```
;robotron 7800 NTSC
;
;32K bytes, checksum = BFDC
;
;disassembled by BLUE AZURE February 2013
;
         processor 6502
;
******************************************************
******************************************************
*                                                    *
*                                                    *
*    ROBOTRON     8-JUNE-83                          *
*                16-JUNE-83                          *
*                18-JULY-83                          *
*                24-AUGUST-83            3:30        *
*                                                    *
*                                                    *
*         RMAIN.S                                    *
*                                                    *
******************************************************


**********
*
*         STUFF FROM LINKER
*
*          EXTRN   STAMPHGH,CRETODST,DIRTOSTE
*          EXTRN   STAMPL,STAMPS,PALNWID,STAMPPWD
*

******************************************
******************************************
*                                        *
*    ROBOTRON RAM AND MAIN ROUTINES      *
*                                        *
******************************************


******************************************
*
*         ZERO PAGE LOCATIONS
*
******************************************
*

TEMP0     EQU     $A0    ;TEMPS
TEMP1     EQU     $A1
TEMP2     EQU     $A2
TEMP3     EQU     $A3
TEMP4     EQU     $A4
TEMP5     EQU     $A5
TEMP6     EQU     $A6
TEMP7     EQU     $A7
TEMP8     EQU     $A8
TEMP9     EQU     $A9
TEMP10    EQU     $AA
TEMP11    EQU     $AB
TEMP12    EQU     $AC
TEMP13    EQU     $AD
TEMP14    EQU     $AE
TEMP15    EQU     $AF
*
*
*
TEMP16    EQU     $B0
TEMP17    EQU     $B1
TEMP18    EQU     $B2
TEMP19    EQU     $B3
TEMP20    EQU     $B4
TEMP21    EQU     $B5
TEMP22    EQU     $B6
```

```
FRMCNT    EQU     $B7    ;FRAME COUNT

TEMPX     EQU     $B8    ;MISCELLANEOUS TEMPS
TEMPY     EQU     $B9

TADDRL    EQU     $BA    ;TEMPS USED TO INDEX THRU TABLES
TADDRH    EQU     $BB
TADDR1L   EQU     $BC
TADDR1H   EQU     $BD

XORIG     EQU     $BE    ;ORIGINAL XPOS OF AN OBJECT (BEFORE A MOVE)
YORIG     EQU     $BF
XINTEND   EQU     $C0    ;INTENDED XPOS OF AN OBJECT (WHERE IT WANTS TO MOVE)
YINTEND   EQU     $C1
XXINTEND  EQU     $BE
YXINTEND  EQU     $BF
*
*
TEMPZON   EQU     $C2    ;TEMP VARIABLE USED TO HOLD A ZONE #

RANDOMX   EQU     $C3    ;RANDOM VALID X AND Y POSITIONS
RANDOMY   EQU     $C4    ;GENERATED BY RANDXY AND RANDXYBX

RNDM      EQU     $C5    ;FOR RANDOM NUMBER GENERATOR
*         ALSO NEEDS $C6


*         OTHER MISCELLANEOUS VARIABLES
TEMPCOL   EQU     $C7    ;TEMP VARIABLE FOR COLOR CYCLING
MCCTMR    EQU     $C8    ;TIMER FOR MC COLLISIONS

CRELEFT   EQU     $C9    ;COUNT OF CREATURES LEFT ALIVE
                  ; INC WHEN ONE BORN, DEC WHEN ONE DIES
*
*         VARIABLES FOR SOUND GENERATION
SOUNDZP   EQU     $CA    ;2 BYTES - ZERO PAGE FOR SOUND
          ;ALSO USES $CB

*         OBJECT SPEED, STEP AND OTHER LOCATIONS
*           SPEEDS, STEPS, ETC. WHICH CHANGE OVER TIME OR FROM
*           RACK TO RACK ARE HERE AS RAM LOCATIONS
*                 CONSTANTS ARE IN ASSEMBLY CONSTANTS SECTION
*
GSPEED    EQU     $CC    ;# OF FRAMES BETWEEN GRUNT MOVES
FSPEED    EQU     $CD    ;# FRAMES BETWEEN FAMILY MOVES
HSPEED    EQU     $CE    ;# OF FRAMES BETWEEN HULK MOVES
SQBTIME   EQU     $CF    ;BASE TIME UNTIL FIRST BIRTH IN A WAVE
QSPEED    EQU     $D0    ;# OF FRAMES BETWEEN QUARK MOVES
TSPEED    EQU     $D1    ;# OF FRAMES BETWEEN TANK MOVES
BSPEED    EQU     $D2    ;# OF FRAMES BETWEEN BRAIN MOVES
BSTIME    EQU     $D3    ;BASE # OF FRAMES BETWEEN BRAIN SHOTS

*         OTHER GLOBAL VARIABLES:
FAMLEVEL  EQU     $D4    ;SCORE LEVEL FOR FAMILY PICKING UP
PLAYER    EQU     $D5    ;0 OR 1 FOR PLAYER 1 OR 2
WAVENUM   EQU     $D6    ;THE NUMBER WAVE WE ARE CURRENTLY ON

*      THESE VARIABLES MUST BE SEQUENTIAL IN RAM
*           THERE ARE STARTNUM OF THESE LOCATIONS - PRESERVE CURRENT STATUS
*      USED AT WAVE START AND BETWEEN TURNS
*      THESE MAY BE WIPED OUT BY WAVE START ROUTINES - DO NOT USE DURING WAVE
*      THESE SHOULD BE RE-COMPUTED FROM THE OBJECT DATA TABLES AT THE END
*                 OF A TURN OR WAVE
*                              THE 14 OBJECT TYPES:
GNUM      EQU     $D7    ;# GRUNTS
MONUM     EQU     $D8    ;# MOMMIES
DNUM      EQU     $D9    ;# DADDIES
MINUM     EQU     $DA    ;# MIKEYS
HNUM      EQU     $DB    ;# HULKS
SNUM      EQU     $DC    ;# SPHEROIDS
QNUM      EQU     $DD    ;# QUARKS
ENUM      EQU     $DE    ;# ENFORCERS (NULL, AT WAVE START)
```

```
TNUM      EQU     $DF    ;# TANKS
BNUM      EQU     $E0    ;# BRAINS
PNUM      EQU     $E1    ;# PROGS (NULL, AT WAVE START)
EMNUM     EQU     $E2    ;# ENFORCER MISSILES (NULL, AT WAVEST)
CMNUM     EQU     $E3    ;# CRUISE MISSILES (NULL, AT WAVE ST)
TMNUM     EQU     $E4    ;# TANK MISSILES (NULL, AT WAVE START)
*                               OTHER WAVE-DEPENDANT NUMBERS:

ELECNUM   EQU     $E5    ;# ELECTRODES


