973                 EXX
    1E46 D1          6974                 POP            DE
    1E47 2A8006      6975                 LD             HL,[WORK_BUFFER]
    1E4A 010008      6976                 LD             BC,8
    1E4D 09          6977                 ADD            HL,BC
    1E4E FD210004    6978                 LD             IY,4
    1E52 CD1C27      6979                 CALL           PUT_VRAM_
                     6980
                     6981 *   END_IF
    1E55             6982 END_IF_4_GRAPHICS
                     6983
                     6984 *   DESTINATION := DESTINATION + 4
    1E55 D9          6985                 EXX
    1E56 23          6986                 INC            HL
    1E57 23          6987                 INC            HL
    1E58 23          6988                 INC            HL
    1E59 23          6989                 INC            HL
                     6990
                     6991 * END
    1E5A C31D8C      6992                 JP             RETURN_HERE
                     6993
                     6994
                     6995
                     6996
    1E5D             6997 COLOR_TEST                     ; TESTS WHETHER PATTERN GENERATORS ARE
                     6998                                ; BEING MANIPULATED AND WHETHER THE
                     6999                                ; GRAPHICS MODE IS 2. IF SO THE ABOVE
                     7000                                ; ROUTINES NEED TO DEAL WITH THE COLOR
                     7001                                ; GENERATORS THAT CORRESPOND TO THE
                     7002                                ; PATTERN GENERATORS THEY ARE OPERATING
                     7003                                ; ON.
                     7004
                     7005                                ; NOT INPUTS, RETURNS WITH TRUE (1) IN
                     7006                                ; A IF CONDITION IS TRUE, FALSE (0) IF
                     7007                                ; NOT.
                     7008
                     7009
                     7010 * BEGIN COLOR_TEST
                     7011
                     7012 * CHECK TABLE CODE IN A'
    1E5D 08          7013                 EX             AF,AF'
    1E5E F5          7014                 PUSH           AF
    1E5F 08          7015                 EX             AF,AF'
    1E60 F1          7016                 POP            AF
    1E61 FE03        7017                 CP             PATTERN_GEN
    1E63 200A        7018                 JR             NZ,EXIT_FALSE
                     7019
                     7020 * CHECK MODE
    1E65 2173C3      7021                 LD             HL,VDP_MODE_WORD
    1E68 CB4E        7022                 BIT            1,[HL]
    1E6A 2803        7023                 JR             Z,EXIT_FALSE
                     7024
                     7025 * EXIT HERE IF TRUE
    1E6C 3E01        7026                 LD             A,TRUE
    1E6E C9          7027                 RET
                     7028
                     7029 * EXIT HERE IF FALSE
```

```
CATION OBJECT CODE LINE     SOURCE LINE

  1E6F 3E00        7030 EXIT_FALSE      LD          A,FALSE
  1E71 C9          7031                 RET
                   7032
                   7033
                   7034
  1E72             7035 PUT_TABLE
                   7036                             ; PUTS THE CONTENTS OF WORK_BUFFER[8..15]
                   7037                             ; IN VRAM AT THE GIVEN DESTINATION.
                   7038
  1E72 08          7039                 EX          AF,AF'
  1E73 F5          7040                 PUSH        AF
  1E74 08          7041                 EX          AF,AF'
  1E75 F1          7042                 POP         AF
  1E76 D9          7043                 EXX
  1E77 E5          7044                 PUSH        HL
  1E78 D9          7045                 EXX
  1E79 D1          7046                 POP         DE
  1E7A 2A8006      7047                 LD          HL,[WORK_BUFFER]
  1E7D 010008      7048                 LD          BC,8
  1E80 09          7049                 ADD         HL,BC
  1E81 FD210001    7050                 LD          IY,1
  1E85 CD1C27      7051                 CALL        PUT_VRAM_
  1E88 C9          7052                 RET
                   7053
                   7054
  1E89             7055 GET_COLOR
                   7056                             ; GETS THE COLOR INFORMATION FROM
                   7057                             ; THE APPROPRIATE PLACE IN VRAM
                   7058
  1E89 3E04        7059                 LD          A,COLOR_TABLE
  1E8B D9          7060                 EXX
  1E8C D5          7061                 PUSH        DE
  1E8D D9          7062                 EXX
  1E8E D1          7063                 POP         DE
  1E8F 2A8006      7064                 LD          HL,[WORK_BUFFER]
  1E92 FD210001    7065                 LD          IY,1
  1E96 CD18A3      7066                 CALL        GET_VRAM_
  1E99 C9          7067                 RET
                   7068
                   7069
  1E9A             7070 PUT_COLOR
                   7071                             ; PUTS COLOR INFORMATION IN THE
                   7072                             ; APPROPRIATE PLACE IN  VRAM
                   7073
  1E9A 3E04        7074                 LD          A,COLOR_TABLE
  1E9C D9          7075                 EXX
  1E9D E5          7076                 PUSH        HL
  1E9E D9          7077                 EXX
  1E9F D1          7078                 POP         DE
  1EA0 2A8006      7079                 LD          HL,[WORK_BUFFER]
  1EA3 FD210001    7080                 LD          IY,1
  1EA7 CD1C27      7081                 CALL        PUT_VRAM_
  1EAA C9          7082                 RET
                   7083  PROG
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7085
                     7086 * THE ROUTINES IN THIS MODULE TAKE A SINGLE 8-BYTE BLOCK AS INPUT AND
                     7087 * PRODUCE 4 8-BYTE BLOCKS AS OUTPUT. THEY PERFORM A 2-TO-1 EXPANSION
                     7088 * AND A SIMPLE QUADRUPLING OPEATION RESPECTIVELY.
                     7089
                     7090
                     7091
                     7092                GLB             MAGNIFY
                     7093                GLB             QUADRUPLE
                     7094 * NAMES OF ENTRY POINTS
                     7095
                     7096
                     7097
                     7098
   1EAB              7099 MAGNIFY                        ; PERFORM A 2-TO-1 EXPANSION ON AN
                     7100                                ; 8-BYTE BLOCK OF DATA.
                     7101
                     7102                                ; SOURCE POINTER IN HL, DESTINATION
                     7103                                ; POINTER IN DE.