*         POINTERS TO SEGMENTS WITHIN OBJECT DATA TABLES
*         THESE CONTAIN INDICES TO THE STARTS OF VARIOUS SECTIONS
                    ;GRUNTS START AT FIRST BYTE, INDEX=1 (NOTE: NOT 0)
FPTR      EQU     $E6    ;START OF FAMILY (NO SEPARATE MO,D,MI)
HPTR      EQU     $E7    ;         HULKS
SPTR      EQU     $E8    ;         SPHEROIDS
QPTR      EQU     $E9    ;         QUARKS
EPTR      EQU     $EA    ;         ENFORCERS
TPTR      EQU     $EB    ;         TANKS
BPTR      EQU     $EC    ;         BRAINS
PPTR      EQU     $ED    ;         PROGS
MPTR      EQU     $EE    ;         MISSILES (NO SEPARATE E,C,T)

***************************************
*                                     *
*         OTHER LOCATIONS             *
*                                     *
***************************************
*

*         NOT-ZERO-PAGE RAM LOCATIONS

*
*         SCORE VARIABLES IN BCD FORMAT
*                 USE START OF FREE RAM NOT SHADOWED ANYWHERE
SCORE1L   EQU     $1000 ;PLAYER 1 LOWER 2 DIGITS
SCORE1M   EQU     $1001 ;         MIDDLE 2 DIGITS
SCORE1H   EQU     $1002 ;         HIGH 2 DIGITS
SCORE1V   EQU     $1003 ;         VERY HIGH 2 DIGITS

SCORE2L   EQU     $1004 ;PLAYER 2 LOWER 2 DIGITS
SCORE2M   EQU     $1005 ;         MIDDLE 2 DIGITS
SCORE2H   EQU     $1006 ;         HIGH 2 DIGITS
SCORE2V   EQU     $1007 ;         VERY HIGH 2 DIGITS

*

*         SOUND ROUTINE VARIABLES
TUNON     EQU     $1300 ;2 BYTES - WHETHER TUNE IS ACTIVE
TUNINDEX  EQU     $1302 ;2 BYTES - WHAT TUNE IS PLAYING
TUNPRIOR  EQU     $1304 ;2 BYTES - WHAT THE PRIORITY OF TUNE IS
TUNBASE   EQU     $1306 ;2 BYTES - BASE ADDRESS OF TUNE DATA
TUNBASE1  EQU     $1308 ;2 BYTES - HI BYTE OF BASE ADDRESS
FREQOFF   EQU     $130A ;2 BYTES - OFFSET INTO DATA FOR FREQ'S
CTLOFF    EQU     $130C ;2 BYTES - OFFSET INTO DATA FOR CTL'S
VOLOFF    EQU     $130E ;2 BYTES - OFFSET INTO DATA FOR VOL'S
FREQTIME  EQU     $1310 ;2 BYTES - NUMBER FRAMES TILL NEXT FREQ
CTLTIME   EQU     $1312 ;2 BYTES - NUMBER FRAMES TILL NEXT CTL
VOLTIME   EQU     $1314 ;2 BYTES - NUMBER FRAMES TILL NEXT VOL
TUNNUM    EQU     $1316 ;WHAT TUNE YOU WANT - PARAMETER
TUNTEMP0  EQU     $1317 ;TEMP VALUE FOR TUNE DRIVER
TUNTEMP1  EQU     $1318 ;TEMP VALUE FOR TUNE DRIVER

************************************************************************************
*
*         OBJECT DATA TABLES
*             THESE ARE PARALLEL TABLES WHICH HAVE VARIOUS STATE VARIABLES
*         FOR EACH OBJECT.  ONE INDEX (OBJECT INDEX) IS USED TO INDEX INTO
*         ANY ONE OF THESE TABLES TO FIND INFO FOR A SPECIFIC OBJECT.
*         THE OBJECT INDEX IS A ONE-BYTE QUANTITY FROM 1 TO MAXOBJS-1
*         OBJECT 0 IS THE MUTANT CLONE,
*                 OBJECTS MAXOBJS-4 THROUGH MAXOBJS-1 ARE THE 4 MC SHOTS.
```

```
MAXOBJS   EQU     80
NUMTBLS   EQU     16    ;NUMBER OF OBJECT DATA TABLES

          ORG     $1319
XTBL      DS      MAXOBJS                                           ;X POSITION
YTBL      DS      MAXOBJS                                           ;Y POSITION
MTTBL     DS      MAXOBJS                                           ;MOVE TIMER - # FRAMES TIL NEXT
MOVE
DXTBL     DS      MAXOBJS                                           ;X-COMP OF CURRENT DIR, SPEED
MOVING
DYTBL     DS      MAXOBJS                                           ;Y-COMP OF CURRENT DIR, SPEED
MOVING
DTTBL     DS      MAXOBJS                                           ;DIR TIMER - # MOVES TIL DIR
CHANGE
SATBL     DS      MAXOBJS                                           ;STEP IN ANIMATION + TYPE DATA
MISCTBL   DS      MAXOBJS                                           ;MISCELLANEOUS USE
DLPHTBL   DS      MAXOBJS                                           ;HIGH BYTE OF ABSOLUTE ADDR OF
DL ENTRY
DLPLTBL   DS      MAXOBJS                                           ;LOW BYTE OF ABSOLUTE ADDR OF
DL ENTRY
DL2PTBL   DS      MAXOBJS                                           ;DIFFERENCE BETWEEN 2 DL
ENTRIES
CRTBL     DS      MAXOBJS                                           ;CREATURE TYPE
XEXTBL    DS      MAXOBJS                                           ;X-POSITION OF HORIZONTAL
EXTENT
YEXTBL    DS      MAXOBJS                                           ;Y-POSITION OF VERTICAL
EXTENT
STTBL     DS      MAXOBJS                                           ;STATUS TABLE
ZONTBL    DS      MAXOBJS                                           ;ZONE - USED BY THE UNLOADER

*         MUTANT CLONE STATISTICS:       ZEROTH ENTRIES IN OBJECT DATA TABLES
MCXPOS    EQU     XTBL   ;X POSITION
MCYPOS    EQU     YTBL   ;Y POSITION
MCXEX     EQU     XEXTBL;X EXTENT (ABSOLUTE POSITION)
MCYEX     EQU     YEXTBL;Y EXTENT (ABSOLUTE POSITION)
MCMTMR    EQU     MTTBL  ;MC MOVE TIMER - # FRAMES TIL MOVE
MCSTMR    EQU     MISCTBL                                           ;MC SHOT TIMER-MC CAN FIRE IF
NEGATIVE
MCDIR     EQU     DXTBL  ;CURRENT DIRECTION OF MC
MCSA      EQU     SATBL  ;CURRENT MC STEP IN ANIMATION

*         TABLE OF THE 4 MC SHOTS:    **** SHOT TABLES ****
*                 THESE ARE PARALLEL DATA TABLES, INDEX INTO THEM WITH THE
*            SHOT NUMBER, 0 TO 3.
*         THESE TABLES USE THE LAST 4 ENTRIES IN THE OBJECT DATA TABLES
SDIRTBL   EQU     DXTBL+MAXOBJS-4                                   ;SHOT DIR (4-BIT), 0 = NULL
SXTBL     EQU     XTBL+MAXOBJS-4                                    ;X POS
SYTBL     EQU     YTBL+MAXOBJS-4                                    ;Y POS
SXEXTBL   EQU     XEXTBL+MAXOBJS-4                                  ;X EXTENT
SYEXTBL   EQU     YEXTBL+MAXOBJS-4                                  ;Y EXTENT
SSATBL    EQU     SATBL+MAXOBJS-4                                   ;ANIMATION STEP - ALWAYS 0
SSTTBL    EQU     STTBL+MAXOBJS-4                                   ;STATUS
SCRTBL    EQU     CRTBL+MAXOBJS-4                                   ;CREATURE TYPE - ALWAYS
#MCSCODE

*         OBJECT DATA TABLES EXTEND UP TO $1319+NUMTBLS*MAXOBJS

*
*
ZONOBJC   EQU     $18A4  ;ZONE OBJECT COUNTS
*
*
ZONOBJL   EQU     $18B0  ;ZONE OBJECT COUNT (TO 1A00)
*
*
*
*   DISPLAY LIST
*
*
*
DL        EQU     $1A00  ;MARIA DISPLAY LIST
```

```
*
*THE DEFINITIONS OF DLZONE0 ETC. ARE FOR THE MARIA 1 KERNEL'S CONVENIENCE
*

DLZONE0   EQU     DL+$80*0
DLZONE1   EQU     DL+$80*1
DLZONE2   EQU     DL+$80*2
DLZONE3   EQU     DL+$80*3
DLZONE4   EQU     DL+$80*4
DLZONE5   EQU     DL+$80*5
DLZONE6   EQU     DL+$80*6
DLZONE7   EQU     DL+$80*7
DLZONE8   EQU     DL+$80*8
DLZONE9   EQU     DL+$80*9
DLZONE10  EQU     DL+$80*10
DLZONE11  EQU     DL+$80*11
DLZONE12  EQU     DL+$80*12

****************************************
*                                      *
*         ASSEMBLY CONSTANTS           *
*                                      *
****************************************
*

*         CONSTANTS DEALING WITH SCREEN BOUNDARIES
MINX      EQU     02     ;MINIMUM X VALUE TO APPEAR ON SCREEN
MAXX      EQU     159-3  ;MAX X ON SCREEN - FOR UPPER LEFT OF STAMP
MINY      EQU     18     ;MIN Y ON SCREEN
MAXY      EQU     191-3  ;MAX Y ON SCREEN FOR UPPER LEFT OF STAMP
SCENTERX  EQU     80     ;X SCREEN CENTER
SCENTERY  EQU     96     ;Y SCREEN CENTER
SBOXMAXX  EQU     80+30
SBOXMINX  EQU     80-30  ;EDGES OF BOX IN CENTER OF SCREEN
SBOXMAXY  EQU     96+25  ;INSIDE WHICH OBJECTS CAN'T APPEAR
SBOXMINY  EQU     96-25

MCWID     EQU     $05    ;MC WIDTH
MCHEIGHT  EQU     $0B    ;MC HEIGHT

MCXINIT   EQU     SCENTERX-MCWID/2                                  ;STARTING MC X POSITION
MCYINIT   EQU     SCENTERY-MCHEIGHT/2                               ;STARTING MC Y POSITION

MASKL     EQU     00001111B                                         ;SAVE LOWER HALF-BYTE
MASKH     EQU     11110000B                                         ;SAVE UPPER HALF-BYTE
MASK1     EQU     00000001B                                         ;SAVE LOWEST BIT
MASK2     EQU     00000011B                                         ;SAVE LOWER 2 BITS
MASK3     EQU     00000111B                                         ;SAVE LOWER 3 BITS
MASK5     EQU     00011111B                                         ;SAVE LOWER 5 BITS

MCSDELAY  EQU     $04    ;# FRAMES BETWEEN MC SHOTS
SHOTSTX   EQU     $07    ;MC SHOT STEP IN X DIRECTION
SHOTSTY   EQU     $0E    ;MC SHOT STEP IN Y DIRECTION
SHOTWID   EQU     $04    ;MC SHOT WIDTH (EXCEPT VERTICAL SHOTS)
SHOTHT    EQU     $08    ;MC SHOT HEIGHT (EXCEPT HORIZONTAL)

MAXINZON  EQU     20     ;MAXIMUM NUMBER OF OBJECTS IN A ZONE

*         OBJECT CODES:  FOUND IN CRTBL
NULLCODE  EQU     $0
GCODE     EQU     $1
MOCODE    EQU     $2
DCODE     EQU     $3
MICODE    EQU     $4
HCODE     EQU     $5
SCODE     EQU     $6
QCODE     EQU     $7
ECODE     EQU     $8
TCODE     EQU     $9
BCODE     EQU     $A
PCODE     EQU     $B
EMCODE    EQU     $C
CMCODE    EQU     $D
```

```
TMCODE    EQU     $E
MCSCODE   EQU     $F

*         NOTE THAT THERE ARE SPECIAL CODES WHICH WAY BE FOUND IN CRTBL:
*                 $00:  NULL OBJECT
*                 $FF:  END OF OBJECT DATA TABLES

*         NOMINAL OBJECT DIMENSIONS  - USE TO COMPUTE EXTENTS WHEN SETTING UP
GWID      EQU     7
GHEIGHT   EQU     12
FWID      EQU     4
FHEIGHT   EQU     11
HWID      EQU     7
HHEIGHT   EQU     15
SQWID     EQU     9
SQHEIGHT  EQU     9
EWID      EQU     8
EHEIGHT   EQU     8
TWID      EQU     10
THEIGHT   EQU     10
BWID      EQU     9
BHEIGHT   EQU     9

*
*         OBJECT SPEED, STEP AND OTHER CONSTANTS
*           SPEEDS, STEPS, ETC. WHICH DO NOT CHANGE OVER TIME OR FROM
*           RACK TO RACK ARE HERE AS ASSEMBLY CONSTANTS
*                 RAM VARIABLES ARE IN THE ZERO PAGE SECTION
*
GSTEPX    EQU     $2     ;# OF PIXELS IN A HORIZONTAL GRUNT STEP
GSTEPY    EQU     $4     ;# OF PIXELS IN A VERTICAL GRUNT STEP
SQSTEP1   EQU     1      ;STARTING ANIMATION STEP FOR S + Q
SQBIRTHS  EQU     4      ;STARTING NUMBER OF BIRTHS LEFT
MAXSSPD   EQU     $7     ;MAX SPH SPEED AT WAVE START - USED AS
                                                                   :  PARM TO RANDPM
TSTIME    EQU     8      ;BASE # OF FRAMES BETWEEN TANK SHOTS