                     7104
                     7105                                ; DESTROYS IX,IY,AF,BC,DE,HL
                     7106
          <000A>     7107 BYTE_COUNT     EQU             BC
          <000E>     7108 SOURCE         EQU             IX
          <000F>     7109 DESTINATION    EQU             IY
                     7110 * STANDARD NAMES FOR REGISTERS IN THIS ROUTINE
                     7111
                     7112 * BEGIN
   1EAB E5           7113                PUSH            HL              ; SET UP POINTERS
   1EAC DDE1         7114                POP             SOURCE
   1EAE D5           7115                PUSH            DE
   1EAF FDE1         7116                POP             DESTINATION
                     7117
                     7118 *   BYTE_COUNT := 8
   1EB1 010008       7119                LD              BYTE_COUNT,8
                     7120
                     7121 *   REPEAT
   1EB4              7122 MAG_LOOP
                     7123
                     7124 *     EXPAND A BYTE FROM SOURCE
   1EB4 DD7E00       7125                LD              A,[SOURCE+0]
   1EB7 DD23         7126                INC             SOURCE
   1EB9 57           7127                LD              D,A
   1EBA 1E04         7128                LD              E,4
   1EBC CB17         7129 EXP_1          RL              A               ; DOUBLE BITS IN HIGH
   1EBE CB14         7130                RL              H               ; ORDER NIBBLE
   1EC0 CB12         7131                RL              D
   1EC2 CB14         7132                RL              H
   1EC4 1D           7133                DEC             E
   1EC5 20F5         7134                JR              NZ,EXP_1
   1EC7 1E04         7135                LD              E,4
   1EC9 CB17         7136 EXP_2          RL              A               ; DOUBLE BITS IN LOW
   1ECB CB15         7137                RL              L               ; ORDER NIBBLE
   1ECD CB12         7138                RL              D
   1ECF CB15         7139                RL              L
   1ED1 1D           7140                DEC             E
   1ED2 20F5         7141                JR              NZ,EXP_2
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     7142
                     7143 *     WRITE IT TO DESTINATION
     1ED4 FD7400     7144                LD              [DESTINATION+0],H
     1ED7 FD7510     7145                LD              [DESTINATION+16],L
     1EDA FD23       7146                INC             DESTINATION
     1EDC FD7400     7147                LD              [DESTINATION+0],H
     1EDF FD7510     7148                LD              [DESTINATION+16],L
     1EE2 FD23       7149                INC             DESTINATION
                     7150
                     7151 *     DECREMENT BYTE_COUNT
     1EE4 0B         7152                DEC             BYTE_COUNT
                     7153
                     7154 *   UNTIL BYTE_COUNT = 0
     1EE5 79         7155                LD              A,C
     1EE6 B0         7156                OR              B
     1EE7 20CB       7157                JR              NZ,MAG_LOOP
                     7158
                     7159 * END
     1EE9 C9         7160                RET
                     7161
     1EEA            7162 QUADRUPLE                      ; PERFORM A QUADRUPLING ON AN
                     7163                                ; 8-BYTE BLOCK OF DATA.
                     7164
                     7165                                ; SOURCE POINTER IN HL, DESTINATION
                     7166                                ; POINTER IN DE.
                     7167
                     7168                                ; DESTROYS AF,BC,DE,HL,IY
                     7169
                     7170 * BEGIN
                     7171
                     7172 *   BYTE_COUNT := 16
     1EEA 010010     7173                LD              BYTE_COUNT,16
                     7174
                     7175 *   SAVE SOURCE
     1EED E5         7176                PUSH            HL
                     7177
                     7178 *   REPEAT
     1EEE            7179 QUAD_LOOP
                     7180
                     7181 *     GET A BYTE FROM SOURCE
     1EEE 7E         7182                LD              A,[HL]
     1EEF 23         7183                INC             HL
                     7184
                     7185 *     WRITE IT TWICE TO DESTINATION
     1EF0 12         7186                LD              [DE],A
     1EF1 13         7187                INC             DE
     1EF2 12         7188                LD              [DE],A
     1EF3 13         7189                INC             DE
                     7190
                     7191 *     DECREMENT BYTE_COUNT
     1EF4 0B         7192                DEC             BYTE_COUNT
                     7193
                     7194 *      IF BYTE_COUNT = 8 THEN RESTORE SOURCE
     1EF5 79         7195                LD              A,C
     1EF6 FE08       7196                CP              8
     1EF8 2001       7197                JR              NZ,SKIPZZ
     1EFA E1         7198                POP             HL
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1EFB             7199 SKIPZZ
                     7200
                     7201 *   UNTIL BYTE_COUNT = 0
    1EFB 79          7202                 LD          A,C
    1EFC B0          7203                 OR          B
    1EFD 20EF        7204                 JR          NZ,QUAD_LOOP
                     7205
                     7206 * END
    1EFF C9          7207                 RET
                     7208  PROG
```

```
CATION OBJECT CODE LINE     SOURCE LINE

                  7210
                  7211 * THE ROUTINES IN THIS FILE TAKE A SINGLE 8-BYTE BLOCK AS INPUT
                  7212 * AND OPERATE ON IT PRODUCING A SINGLE 8-BYTE BLOCK AS OUTPUT.
                  7213 * THEY PERFORM MIRRORING AROUND THE VERTICAL AXIS, MIRRORING
                  7214 * AROUND THE HORIZONTAL AXIS, AND 90 DEGREE ROTATION.
                  7215
                  7216                GLB         MIRROR_L_R
                  7217                GLB         ROTATE
                  7218                GLB         MIRROR_U_D
                  7219
                  7220
  1F00            7221 MIRROR_L_R                 ; REFLECTS AN 8X8 PIXEL DATA BLOCK
                  7222                            ; AROUND THE VERTICAL AXIS.
                  7223
                  7224                            ; SOURCE IN HL, DEST IN DE
                  7225                            ; DESTROYS AF,BC,DE,HL
                  7226
  1F00 010008     7227          LD    BC,8        ;SET BLOCK BYTE COUNT
  1F03            7228 MIR_L_R10
  1F03 46         7229          LD    B,[HL]      ;GET SOURCE BYTE
  1F04 3E80       7230          LD    A,80H       ;SET WORK REGISTER MASK
  1F06            7231 MIR_L_R20
  1F06 CB10       7232          RL    B           ;PUT SOURCE IN CARRY
  1F08 1F         7233          RRA               ;PUT CARRY INTO WORK REGISTER
  1F09 30FB       7234          JR    NC,MIR_L_R20 ;CONTINUE UNTIL MASK BIT IS IN CARRY
  1F0B 12         7235          LD    [DE],A      ;WRITE MIRRORED BYTE TO DESTINATION
  1F0C 23         7236          INC   HL          ;ADVANCE POINTERS
  1F0D 13         7237          INC   DE
  1F0E 0D         7238          DEC   C
  1F0F 20F2       7239          JR    NZ,MIR_L_R10 ;CHECK COUNT
  1F11            7240 MIR_L_RX
  1F11 C9         7241          RET
                  7242
                  7243
                  7244
  1F12            7245 ROTATE                     ; ROTATE OBJECT 90 DEGREES
                  7246
                  7247                            ; SOURCE IN HL DESTINATION IN DE.