FDIEWAIT  EQU     60     ;# FRAMES TO KEEP UP A SKULL OR FAMILY
                                                                   ;  SCORE

STARTNUM  EQU     15     ;NUMBER OF BYTES TO GET FROM WAVETBL
                                                                   ; WHEN SETTING UP FOR A NEW
WAVE

WSWAIT    EQU     $13    ;# OF FRAMES AT WAVE START BEFORE ACTION
                                                                   ;  STARTS

STACK     EQU     $FF    ;INITIAL VALUE FOR STACK POINTER

####################################################################################
;NOTES FROM DAN BORIS AND "SCOTTY"; NOT CHECKED BUT ASSUMED TO BE CORRECT.
;  THE ABOVE EQUATES WERE NOT CHECKED 100% EITHER, ALTHOUGH CURSORY CHECK
;  APPEARS SOME LOOKS RIGHT.

;Variables

$40-$43     ; Player 1 score
$44-$47     ; Player 2 score
$48,$4A,$4C ; Player 1 next life score
$49,$4B,$4D ; Player 2 next life score

$59,$5A     ; Pointer to next DLI routine
$5B            ; Number of family on screen (see also $1907 - $1909)

CURPLAY        $61  ;Current Player
CURPLAYERS     $62  ;Current Number of Players
SKILL   $64  ;Skill Level

CURRENTOBJ     $6A      ;Current object being processed's index. Counts up to 84 then
                               ;resets to 1.
NUMPLAYERS  $6C  ;Number of Players at Start
```

```
; These variables are set based on the skill level selected
$6E  ; Skill Variable - this is incremented every time a wave is completed.
       ;  Cannot be more than $73.
$6F  ; Skill Variable
$70  ; Skill Variable
$71  ; Skill Variable
$72  ; Skill Variable
$73  ; Skill Variable - must be max wave that can be reached
$75  ; Skill Variable

$74  ; 1 = Play mode

$7D  ; Number of tank shots on screen

$A0  ; Temp field used in conjunction with $A1 to access function that handles a
         particular enemy type

$AA  temp field used to hold object X when calling a function
$AB  temp field used to hold object X + its width (x extent) when calling a function
$AC  ; temp field used to hold object  Y  when calling a function
$AD  temp field used to hold player Y + its height (y extent) when calling a function
$AE  temp field used as current object index when calling a function.

$B2

$B7  ; Used in collision detection routine, if set to 1 then collision has occurred.
$BE  ; XINTEND in original source. temp field used to hold object X when calling a function
$BF  ; YINTEND in original source. temp field used to hold object  Y  when calling a function
$C0  ; XXINTEND in original source. temp field used to hold object X + its width (x extent)
$C1  ; YYINTEND in original source. temp field used to hold object Y + its height (y extent)

$C2  ;

$C3  ; Used to save random X pos generated by $D3BD and $D3D8 functions
$C4  ; Used to save random Y pos generated by $D3BD  and $D3D8 functions
$C5  ; Used to save a random number generated by $D3A8

$C9  ; Number of enemies currently alive on screen. If you set this to 0 via a machine code
         monitor, you win the level.
$CB  ; A flag, for what I don't know. Is only ever 0 or 1.
$CC  ; Used for grunts when computing move delay. I think this is the minimum number of frames
         to wait - same as in original source
$CD  ; Used for family members when computing move delay.  Strange though, its the Hulk routine
         that sets it (always to 10)
$CE  ; Used for Hulks when computing move delay.  This is the minimum number of moves to make
         before considering a direction change.
$CF  ; Used for Spheroids when computing move delay.  This is the minimum number of moves to
         make before considering a direction change.

$D2  ; Used for Brains when computing move delay.

$D5  ; Level number (1 based)
$D6  ; Sprite index of first Human
$D7  ; Sprite index of first Hulk
$D8  ; Sprite index of first Spheroid
$D9  ; Sprite index of first Quark
$DC  ; Sprite index of first Brain
$DE  ;
$DF  ; Sprite index of first Electrode

$E0  ; Player 1 lives
$E1  ; Player 2 lives
$E2  ; Play State     0 = Normal, 1 = Screen being drawn
$E3  ; 1 = Attract mode
$E8  ; Level player 1
$E9  ; Level player 2

$EC  ; Used to determine when a Quark gives birth to tanks.  If a random number from 0..67
         < the contents of this variable, give birth
$EE  ; I *think* this is number of sparks fired by Enforcers so far.  Limit seems to be 16
$EF  ;

; Number of sprites of each type on current players screen, *to start with*
```

```
$1906 ;Number of grunts
$1907 ;Number of Mommies
$1908 ;Number of Daddies
$1909 ;Number of Mikies
$190A   ; Hulk count
$190B ; Spheroid count
$190C   ; Quarks
$190D ; Enforcers
$190E ; Tanks
$190F   ; Brains
$1914 ; Electrode type for current screen
$1915   ; Number of electrodes

$1916-$1925    ; Inactive player's wave information.  Same purpose as $1906-$1915 but for
                inactive player.  When a player dies and there's a 2 player game,
                1906-1915 is swapped with $1916-$1925 to "swap" player screens.

$1926

In original docs MAXOBJS was 80, in real game it is 87. The following lookup tables are
MAXOBJS bytes in size.  Index 0 in all tables is Player

$1ACF  ; XTBL in original source.  X position of left side of sprite
$1B26  ; YTBL in original source.  Y Position of top of sprite
$1EE3  ; XEXTBL in original source.  Sprite X position of right side of Sprite, also known as
          sprite X extent.
$1F3A  ; YEXTBL in original source.  Sprite Y position of bottom of sprite, also known as
          sprite Y extent.

$1B79  ; SYTBL in original source.  Sprite Y position of shot
$1B7D  ; MTTBL in original source.  Number of frames before sprite can move - see also DTTBL
$1BD4  ; DXTBL in original source.  Delta X to add to current X of an object.
          May be negative.

Note For 4 and 8 directional animations like human, Family members, Hulk, DXTBL is not a delta,
  it's an index.  It will contain a number (0-7) which is an index into XDIRTBL and YDIRTBL.
  DYTBL will be used for something else, e.g. the Brains use it to remember their "target".

$1C27  ; SDIRTBL in original source.  Shot direction,0 = NULL

$1C2B  ; DYTBL in original source.  Delta Y to add to current Y of an object.  Is unioned with
          BRAIN_TARGET_INDEX
$1C2B  ; BRAIN_TARGET_INDEX - for Brain enemy types, records the index of the current target,
          or family member being Prog'd.

$1C82  ; DTTBL in original source.  Direction change counter table. Used by family & hulks &
          Enforcers.  Each byte counter is decremented until 0.  When 0, the Human/Hulk/
          Enforcer changes direction.

$1CD9  ; SATBL in original source.  Step in animation table.
$1D30  ; MISCTBL in original source.  Miscellaneous use. Unioned with BRAIN_FIRE_COUNTDOWN,
          SPARK_LIFE, TANK_SHOT_LIFE.

$1D30  ; BRAIN_FIRE_COUNTDOWN - for Brains, counts down to 0.  When 0, a cruise missile
          is fired.
$1D30  ; SPARK_LIFE - for Sparks, counts down to 0.  When 0, Spark fizzles out and dies
$1D30  ; TANK_SHOT_LIFE - for Tank shots, counts down to 0.  When 0, tank shot expires

$1E35  ;
$1E8C  ; Sprite type Table
      0 = player
      1 = Grunt
      2 = Mommy
      3 = Daddy
      4 = Mikie
      5 = Hulk
      6 = Spheroid
      7 = Quark
      8 = Enforcer
      9 = Tank
      A = Brain
      B = Prog
      C = Enforcer shot (spark)
      D = Brain shot (cruise missile)
      E = Tank shot
      F = Electrode style
      10 = Electrode

$1D87,Y  Low byte of sprite data
$1DDE,Y  High byte of sprite data
1FE8   ; I am sure this is used during the sprite drawing. It is accessed at $DF87 just
           after the sprite is "blitted".

$1F91  ; STTBL in original source.  Sprite state Table.  Bit flag - bit 0 set = active,
           bit 1 = dying
; states

; these are read-only lookup tables
$EBE5 - Jump table for handlers for sprite types.  Use (sprite type index - 1) to calculate
          as player has own routine and is not in jump table
$EC1D - XDIRTBL in source.  X direction table - ie. - how much to add to X component when
          object moves.
$EC25 - YDIRTBL in source ie. - how much to add to Y component when object moves
$EC2D - QUARKXDIRTBL (not in orig. source) - how much to add to Quark's X component when
          quark moves.
$EC35 - QUARKYDIRTBL (not in orig. source) - how much to add to Quark's Y component when
          quark moves.
$EC3D - XDIRTBL4 in source.  X Direction table
$EC4D - YDIRTBL4 in source.  Y Direction table
$EC9C - CRUISEXDIRTBL (not in orig. source) - how much to add to cruise missile's X
          component when cruise moves.
$ECA4 - CRUISEYDIRTBL (not in orig. source) - how much to add to cruise missile's Y
          component when cruise moves.