                  7248                            ; DESTROYS AF,BC,DE,HL
                  7249
  1F12 E5         7250          PUSH  HL
  1F13 DDE1       7251          POP   IX
  1F15 EB         7252          EX    DE,HL
  1F16 010008     7253          LD    BC,8
  1F19            7254 TRANSP_10
  1F19 DDCB0016   7255          RL    [IX+0]      ;PUT HI BIT OF FIRST SOURCE BYTE IN CARRY
  1F1D CB1E       7256          RR    [HL]        ;PUT CARRY IN DESTINATION BYTE
  1F1F DDCB0116   7257          RL    [IX+1]
  1F23 CB1E       7258          RR    [HL]
  1F25 DDCB0216   7259          RL    [IX+2]
  1F29 CB1E       7260          RR    [HL]
  1F2B DDCB0316   7261          RL    [IX+3]
  1F2F CB1E       7262          RR    [HL]
  1F31 DDCB0416   7263          RL    [IX+4]
  1F35 CB1E       7264          RR    [HL]
  1F37 DDCB0516   7265          RL    [IX+5]
  1F3B CB1E       7266          RR    [HL]
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1F3D DDCB0616    7267           RL   (IX+6)
    1F41 CB1E        7268           RR   (HL)
    1F43 DDCB0716    7269           RL   (IX+7)
    1F47 CB1E        7270           RR   (HL)
    1F49 23          7271           INC  HL
    1F4A 0D          7272           DEC  C
    1F4B 20CC        7273           JR   NZ,TRANSP_10
    1F4D             7274 TRANSP_X
    1F4D C9          7275           RET
                     7276
                     7277
                     7278
    1F4E             7279 MIRROR_U_D                    ;REFLECT 8X8 PIXEL BLOCK AROUND THE
                     7280                               ; HORIZONTAL AXIS
                     7281
                     7282                               ; SOURCE IN HL, DESTINATION IN DE
                     7283                               ; DESTROYS AF,BC,DE,HL
                     7284
                     7285 * SOURCE  := SOURCE  + 7
    1F4E 010007      7286                LD             BC,7
    1F51 09          7287                ADD            HL,BC
                     7288
                     7289 * BYTE COUNT :=  8
    1F52 03          7290                INC            BC
                     7291
                     7292 * REPEAT
          <1F53>     7293 REFLECT_LOOP   EQU            $
                     7294
                     7295 *   [DESTINATION] := [SOURCE]
    1F53 7E          7296                LD             A,[HL]
    1F54 12          7297                LD             [DE],A
                     7298
                     7299 *    DESTINATION  := SUCC (DESTINATION)
    1F55 13          7300                INC            DE
                     7301
                     7302 *    SOURCE  := PRED (SOURCE)
    1F56 2B          7303                DEC            HL
                     7304
                     7305 *    BYTE COUNT := PRED BYTE COUNT
    1F57 0B          7306                DEC            BC
                     7307
                     7308 * UNTIL BYTE COUNT = 0
    1F58 78          7309                LD             A,B
    1F59 B1          7310                OR             C
    1F5A 20F7        7311                JR             NZ,REFLECT_LOOP
                     7312 * END
    1F5C C9          7313                RET
                     7314
                     7315                ;Modified February 14, 1983.  Filler locations were
                     7316                ;changed to 0FFH to reflect OS_7PRIME.
                     7317
    1F5D FFFFFFFF    7318 HEX FF,FF,FF,FF               ;Filler
                     7319
                     7320
                     7321
```

```
XCATION OBJECT CODE LINE     SOURCE LINE

                    7324 ************************** ROM JUMP_TABLE ****************************
                    7325 ;
                    7326 ;JUMP_TABLE  THIS IS THE JUMP TABLE TO BE USED IN ACCESSING CODE
                    7327 ;            RESIDING IN THE O.S. ROM.  THIS TABLE MUST HAVE ITS
                    7328 ;            ORIGIN REDEFINED TO ACCOUNT FOR GROWTH. PILE NEW ROUTINES
                    7329 ;            AT THE BEGINNING OF THE TABLE MAKING SURE TO INCREMENT
                    7330 *            THE NO_OF_ROUTINES VALUE.
                    7331
                    7332 * NOTE ****
                    7333
                    7334 *               **** NO DELETIONS SHOULD BE MADE FROM ****
                    7335 *               **** THIS TABLE                       ****