$ED34 - Electrode width table, $1914 is index into lookup
$ED3C - Electrode height table
$EDBC - Hulk related, unsure what
$EE02 - Hulk related, unsure what

;Display memory
$2100 - $2106   ; Player 1 score
$2107 - $210D   ; Player 2 score
$210E - $2114   ; Lives display Player 1
$2115 - $211B   ; Lives display Player 2
$211C - $211D   ; Level player 1
$2123 - $2124   ; Level player 2
######################################################################################
BEGIN CARTRIDGE ($8000-$FFFF)

;GRAPHICS DATA ($8000-$8FFF)
8000                     .BYTE $BF,$FF,$FF,$80,$2F,$80,$02,$F8,$0B,$FF,$FF,$FE,$02,$F8,$00,$2F
8010                     .BYTE $80,$00,$0B,$F8,$00,$0B,$FF,$FF,$F8,$02,$F8,$00,$2F,$80,$BF,$80
8020                     .BYTE $0B,$E0,$00,$00,$2A,$00,$02,$80,$0A,$AA,$AA,$A0,$02,$AA,$AA,$A8
8030                     .BYTE $00,$AA,$AA,$AA,$00,$00,$02,$A0,$00,$02,$A0,$00,$28,$00,$AA,$AA
8040                     .BYTE $AA,$00,$2A,$00,$02,$80,$00,$00,$0F,$FF,$C0,$FA,$BC,$0E,$AC,$AA
8050                     .BYTE $EA,$8E,$AF,$FA,$AF,$3E,$AB,$C0,$00,$00,$00,$00,$00,$00,$00,$00
8060                     .BYTE $0F,$AA,$EA,$BC,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8070                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8080                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8090                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
80A0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
80B0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
80C0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
80D0                     .BYTE $00,$00,$00,$00,$00,$00,$3E,$94,$16,$BC,$00,$00,$00,$00,$00,$00
80E0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
80F0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8100                     .BYTE $BF,$FF,$FF,$80,$2F,$80,$02,$F8,$0B,$FF,$FF,$FA,$02,$F8,$00,$2F
8110                     .BYTE $80,$00,$0B,$F8,$00,$0B,$FF,$FF,$F8,$02,$F8,$00,$2F,$80,$BF,$80
8120                     .BYTE $0B,$E0,$00,$00,$AE,$80,$0B,$A0,$2B,$FF,$FF,$E8,$0A,$FF,$FF,$FA
8130                     .BYTE $02,$BF,$FF,$FE,$80,$00,$0A,$E8,$00,$0A,$E0,$00,$BA,$02,$BF,$FF
8140                     .BYTE $FE,$80,$AE,$80,$0B,$A0,$00,$00,$0F,$AA,$FF,$FA,$BC,$0E,$AC,$AA
8150                     .BYTE $EA,$8E,$AF,$3E,$AB,$FE,$AB,$C0,$00,$00,$00,$00,$00,$00,$00,$00
8160                     .BYTE $3E,$AA,$EA,$AF,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8170                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8180                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8190                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
```

```
81A0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
81B0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
81C0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
81D0                     .BYTE $00,$00,$00,$00,$00,$00,$3E,$94,$16,$BC,$FF,$FF,$55,$55,$00,$00
81E0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
81F0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8200                     .BYTE $BF,$FF,$FF,$80,$2F,$80,$02,$F8,$0B,$FF,$FF,$F8,$02,$F8,$00,$2F
8210                     .BYTE $80,$00,$0B,$E8,$00,$0B,$FF,$FF,$F8,$02,$F8,$00,$2F,$80,$BE,$00
8220                     .BYTE $0B,$E0,$00,$00,$BF,$80,$0B,$E0,$2F,$FF,$FF,$F8,$0B,$FF,$FF,$FE
8230                     .BYTE $02,$FF,$FF,$FF,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$FF,$FF
8240                     .BYTE $FF,$80,$BF,$80,$0B,$E0,$00,$00,$03,$EA,$AA,$AA,$BC,$0E,$AB,$AA
8250                     .BYTE $EA,$BA,$AF,$0F,$AA,$FE,$AB,$C0,$00,$00,$00,$00,$00,$00,$00,$00
8260                     .BYTE $FA,$AA,$EA,$AB,$C0,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8270                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8280                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8290                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
82A0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
82B0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
82C0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
82D0                     .BYTE $00,$00,$00,$00,$00,$00,$3E,$94,$16,$BC,$FF,$FF,$55,$55,$00,$00
82E0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
82F0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8300                     .BYTE $BF,$AA,$BE,$00,$2F,$80,$02,$F8,$0B,$FA,$AB,$E8,$02,$F8,$00,$2F
8310                     .BYTE $80,$00,$0B,$E0,$00,$0B,$FA,$AB,$E0,$02,$F8,$00,$2F,$80,$BE,$00
8320                     .BYTE $0B,$E0,$00,$00,$BF,$80,$0B,$E0,$2F,$FF,$FF,$F8,$0B,$FF,$FF,$FE
8330                     .BYTE $02,$FF,$FF,$FF,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$FF,$FF
8340                     .BYTE $FF,$80,$BF,$80,$0B,$E0,$00,$00,$00,$FA,$AA,$AA,$BC,$0E,$AB,$AA
8350                     .BYTE $EA,$BA,$AF,$03,$EA,$AA,$AB,$C0,$00,$00,$00,$00,$00,$00,$00,$03
8360                     .BYTE $EA,$AA,$EA,$AA,$F0,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8370                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8380                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8390                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