                    7336
          <2000>    7337 ROM_END         EQU            2000H
                    7338 * THIS IS THE END OF OS ROM
                    7339
          <0035>    7340 NO_OF_ROUTINES  EQU 53
                    7341 * THIS NUMBER KEEPS COUNT OF THE NUMBER OF ROUTINES ACCESSED THROUGH
                    7342 * THE JUMP TABLE.
                    7343
                    7344 JUMP_TABLE      ORG  ROM_END-(NO_OF_ROUTINES*3)
                    7345
  1F61 C30300       7346 PLAY_SONGS      JP PLAY_SONGS_
  1F64 C30488       7347 ACTIVATEP       JP ACTIVATEQ
  1F67 C306C7       7348 PUTOBJP         JP PUTOBJQ
  1F6A C31D5A       7349 REFLECT_VERTICAL JP RFLCT_VERT
  1F6D C31D60       7350 REFLECT_HORIZONTAL JP RFLCT_HOR
  1F70 C31D66       7351 ROTATE_90       JP ROT_90
  1F73 C31D6C       7352 ENLARGE         JP ENLRG
  1F76 C3114A       7353 CONTROLLER_SCAN JP CONT_SCAN
  1F79 C31188       7354 DECODER         JP DECODER_
  1F7C C31979       7355 GAME_OPT        JP GAME_OPT_
  1F7F C31927       7356 LOAD_ASCII      JP LOAD_ASCII_
  1F82 C318D4       7357 FILL_VRAM       JP FILL_VRAM_
  1F85 C318E9       7358 MODE_1          JP MODE_1_
  1F88 C3116A       7359 UPDATE_SPINNER  JP UPDATE_SPINNER_
  1F8B C31B0E       7360 INIT_TABLEP     JP INIT_TABLEQ
  1F8E C31B8C       7361 GET_VRAMP       JP GET_VRAMQ
  1F91 C31C10       7362 PUT_VRAMP       JP PUT_VRAMQ
  1F94 C31C5A       7363 INIT_SPR_ORDERP JP INIT_SPR_ORDERQ
  1F97 C31C76       7364 WR_SPR_NM_TBLP  JP WR_SPR_NM_TBLQ
  1F9A C30F9A       7365 INIT_TIMERP     JP INIT_TIMERQ
  1F9D C30F88       7366 FREE_SIGNALP    JP FREE_SIGNALQ
  1FA0 C31044       7367 REQUEST_SIGNALP JP REQUEST_SIGNALQ
  1FA3 C3108F       7368 TEST_SIGNALP    JP TEST_SIGNALQ
  1FA6 C31CBC       7369 WRITE_REGISTERP JP REG_WRITEQ
  1FA9 C31CED       7370 WRITE_VRAMP     JP VRAM_WRITEQ
  1FAC C31D2A       7371 READ_VRAMP      JP VRAM_READQ
  1FAF C30655       7372 INIT_WRITERP    JP INIT_QUEUEQ
  1FB2 C30203       7373 SOUND_INITP     JP INIT_SOUNDQ
  1FB5 C30251       7374 PLAY_ITP        JP JUKE_BOXQ
  1FB8 C31B1D       7375 INIT_TABLE     JP INIT_TABLE_
  1FBB C31BA3       7376 GET_VRAM       JP GET_VRAM_
  1FBE C31C27       7377 PUT_VRAM       JP PUT_VRAM_
  1FC1 C31C66       7378 INIT_SPR_ORDER JP INIT_SPR_ORDER_
  1FC4 C31C82       7379 WR_SPR_NM_TBL  JP WR_SPR_NM_TBL_
  1FC7 C30FAA       7380 INIT_TIMER     JP INIT_TIMER_
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    1FCA C30FC4      7381 FREE_SIGNAL     JP FREE_SIGNAL_
    1FCD C31053      7382 REQUEST_SIGNAL  JP REQUEST_SIGNAL_
    1FD0 C310CB      7383 TEST_SIGNAL     JP TEST_SIGNAL_
    1FD3 C30F37      7384 TIME_MGR        JP TIME_MGR_
    1FD6 C3023B      7385 TURN_OFF_SOUND  JP ALL_OFF
    1FD9 C31CCA      7386 WRITE_REGISTER  JP REG_WRITE
    1FDC C31D57      7387 READ_REGISTER   JP REG_READ
    1FDF C31D01      7388 WRITE_VRAM      JP VRAM_WRITE
    1FE2 C31D3E      7389 READ_VRAM       JP VRAM_READ
    1FE5 C30664      7390 INIT_WRITER     JP INIT_QUEUE
    1FE8 C30679      7391 WRITER          JP WRITER_
    1FEB C311C1      7392 POLLER          JP POLLER_
    1FEE C30213      7393 SOUND_INIT      JP INIT_SOUND
    1FF1 C3025E      7394 PLAY_IT         JP JUKE_BOX
    1FF4 C3027F      7395 SOUND_MAN       JP SND_MANAGER
    1FF7 C304A3      7396 ACTIVATE        JP ACTIVATE_
    1FFA C306D8      7397 PUTOBJ          JP PUTOBJ_
    1FFD C30038      7398 RAND_GEN        JP RAND_GEN_
                     7399

Errors=    0
```

LINE#	SYMBOL	TYPE	REFERENCES
7396	ACTIVATE	A	242
7347	ACTIVATEP	A	248
1610	ACTIVATEQ	P	1609,7347
1627	ACTIVATE_	P	1587,1683,7396
1607	ACTIVATE_P	P	1611
1915	ACT_0SPRT	P	1648
1917	ACT_1SPRT	P	1650
1662	ACT_CMPLX	P	1652
1903	ACT_MOBILE	P	1646
1694	ACT_SEMI	P	1644
884	ADD816	P	703,883
5152	ADDR_ADJ	P	5172
5171	ADD_8	P	5152
6141	ADJUST_COUNT	P	6145
6130	ADJUST_INDEX	P	6134
3723	AD_EXIT	P	3719
3718	AD_LP	P	3721
463	AFTER_RANDOM	P	
85	ALEN	A	
1052	ALL_OFF	P	992,7385
6797	ALL_X	P	
492	AMERICA	P	236
87	APS	A	
88	APSV	A	733
964	AREA_SONG_IS	P	963,1216
4504	ARM	A	4856,5027
5008	ARM_DBNCE	P	4859
5037	ARM_EXIT	P	5022,5028,5033
4522	ARM_MASK	A	4746,5011
4516	ARM_OLD	A	5013,5021,5034
5024	ARM_REG	P	5020
5030	ARM_ST1	P	5016
4517	ARM_STATE	A	5014,5026,5036
499	ASCII_TABLE	P	254
5458	ASCII_TBL	P	499,5080
5382	ASC_TABLE	P	5681
86	ASTEP	A	1373,1445,1464
79	ATN	A	1370
724	ATN_SWEEP	P	721,1229
1038	B1	P	1040
6065	BASE_FACTORS	P	6049
3211	BK_CLR	A	3384,3413,3600,3658,3744,3833
3209	BK_PTN	A	3351,3413,3547,3650,3735,3744,3833
368	BOOT_UP	P	
1980	BUFFER	D	2021,2069,2129
7107	BYTE_COUNT	S	
2689	CALC_OFFSET	P	2480,2644
457	CARRY_READY	P	454
261	CARTRIDGE	A	260,534
6042	CASE_OF_CLR10	P	6039
6035	CASE_OF_COLOR	P	6023
6025	CASE_OF_GEN	P	6021
6032	CASE_OF_GEN10	P	6029
5705	CENTER_PRT	P	5234,5244
75	CH	A	
99	CHO	A	
90	CHOEND	A	

LINE#	SYMBOL	TYPE	REFERENCES
94	CHOREP	A	
100	CH1	A	
91	CH1END	A	
95	CH1REP	A	
101	CH2	A	
92	CH2END	A	
96	CH2REP	A	
102	CH3	A	
93	CH3END	A	
97	CH3REP	A	
4784	CHK_PLYR_1	P	4766,4780
4777	CHK_SEG_01	P	4772
4801	CHK_SEG_11	P	4796
4579	CINIT1	P	4587
1677	CMPLX4	P	1690
1691	CMPLX9	P	1676
2823	COLOR	A	
6337	COLORTABLE	D	5902,6330
2830	COLOR_AND_TAG	A	2936,3024,3064,3141
6670	COLOR_TABLE	A	6970,7059,7074
6997	COLOR_TEST	P	6817,6855,6903,6953
3201	COLR	A	3472,3602,3835
3559	COMBINE_LOOP	P	3593
3797	COM_PAT_COL	P	3576
3029	CONTINUE	P	2900,2988
6755	CONTINUE_GRAPHI	P	6690,6709,6730
4534	CONTROLLER_0	A	4611,4712
4535	CONTROLLER_1	A	
4570	CONTROLLER_INIT	P	555,4569
285	CONTROLLER_MAP	A	284,4573,4762
7353	CONTROLLER_SCAN	A	212
4609	CONT_READ	P	4720,4735
4615	CONT_READ1	P	4612
4617	CONT_READX	P	4614
4626	CONT_SCAN	P	4625,4760,7353
80	CTRL	A	1307
4531	CTRL_0_PORT	A	4613,4627,4635,4654
4530	CTRL_1_PORT	A	4616,4630,4638,4668
6432	CTRL_PORT	A	6433,6482,6487,6540,6544,6590,6594,6624
3594	C_LP_EXIT	P	3591
6431	DATA_PORT	A	6433,6550,6599
5050	DBNCE_BUFF	D	4576,4761
4092	DCR_L_MODE_TBL	P	
4104	DCR_L_RPT_TBL	P	4091
4129	DCR_S_MODE_TBL	P	4089
4086	DCR_TIMER	P	4072
833	DECLSN	P	689,735,741,832
850	DECMSN	P	849
7354	DECODER	A	213
4748	DECODERX	P	4728
4701	DECODER_	P	4700,7354
4820	DECODE_0	P	4776,4800
4842	DECODE_0X	P	4835
4854	DECODE_1	P	4783,4807
4866	DECODE_1X	P	4863
4827	DEC_FIRE	P	4823
4861	DEC_KBD	P	4857

LINE#	SYMBOL	TYPE	REFERENCES
4549	DEC_KBD_TBL	P	4739,4899
4715	DEC_PLYR	P	4713
4733	DEC_SEG1	P	4704
4833	DEC_SPNR	P	4829
59	DEDAREA	A	60,61,62,63,64,65
2181	DEFER	A	2184
630	DEFER_WRITES	D	238,559,2086,2091,2153,2183
4599	DELAY	P	4542
5723	DELAY_10	P	5258
7109	DESTINATION	S	
1535	DE_TO_DEST	P	1343,1441,1461,1487
5126	DISPLAY_LOGO	P	566,5082
5054	DIVIDE	P	6058
3304	DLP1	P	3324
3355	DLP2	P	3425
3479	DLP4	P	3542
3613	DLP5	P	3619
3638	DLP6	P	3642
4048	DONE	A	4473,4486
5167	DONE_LOGO	P	5147
1203	DONE_SNDMAN	P	1178
3120	DONT_PUT	P	2876,2879,2889,2892,2964,2967,2977,2980
2188	DO_PUTOBJ	P	2113,2185
1063	DUMAREA	P	992,1044,1168
1876	DUPLI	P	1878
3681	DVEX	P	3677
3676	DVLP	P	3680
1399	EFFECT	P	1342
1233	EFXOVER	P	1213
3705	ELSE04	P	3646
3223	ELSE1	P	3219
3530	ELSE10	P	3516
3608	ELSE13	P	3605
3810	ELSE18	P	3800
3282	ELSE2	P	3278
3841	ELSE23	P	3838
3339	ELSE5	P	3336
3400	ELSE6	P	3378
3463	ELSE8	P	3460
3501	ELSE9	P	3483
4227	ELSEZZ	P	6198,6201
2277	ELSE_1	P	2268
2702	ELSE_11	P	2694
2726	ELSE_12	P	2718
2536	ELSE_8	P	2503
2559	ELSE_9	P	2543,2544
3757	END04	P	3704
3225	END1	P	3221
3537	END10	P	3529
3587	END11	P	3584
3620	END12	P	3598
3610	END13	P	3607
3628	END14	P	3626
3750	END15	P	3728
3782	END16	P	3766
3781	END17	P	3773
3828	END18	P	3809

LINE#	SYMBOL	TYPE	REFERENCES
3815	END19	P	3813
3284	END2	P	3280
3821	END20	P	3819
3827	END21	P	3825
3872	END22	P	3831
3843	END23	P	3840
3852	END24	P	3847
3861	END25	P	3856
3870	END26	P	3865
3321	END3	P	3308
3313	END4	P	3311
3341	END5	P	3338
3420	END6	P	3399
3419	END7	P	3406
3466	END8	P	3462
3538	END9	P	3500
1389	ENDNOREP	P	1342,1382
1384	ENDREP	P	1342
73	ENDSDATA	A	1143,1177
6147	END_ADJ_COUNT	P	6140
6135	END_ADJ_INDEX	P	
376	END_BOOTUP	P	
6232	END_IFZZ	P	6225
2320	END_IF_1	P	2276
2610	END_IF_10	P	2584,2589
2707	END_IF_11	P	2698
2731	END_IF_12	P	2722
6828	END_IF_1_GRAPHI	P	6819
2344	END_IF_2	P	2326
6875	END_IF_2_GRAPHI	P	6857
2363	END_IF_3	P	2358
6914	END_IF_3_GRAPHI	P	6905
2419	END_IF_4	P	2395,2396
6982	END_IF_4_GRAPHI	P	6955
2572	END_IF_8	P	2532
2568	END_IF_9	P	2555
6611	END_INPUT	P	6607
6562	END_OUTPUT	P	6558
7352	ENLARGE	A	246
6746	ENLRG	P	6745,7352
6925	ENLRG_	P	6751
4051	EOT	A	4073,4215,4246,4407,4425,4464
2271	EQUAL_TO	P	2267
4305	EXIT	P	4264
7030	EXIT_FALSE	P	7018,7023
3155	EXIT_PUT_SPR	P	3118
7129	EXP_1	P	7134
7136	EXP_2	P	7141
526	FALSE	A	558,2090,7030
5616	FILL	P	5621
7357	FILL_VRAM	A	252,5780,5905
5610	FILL_VRAM_	P	5084,5129,7357
4501	FIRE	A	4828,4945
4926	FIRE_DBNCE	P	4831
4955	FIRE_EXIT	P	4940,4946,4951
4521	FIRE_MASK	A	4726,4929
4510	FIRE_OLD	A	4931,4939,4952

NE#	SYMBOL	TYPE	REFERENCES
4942	FIRE_REG	P	4938
4948	FIRE_ST1	P	4934
4511	FIRE_STATE	A	4932,4944,4954
3811	FIRST_GEN_NAME	A	3104
4202	FLAGS	A	3225,3375,3596,3603,3644,3798,3829,3836