83A0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
83B0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
83C0                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
83D0                     .BYTE $00,$00,$00,$00,$00,$00,$3E,$94,$16,$BC,$AA,$AA,$AA,$AA,$00,$00
83E0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
83F0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8400                     .BYTE $BE,$80,$BE,$00,$2F,$80,$02,$F8,$0B,$F8,$0B,$E0,$02,$F8,$00,$2F
8410                     .BYTE $80,$00,$0B,$E0,$00,$0B,$E8,$0B,$E0,$02,$F8,$00,$2F,$80,$BE,$00
8420                     .BYTE $0B,$E0,$02,$A8,$BF,$80,$0B,$E0,$2F,$EA,$AB,$F8,$0B,$FA,$AA,$BE
8430                     .BYTE $02,$FE,$AA,$BF,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$FE,$AA
8440                     .BYTE $BF,$80,$BF,$80,$0B,$E0,$00,$00,$00,$3E,$AA,$AA,$BC,$0E,$AA,$AA
8450                     .BYTE $EA,$AA,$AF,$00,$FA,$AA,$AB,$C0,$00,$00,$FF,$FF,$FF,$FF,$FF,$FF
8460                     .BYTE $AA,$AA,$EA,$AA,$BC,$00,$00,$3F,$FF,$C0,$00,$00,$00,$00,$00,$55
8470                     .BYTE $44,$14,$40,$45,$50,$05,$50,$10,$15,$45,$04,$F0,$0F,$F3,$0F,$03
8480                     .BYTE $03,$FC,$F0,$C0,$FC,$FC,$F3,$3F,$C0,$C0,$CF,$F3,$FC,$C3,$CC,$0F
8490                     .BYTE $0F,$F3,$3C,$C3,$CF,$F3,$F0,$30,$CF,$03,$C3,$FC,$CF,$3F,$00,$C0
84A0                     .BYTE $CF,$3C,$CF,$F3,$FC,$FC,$FF,$30,$F3,$C0,$FF,$33,$C3,$C0,$FF,$30
84B0                     .BYTE $CC,$F3,$F3,$F3,$FC,$F3,$3F,$CF,$F0,$3F,$C0,$C3,$F3,$FC,$0C,$00
84C0                     .BYTE $33,$C0,$F0,$CF,$33,$CC,$3F,$C0,$C3,$F3,$FC,$FF,$00,$3F,$F3,$3F
84D0                     .BYTE $3F,$33,$3C,$C3,$CF,$F0,$3E,$94,$16,$BC,$AA,$AA,$AA,$AA,$00,$00
84E0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
84F0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8500                     .BYTE $BE,$00,$BE,$00,$2F,$80,$02,$F8,$0B,$E0,$0B,$E0,$02,$F8,$00,$2F
8510                     .BYTE $80,$00,$0B,$E0,$00,$0B,$E0,$0B,$E0,$02,$F8,$00,$2F,$80,$BE,$00
8520                     .BYTE $0B,$E0,$0A,$FA,$BF,$80,$0B,$E0,$2F,$A0,$0A,$F8,$0B,$F8,$00,$BE
8530                     .BYTE $02,$FA,$00,$AF,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$FA,$00
8540                     .BYTE $AF,$80,$BF,$80,$0B,$E0,$00,$00,$00,$0F,$FF,$FF,$FC,$0E,$AA,$AA
8550                     .BYTE $EA,$AA,$AF,$00,$3E,$AA,$AB,$C0,$00,$00,$FA,$AA,$AA,$AA,$AA,$AE
8560                     .BYTE $AA,$AA,$EA,$AA,$AF,$00,$00,$3E,$AB,$C0,$00,$00,$00,$00,$00,$50
8570                     .BYTE $44,$14,$40,$44,$00,$04,$10,$44,$14,$05,$04,$F0,$0F,$F3,$0F,$03
8580                     .BYTE $03,$C0,$F0,$C0,$30,$C0,$F3,$30,$C3,$30,$CF,$03,$C0,$C3,$CC,$0F
8590                     .BYTE $0F,$03,$3C,$C3,$CF,$03,$CC,$30,$CF,$03,$C3,$C0,$CF,$3C,$C3,$30
85A0                     .BYTE $CF,$3C,$CF,$03,$C0,$F3,$F0,$30,$F3,$C0,$F0,$33,$C3,$C0,$F0,$30
85B0                     .BYTE $CC,$F3,$F3,$F3,$C0,$F3,$30,$CF,$00,$F0,$F0,$C0,$33,$0C,$0C,$00
85C0                     .BYTE $33,$C0,$F0,$CF,$33,$CC,$F0,$F0,$C0,$33,$0C,$C0,$00,$33,$33,$30
85D0                     .BYTE $30,$33,$3C,$C3,$CF,$30,$3E,$94,$16,$BC,$55,$55,$FF,$FF,$00,$00
85E0                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
```

```
85F0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8600                    .BYTE $BE,$00,$BE,$00,$2F,$80,$02,$F8,$0B,$E0,$0B,$E0,$02,$F8,$00,$2F
8610                    .BYTE $80,$00,$0B,$E0,$00,$0B,$E0,$0B,$E0,$02,$F8,$00,$2F,$80,$BE,$00
8620                    .BYTE $0B,$E0,$0B,$FE,$BF,$80,$0B,$E0,$2F,$80,$02,$F8,$0B,$F8,$00,$BE
8630                    .BYTE $02,$F8,$00,$2F,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$F8,$00
8640                    .BYTE $2F,$80,$BF,$80,$0B,$E0,$02,$A8,$00,$00,$00,$00,$00,$0F,$AA,$AA
8650                    .BYTE $EA,$AA,$BC,$00,$0F,$FF,$FF,$C0,$00,$00,$3E,$AA,$AA,$AA,$AA,$AE
8660                    .BYTE $AA,$AA,$EA,$AA,$AF,$00,$00,$3E,$AB,$C0,$00,$00,$00,$00,$00,$50
8670                    .BYTE $44,$14,$40,$44,$00,$04,$11,$41,$14,$05,$04,$F0,$0F,$F3,$0F,$03
8680                    .BYTE $03,$C0,$F0,$C0,$30,$C0,$F3,$30,$CC,$3C,$CF,$03,$C0,$C3,$CC,$0F
8690                    .BYTE $0F,$03,$3C,$C3,$CF,$03,$CC,$30,$CF,$03,$C3,$C0,$CF,$3C,$CC,$3C
86A0                    .BYTE $CF,$3C,$CF,$03,$C0,$F3,$F0,$30,$F3,$C0,$F0,$33,$C3,$C0,$F0,$30
86B0                    .BYTE $CC,$F3,$03,$03,$C0,$F3,$30,$CF,$00,$CF,$30,$C0,$33,$0C,$0C,$00
86C0                    .BYTE $33,$C0,$F0,$CF,$33,$CC,$CF,$30,$C0,$33,$0C,$C0,$00,$33,$33,$30
86D0                    .BYTE $30,$33,$3C,$C3,$C0,$30,$3E,$94,$16,$BC,$55,$55,$FF,$FF,$00,$00
86E0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
86F0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8700                    .BYTE $BE,$00,$BE,$00,$2F,$80,$02,$F8,$0B,$E0,$0B,$E0,$02,$F8,$00,$2F
8710                    .BYTE $80,$00,$0B,$E0,$00,$0B,$E0,$0B,$E0,$02,$F8,$00,$2F,$80,$BE,$00
8720                    .BYTE $0B,$E0,$0B,$FE,$BF,$80,$0B,$E0,$2F,$80,$02,$F8,$0B,$F8,$00,$BE