82	FPS	A	
83	FPSV	A	686
3817	FRAME	A	2926,3014,3055,3092
3814	FRAME_TABLE_PTR	A	2915,3003,3044,3081
4050	FREE	A	4071,4222,4224,4248,4344,4432,4471,4483
4214	FREE1	P	4219
4237	FREE_COUNTER_	P	
4311	FREE_EXIT	P	4216
4221	FREE_MATCH	P	4212
4309	FREE_SET	P	4223,4226,4228
3381	FREE_SIGNAL	A	224
3366	FREE_SIGNALP	A	229
4200	FREE_SIGNALQ	P	4041,7366
4206	FREE_SIGNAL_	P	4040,7381
4197	FREE_SIG_PAR	P	4201
78	FREQ	A	807,811,814
4671	FREQ_SWEEP	P	668,1230
4203	FRM	A	3230,3441,3624
84	FSTEP	A	673,702,1372,1444
4204	F_GEN	A	3269,3301,3623
4333	GAME_NAME	A	332,5230,5232,5237,5247
4355	GAME_OPT	A	250
4777	GAME_OPT	P	5761,7355
4643	GET_BKGRND	P	2377,2642,3262
4055	GET_COLOR	P	6822,6860,6908,6958
4278	GET_NEXT	P	4249,4253,4258,4263,4277
4294	GET_OLD	P	2291
4376	GET_VRAM	A	189,3132,3380,3398,3411,3418
4361	GET_VRAMP	A	194
4077	GET_VRAMQ	P	5980,7361
4087	GET_VRAM_	P	2659,5979,6776,7066,7376
4074	GET_VRAM_P	P	6078
3305	GRAPHICS	A	2913,2914,3001,3002,3042,3043,3079,3080,3102,3103
4772	HEAD_ADDRESS	D	1997,2022,2032,2070
4585	IF11	P	3582
71	INACTIVE	A	1063,1180,1217,1351,1389,1521
4321	INIT_80	P	1822
4358	INIT_QUEUE	P	2057,7390
4350	INIT_QUEUEQ	P	2049,7372
4345	INIT_QUEUE_P	P	2051
4326	INIT_SOUND	P	992,7393
4318	INIT_SOUNDQ	P	1011,7373
4585	INIT_SOUND_DATA	D	1020,1022,1024
4312	INIT_SOUND_PAR	P	1019
4255	INIT_SPR10	P	6260
4378	INIT_SPR_ORDER	A	191
4363	INIT_SPR_ORDERP	A	196
4242	INIT_SPR_ORDERQ	P	5980,7363
4248	INIT_SPR_ORDER_	P	5979,7378
4240	INIT_SPR_P	P	6243
4375	INIT_TABLE	A	188,5649,5654,5659,5664,5669
4346	INIT_TABLE80	P	6017,6024

LINE#	SYMBOL	TYPE	REFERENCES
6061	INIT_TABLE90	P	6031,6034,6041,6044
7360	INIT_TABLEP	A	193
5990	INIT_TABLEQ	P	5980,7360
5997	INIT_TABLE_	P	5979,7375
5987	INIT_TABLE_P	P	5991
7380	INIT_TIMER	A	223
7365	INIT_TIMERP	A	228
4174	INIT_TIMERQ	P	4039,7365
4180	INIT_TIMER_	P	4038,7380
4430	INIT_TIMER_EXI	P	4370,4396,4404
587	INIT_TIME_DATA	D	4176
4171	INIT_TIME_PAR	P	4175
7390	INIT_WRITER	A	203
7372	INIT_WRITERP	A	208
1807	INIT_XP_OS	P	1696,1905
6601	INPUT_LOOP	P	6605,6608
423	IRQ_INTERRUPT	P	
325	IRQ_INT_VECT	A	324,424
4502	JOY	A	4822,4986
4967	JOY_DBNCE	P	4825
4996	JOY_EXIT	P	4981,4987,4992
4523	JOY_MASK	A	4722,4970
4512	JOY_OLD	A	4972,4980,4993
4983	JOY_REG	P	4979
4989	JOY_ST1	P	4975
4513	JOY_STATE	A	4973,4985,4995
1102	JUKE_BOX	P	1075,1385,7394
1095	JUKE_BOXQ	P	1094,7374
1083	JUKE_BOX_PAR	P	1096
7344	JUMP_TABLE	A	
4505	KBD	A	4862,4903
4878	KBD_DBNCE	P	4865
4913	KBD_EXIT	P	4893,4904,4909
4520	KBD_MASK	A	4738,4882
4529	KBD_NULL	A	4550,4557,4561,4565
4518	KBD_OLD	A	4884,4892,4910
4895	KBD_REG	P	4891
4906	KBD_ST1	P	4887
4519	KBD_STATE	A	4885,4897,4912
1143	L1	P	1152
1229	L10	P	1222
1244	L12	P	1232,1240
1377	L13	P	1361
1394	L14	P	1379
1424	L15	P	1397
1448	L16	P	1429
1467	L17	P	1450
1482	L18	P	1479
1534	L19	P	
1176	L2	P	1202
684	L20	P	676
1528	L20_LOAD_NEX	P	1526
708	L21	P	691
766	L22	P	737,761
763	L23	P	743
793	L24	P	787
1306	L5	P	1302

LINE#	SYMBOL	TYPE	REFERENCES
1318	L6	P	1304,1312
1327	L7	P	1323
1330	L8	P	1325
1199	L9	P	1182
942	LEAVE_EFFECT	P	941
5922	LINE_1	P	5797
5931	LINE_10	P	5854
5932	LINE_11	P	5959
5933	LINE_12	P	5965
5923	LINE_2	P	5803
5924	LINE_3	P	5941
5925	LINE_4	P	5947
5926	LINE_5	P	5950
5927	LINE_6	P	5953
5928	LINE_7	P	5842
5929	LINE_8	P	5846
5930	LINE_9	P	5850
7356	LOAD_ASCII	A	251,5792
5681	LOAD_ASCII_	P	5083,5137,7356
1346	LOAD_NEXT_NOTE	P	1120,1235,1342
268	LOCAL_SPR_TBL	A	267,6207,6300
5304	LOGO_COLORS	P	5205
5329	LOGO_GEN	P	5150
5309	LOGO_NAMES	P	5177,5183,5197,5270,5276,5282
4052	LONG	A	4088,4227,4372
4371	LONG_TIMER	P	4355
3961	LP1	P	4002
4009	LP2	P	4022
2512	LT33	P	2510
7099	MAGNIFY	P	6935,7092
7122	MAG_LOOP	P	7157
6763	MAIN_LOOP	P	6794
4415	MAKE_NEW_TIMER	P	4408
7221	MIRROR_L_R	P	6811,7216
7279	MIRROR_U_D	P	6849,6869,7218
7228	MIR_L_R10	P	7239
7231	MIR_L_R20	P	7234
7240	MIR_L_RX	P	
1517	MODB0	P	1343,1374,1391,1419,1446,1465,1496
5075	MODE_0_PORT	A	5617,6312
7358	MODE_1	A	253,5783
5636	MODE_1_	P	5085,5132,7358
5076	MODE_1_PORT	A	5612,5615,6293,6296
4285	MOVE_IT	P	4247
865	MSNTOLSN	P	693,739,864
636	MUX_SPRITES	D	237,562,6199
2304	M_XY	P	2303,2305
2411	M_XY2	P	2410,2412
2829	NAME	A	3107
2453	NEGTV	P	2445
4245	NEXT	P	4284
77	NEXTNOTEPTR	A	
1868	NEXT_COLOR	P	1891
2820	NEXT_GEN	A	
4070	NEXT_TIMER0	P	4078
4406	NEXT_TIMER1	P	4345
4496	NEXT_TIMEP *A	D	4031,4184,4291,4303,4378,4395

LINE#	SYMBOL	TYPE	REFERENCES
81	NLEN	A	677,680,1371
485	NMI_INTERRUPT	P	
329	NMI_INT_VECT	A	328,486
3695	NOTOS	P	3683,3687
4397	NOT_A_LONG_REPE	P	4375
4365	NOT_A_REPEAT_TI	P	4363
448	NOT_ON	P	443
6495	NOT_REG_0	P	6492
6503	NOT_REG_1	P	6500
2026	NOT_TOO_BIG	P	2014
5270	NO_CARTRIDGE	P	5222,5226
7340	NO_OF_ROUTINES	A	7344
545	NO_TEST_	P	537,540
1272	NSRHI	A	
1274	NSRLOW	A	
1273	NSRMED	A	
1275	NSRTG3	A	
1624	NTZZZ_	P	1621
503	NUMBER_TABLE	P	255
5422	NUMBER_TBL	P	503,5081
2813	NUMGEN	A	
4525	NUM_DEV	A	4577,4791,5050
2810	OBJECT_TYPE	A	
3210	OBJ_PTR	A	3452,3475,3554
5586	OBJ_TABLE	P	5140
67	OFF	A	1054,1056,1058,1060,1278,1284,1290,1296
2880	OK__1	P	2874
2893	OK__2	P	2887
2968	OK__3	P	2962
2981	OK__4	P	2975
1862	ONE_BYTE	P	1850
1822	OS_IN_VRAM	P	1817
6552	OUTPUT_LOOP	P	6556,6559
6313	OUTPUT_LOOP10	P	6317
6299	OUTPUT_LOOP_TAB	P	6322
1854	O_B_RET	P	1892
649	PARAM_	P	647,1021,1098,1613,2053,2176,4177,4203,4333,4452,5993,6080,6179,6245,6272,6472,6523,6580
584	PARAM_AREA	D	1097,1099,2052,2054,2055,2175,2177,2178,5992,5994,5995,6079,6081,6082,6083,6084,6178,6180,6181,6182,6183 6244,6246,6271,6273,6471,6473,6522,6524,6525,6526,6579,6581,6582,6583
5697	PARSE	P	5231,5238,5242,5248,5250
1416	PASS1	P	1343,1413
6669	PATTERN_GEN	A	7017
6336	PATTRNGENTBL	D	6329
6335	PATTRNNAMETBL	D	6328
1271	PERIOD	A	
2485	PF1	P	
2538	PF2	P	
2561	PF3	P	
2579	PF4	P	
7394	PLAY_IT	A	217
7374	PLAY_ITP	A	221
7346	PLAY_SONGS	A	219
1276	PLAY_SONGS_	P	1255,7346
4506	PLYR_0	A	4769
4507	PLYR_1	A	4793
3256	PM1	P	
3513	PM10	P	

LINE#	SYMBOL	TYPE	REFERENCES
3545	PM11	P	
3644	PM12	P	
3706	PM13	P	
3761	PM14	P	
3829	PM15	P	
3264	PM2	P	
3291	PM3	P	
3328	PM4	P	
3347	PM5	P	
3372	PM51	P	3369
3368	PM52	P	3371
3377	PM6	P	
3404	PM7	P	
3432	PM8	P	
3473	PM9	P	
7392	POLLER	A	210
4759	POLLER_	P	4758,7392
4808	POLLER_X	P	4788,4804
888	POS	P	886
531	POWER_UP	P	375
586	PRM_AREA	D	1612,1614,1619
1215	PROCESS_DATA_AR	P	1147,1213
2812	PTRN_POINTER	A	
60	PTR_TO_LST_OF_S	A	909,1028
61	PTR_TO_S_ON_0	A	1045,1169,1190,1298
62	PTR_TO_S_ON_1	A	1046,1170,1281
63	PTR_TO_S_ON_2	A	1047,1171,1287
64	PTR_TO_S_ON_3	A	1048,1172,1293
906	PT_IX_TO_SxDATA	P	905,1105,1141,1174
2856	PUTOSPRITE	P	2197,2853
2944	PUT1SPRITE	P	2199,2853
1913	PUTCOMPLEX	P	2200,3911
2473	PUTFRAME	P	2257,2340,2387,2472,3780,3791
7397	PUTOBJ	A	241,4018
7348	PUTOBJP	A	247
2173	PUTOBJQ	P	2165,7348
2182	PUTOBJ_	P	2158,7397
2166	PUTOBJ_PAR	P	2174
2216	PUTSEMI	P	2193,2213
6070	PUT_COLOR	P	6825,6872,6911
3212	PUT_MOBILE	P	2195,3196
6035	PUT_TABLE	P	6814,6852,6900
7377	PUT_VRAM	A	190,3116,3150,3654,3692,3742,3749,5160,5181,5187,5194,5201,5209,5255,5274,5280,5286,5685,5692,5720,5801 5807,5944,5956,5962,5968
7362	PUT_VRAMP	A	195
5176	PUT_VRAMQ	P	5980,7362
5185	PUT_VRAM_	P	1775,1803,1838,1853,1882,1946,2599,5979,6950,6979,7051,7081,7377
5173	PUT_VRAM_P	P	6177
4067	PUT_Y_AND_NAME	P	2938,3026
4435	PX_TO_PTRN_POS	P	2222,2227,2434,3242,3251
698	P_LOOP	P	5703
7162	QUADRUPLE	P	6967,7093
7179	QUAD_LOOP	P	7204
968	QUEUE_HEAD	D	2008,2018,2029,2065,2096
964	QUEUE_SIZE	D	2012,2061,2120
969	QUEUE_TAIL	D	2066,2095,2116,2126,2138
7398	RAND_GEN	A	239

LINE#	SYMBOL	TYPE	REFERENCES
440	RAND_GEN_	P	7398
644	RAND_NUM	D	441,460,552,643
7387	READ_REGISTER	A	200,5622
7389	READ_VRAM	A	202,3279,3461,3521
7371	READ_VRAMP	A	207
7350	REFLECT_HORIZON	A	244
7293	REFLECT_LOOP	P	7311
7349	REFLECT_VERTICA	A	243
6621	REG_READ	P	6620,7387
6478	REG_WRITE	P	6062,6477,7386
6469	REG_WRITEQ	P	6468,7369
6464	REG_WRITE_P	P	6463,6470
4049	REPEAT	A	4090,4135,4225,4364,4380,4481
596	REPEAT_SIG_CODE	D	4332,4335
7382	REQUEST_SIGNAL	A	225
7367	REQUEST_SIGNALP	A	230
4330	REQUEST_SIGNALQ	P	4043,7367
4337	REQUEST_SIGNAL_	P	4042,7382
4327	REQUEST_SIG_PAR	P	4331
455	RESET	P	447,450
1363	REST	P	1342
6782	RETURN_HERE	P	6835,6882,6921,6992
6702	RFLCT_HOR	P	6701,7350
6839	RFLCT_HOR_	P	6706
6683	RFLCT_VERT	P	6682,7349
6801	RFLCT_VERT_	P	6687
7337	ROM_END	A	7344
7245	ROTATE	P	6897,7217
7351	ROTATE_90	A	245
6723	ROT_90	P	6722,7351
6886	ROT_90_	P	6727
3664	RPT1	P	3701
3708	RPT2	P	3755
2577	RPT_1	P	2627
2652	RPT_2	P	2675
393	RST_10H	P	
305	RST_10H_RAM	A	304,393
396	RST_18H	P	
309	RST_18H_RAM	A	308,396
399	RST_20H	P	
313	RST_20H_RAM	A	312,399
402	RST_28H	P	
317	RST_28H_RAM	A	316,402
405	RST_30H	P	
321	RST_30H_RAM	A	320,405
390	RST_8H	P	
301	RST_8H_RAM	A	300,390
5054	S0_C0	D	4591,4629,4774
5055	S0_C1	D	4592,4632,4798
5056	S1_C0	D	4593,4637,4782
5057	S1_C1	D	4594,4640,4806
6342	SAVED_COUNT	D	6113,6137,6146,6159
4153	SAVE_2_BYTES	P	4100,4116
65	SAVE_CTRL	A	1051,1309
6341	SAVE_TEMP	D	
4079	SCRAM	P	4074
72	SEFFECT	A	

LINE#	SYMBOL	TYPE	REFERENCES
4508	SEG_0	A	4771,4795
4509	SEG_1	A	4779,4803
1750	SEMI_BOT	P	1748
1755	SEMI_EXIT	P	1711,1753
1759	SEMI_GRI	P	1715
1745	SEMI_MID	P	1743
452	SET	P	446
6102	SET_COUNT	P	6097,6229
6124	SET_COUNT10	P	6118
6150	SET_COUNT20	P	6122,6129
6160	SET_COUNTX	P	
4145	SET_DONE_BIT	P	4103,4128,4136
2035	SET_UP_ENDIF	P	2024
1991	SET_UP_WRITE	P	2186
2824	SHAPE	A	
3575	SHFEX	P	3571
3570	SHFLP	P	3574
6164	SHIFT_CT	P	6125
4475	SIGNAL_FALSE	P	4465,4472
4470	SIGNAL_MATCH	P	4461
594	SIGNAL_NUM	D	4202,4204
4480	SIGNAL_TRUE	P	4474
4484	SIGNAL_TRUE1	P	4482
7199	SKIPZZ	P	7197
2317	SKIP_OLD	P	2293
1825	SM_BY_OLD	P	1814,1820
1138	SND_MANAGER	P	1135,7395
76	SONGNO	A	
7393	SOUND_INIT	A	215
7373	SOUND_INITP	A	220
7395	SOUND_MAN	A	218
69	SOUND_PORT	A	795,810,821,1055,1057,1059,1061,1303,1324
7108	SOURCE	S	
5390	SPACE	P	5688
4503	SPIN	A	4834,4838,4839
4514	SPIN_OLD	A	
4515	SPIN_STATE	A	
5051	SPIN_SW0_CT	D	4589,4655,4710,4775
5052	SPIN_SW1_CT	D	4590,4799
4524	SPNR_MASK	A	
5334	SPRITEGENTBL	D	6327
5333	SPRITENAMETBL	D	6326
2807	SPRITE_INDEX	A	3112,3129,3147
275	SPRITE_ORDER	A	274,6254,6283
2798	SPRITE_PTR	S	
997	SR1ATN	A	1054,1278,1279
996	SR1FRQ	A	1280
999	SR2ATN	A	1056,1284,1285
998	SR2FRQ	A	1286
1001	SR3ATN	A	1058,1290,1291
1000	SR3FRQ	A	1292
1004	SRNATN	A	1060,1296,1297
1003	SRNCTL	A	1314
578	STACK	D	233,371
290	START_GAME	A	289,541,5266
2806	STATUS	A	2865,2866,2924,2925,2953,2954,3012,3013,3034,3035,3053,3054,3070,3071,3090,3091
4532	STRB_RST_PORT	A	4571,4641,4737

LINE#	SYMBOL	TYPE	REFERENCES
4533	STRB_SET_PORT	A	4633,4734
5053	STROBE_FLG	D	
4536	STROBE_RESET	A	
4537	STROBE_SET	A	4703
4266	SUBTRACT_4	P	4259
1831	SUP_GEN_CLR	P	1744,1749,1754
1894	SUP_UPDATE	P	1746,1751
2347	SV1	P	
2393	SV2	P	
573	SYSTEM_RAM_AREA	D	578
2260	S_OLD_SCRN	P	2255
1973	TAIL_ADDRESS	D	2071,2101,2130,2142
3981	TBL0	P	3978
588	TEMP1	D	4178
591	TEMP2	D	4179
4463	TEST1	P	4468
4489	TEST_EXIT	P	4478
7383	TEST_SIGNAL	A	226
7368	TEST_SIGNALP	A	231
4449	TEST_SIGNALQ	P	4045,7368
4455	TEST_SIGNAL_	P	4044,7383
601	TEST_SIG_NUM	D	4451,4453
4446	TEST_SIG_PARAM	P	4450
2802	THIS_SPRITE	S	
4343	TIMER1	P	4413,4428
4399	TIMER2	P	
5724	TIMER_1	P	5289,5732
5725	TIMER_2	P	5728
4147	TIMER_EXIT	P	4132,4157
598	TIMER_LENGTH	D	4334
4495	TIMER_TABLE_BAS	D	4030,4069,4181,4208,4243,4340,4457
7384	TIME_MGR	A	227
4067	TIME_MGRQ	P	4047
4068	TIME_MGR_	P	4046,7384
1319	TONE_OUT	P	1255,1282,1288,1294
5322	TRADEMARK	P	5190
7254	TRANSP_10	P	7273
7274	TRANSP_X	P	
527	TRUE	A	6818,6856,6904,6954,7026
7385	TURN_OFF_SOUND	A	216,548
1432	TYPE0	P	1342
1453	TYPE1	P	1342
1472	TYPE2	P	1342
1501	TYPE3	P	1342,1469
1625	TZZZ	P	1623
783	UPATHCTRL	P	781,1306,1315,1327
4665	UPDATE_R0	P	4660
4679	UPDATE_R1	P	4674
4668	UPDATE_S1	P	4657,4663
7359	UPDATE_SPINNER	A	211
4653	UPDATE_SPINNER_	P	4652,7359
4681	UPDATE_SPINX	P	4670,4677
806	UPFREQ	P	805,1328
1166	UP_CH_DATA_PTRS	P	1121,1164,1241
608	VDP_MODE_WORD	D	235,1713,3216,6015,6120,6494,6502,7021
623	VDP_STATUS_BYTE	D	234
6332	VRAM_ADDR_TABLE	D	6009,6114,6286,6325

LINE#	SYMBOL	TYPE	REFERENCES
6586	VRAM_READ	P	2287,2316,6098,6585,7389
6577	VRAM_READQ	P	6576,7371
6572	VRAM_READ_P	P	6578
6529	VRAM_WRITE	P	1824,2417,6230,6528,7388
6520	VRAM_WRITEQ	P	6519,7370
6515	VRAM_WRITE_P	P	6521
1270	WAITE	A	
280	WORK_BUFFER	A	279,1873,2280,2361,2401,2859,2947,3215,3256,3268,3271,3291,3295,3300,3329,3347,3373,3382,3440,3451,3467 3471,3474,3545,3546,3599,3633,3649,3656,3657,3660,3703,3723,3737,3761,3774,3784,3785,6775,6805,6843,6863 6891,6929,6946,6961,6975,7047,7064,7079
7391	WRITER	A	204
2083	WRITER	P	2082,7391
5954	WRITE_CHAR	P	5844,5848,5852,5856,5948,5951
5959	WRITE_L11	P	5868,5871,5874,5877
5965	WRITE_L12	P	5889,5892,5895,5898
5941	WRITE_L3	P	5810,5813,5816,5819,5822,5825,5828,5831
5947	WRITE_L4	P	5834,5859,5880
5950	WRITE_L5	P	5837,5862,5883
5953	WRITE_L6	P	5840,5865,5886
5142	WRITE_LOOP	P	5165
5177	WRITE_NAMES	P	5169
7386	WRITE_REGISTER	A	199,5216,5263,5293,5638,5642,5674,5788,5912
7369	WRITE_REGISTERP	A	205
7388	WRITE_VRAM	A	201,3337
7370	WRITE_VRAMP	A	206
2135	WRTR_ELSE	P	2122
2145	WRTR_END_IF	P	2133
2149	WRTR_END_WHILE	P	2098
2094	WRTR_WHILE	P	2147
7379	WR_SPR_NM_TBL	A	192
7364	WR_SPR_NM_TBLP	A	197
6269	WR_SPR_NM_TBLQ	P	5980,7364
6275	WR_SPR_NM_TBL_	P	5979,7379
6267	WR_SPR_P	P	6270
2828	X	A	2910,2998,3039,3137
3200	XDISP	A	3239,3568
3208	XP_BK	A	3243,3260,3330,3681,3768,3788
2490	XP_NEG	P	2487
3206	XP_OS	A	3272,3762,3777
2498	X_IN_BOUNDS	P	
2818	X_LOCATION	A	2867,2955,3036
2827	Y	A	3075
3199	YDISP	A	3248,3549,3577,3579
3207	YP_BK	A	3252,3257,3259,3292,3348,3374,3634,3661,3685,3724,3769,3786,3789
3205	YP_OS	A	3296,3767,3775,3778
2819	Y_LOCATION	A	3072