8730                    .BYTE $02,$F8,$00,$2F,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$F8,$00
8740                    .BYTE $2F,$80,$BF,$80,$0B,$E0,$0A,$FA,$00,$00,$00,$00,$00,$03,$EA,$AA
8750                    .BYTE $EA,$AA,$F0,$00,$00,$00,$00,$00,$00,$00,$0F,$AA,$AA,$AA,$AA,$AE
8760                    .BYTE $AB,$AA,$EA,$BA,$AF,$00,$00,$3E,$AB,$C0,$00,$00,$00,$00,$00,$51
8770                    .BYTE $44,$14,$40,$44,$00,$04,$11,$41,$14,$05,$04,$F0,$0C,$03,$0F,$03
8780                    .BYTE $03,$C0,$F0,$C0,$30,$C0,$F3,$30,$CC,$3C,$CF,$03,$C0,$C3,$CC,$0F
8790                    .BYTE $0F,$03,$3C,$C3,$CF,$03,$CC,$30,$CF,$03,$C3,$C0,$CF,$3C,$CC,$3C
87A0                    .BYTE $CF,$3C,$CF,$03,$C0,$F3,$F0,$0C,$C3,$C0,$F0,$33,$C3,$C0,$F0,$30
87B0                    .BYTE $CC,$F3,$03,$03,$C0,$F3,$33,$CF,$00,$CC,$30,$C0,$33,$0C,$0C,$00
87C0                    .BYTE $33,$C0,$F0,$CF,$33,$CC,$CC,$30,$C0,$33,$0C,$C0,$00,$30,$33,$30
87D0                    .BYTE $30,$33,$3C,$C3,$C0,$30,$3E,$94,$16,$BC,$00,$00,$00,$00,$00,$00
87E0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
87F0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8800                    .BYTE $BE,$00,$BE,$00,$2F,$80,$0B,$F8,$0B,$E0,$0B,$E0,$02,$F8,$00,$BF
8810                    .BYTE $80,$00,$0B,$E0,$00,$0B,$E0,$0B,$E0,$02,$F8,$00,$BF,$80,$BE,$00
8820                    .BYTE $0B,$E0,$0B,$FE,$BF,$80,$0B,$E0,$2F,$80,$02,$F8,$0B,$F8,$00,$BE
8830                    .BYTE $02,$F8,$00,$2F,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$F8,$00
8840                    .BYTE $2F,$80,$BF,$80,$0B,$E0,$0B,$FE,$00,$00,$00,$00,$00,$00,$FA,$AA
8850                    .BYTE $EA,$AB,$C0,$00,$00,$00,$00,$00,$00,$00,$03,$EA,$BF,$FF,$FE,$AE
8860                    .BYTE $AB,$AA,$EA,$BA,$AF,$00,$00,$3E,$AB,$C0,$00,$00,$00,$00,$00,$50
8870                    .BYTE $05,$54,$40,$45,$50,$04,$11,$41,$15,$45,$54,$FF,$CC,$03,$FF,$3F
8880                    .BYTE $F3,$FC,$FF,$C0,$30,$FF,$C3,$30,$CC,$3C,$CF,$03,$FC,$C3,$0C,$0F
8890                    .BYTE $0F,$F3,$FC,$C0,$CF,$F3,$CC,$30,$FF,$03,$C3,$FC,$FF,$3C,$CC,$3C
88A0                    .BYTE $FF,$30,$CF,$03,$FC,$F3,$FF,$03,$03,$FC,$FF,$3F,$C3,$C0,$F0,$3F
88B0                    .BYTE $CF,$F3,$03,$03,$FC,$C3,$30,$0F,$F0,$CC,$30,$CF,$F3,$FC,$FF,$00
88C0                    .BYTE $3F,$C0,$F0,$FF,$3F,$CC,$CC,$30,$CF,$F3,$FC,$FF,$00,$30,$F3,$30
88D0                    .BYTE $30,$33,$FC,$C0,$CF,$F0,$16,$BC,$3E,$94,$00,$00,$00,$00,$00,$00
88E0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
88F0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8900                    .BYTE $BE,$00,$BE,$00,$2F,$80,$2F,$F8,$0B,$E0,$0B,$E0,$02,$F8,$02,$FF
8910                    .BYTE $80,$00,$0B,$E0,$00,$0B,$E0,$0B,$E0,$02,$F8,$02,$FF,$80,$BE,$00
8920                    .BYTE $0B,$E0,$0A,$FA,$BF,$80,$0B,$E0,$2F,$80,$02,$F8,$0B,$F8,$00,$BE
8930                    .BYTE $02,$F8,$00,$2F,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$F8,$00
8940                    .BYTE $2F,$80,$BF,$80,$0B,$E0,$0B,$FE,$00,$00,$00,$00,$00,$00,$3E,$AA
8950                    .BYTE $EA,$AF,$00,$00,$00,$00,$00,$00,$00,$00,$00,$FA,$AF,$00,$0F,$FE
8960                    .BYTE $AC,$AA,$EA,$8E,$AF,$00,$00,$3E,$AB,$C0,$00,$00,$00,$00,$00,$51
8970                    .BYTE $44,$14,$44,$44,$00,$04,$51,$41,$10,$05,$10,$F0,$CC,$03,$0F,$3C
8980                    .BYTE $33,$00,$C3,$00,$30,$03,$C3,$33,$CC,$3C,$CF,$03,$00,$C3,$0C,$0C
8990                    .BYTE $0C,$03,$30,$CC,$CC,$03,$CC,$30,$CF,$03,$03,$00,$CF,$3C,$CC,$3C
89A0                    .BYTE $CF,$30,$CF,$03,$00,$F3,$C0,$0C,$C3,$0C,$C0,$33,$03,$00,$F0,$30
89B0                    .BYTE $CC,$F3,$03,$03,$00,$C3,$30,$0C,$00,$CF,$30,$CC,$33,$0C,$CC,$00
89C0                    .BYTE $33,$C0,$C0,$CF,$33,$0C,$CF,$30,$CC,$33,$0C,$03,$00,$30,$F3,$30
89D0                    .BYTE $30,$33,$3C,$CC,$CC,$00,$16,$BF,$FE,$94,$15,$55,$55,$54,$00,$00
89E0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
89F0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8A00                    .BYTE $BE,$00,$BE,$00,$2F,$A0,$2F,$F8,$0B,$E0,$0B,$E0,$02,$FA,$02,$FF
8A10                    .BYTE $80,$00,$0B,$E0,$00,$0B,$E0,$0B,$E0,$02,$FA,$02,$FF,$80,$BE,$80
8A20                    .BYTE $2B,$E0,$02,$A8,$BF,$80,$0B,$E0,$2F,$80,$02,$F8,$0B,$F8,$00,$BE
```

```
8A30                        .BYTE $02,$F8,$00,$2F,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$F8,$00
8A40                        .BYTE $2F,$80,$BF,$80,$0B,$E0,$0B,$FE,$00,$00,$00,$00,$00,$00,$0F,$AA
8A50                        .BYTE $EA,$BC,$00,$00,$00,$00,$00,$00,$00,$00,$00,$3E,$AB,$FF,$FC,$0E
8A60                        .BYTE $AC,$AA,$EA,$8E,$AF,$FF,$FF,$FE,$AB,$FF,$FF,$C0,$00,$00,$00,$51
8A70                        .BYTE $44,$14,$44,$44,$00,$04,$51,$41,$10,$05,$10,$F0,$CC,$03,$0F,$3C
8A80                        .BYTE $33,$00,$C3,$00,$30,$03,$C3,$33,$CC,$3C,$CF,$03,$00,$C3,$0C,$0C
8A90                        .BYTE $0C,$03,$30,$CC,$CC,$03,$CC,$30,$CF,$03,$03,$00,$CF,$3C,$CC,$3C
8AA0                        .BYTE $CF,$30,$CF,$03,$00,$F3,$C0,$3C,$33,$0C,$C0,$33,$03,$00,$F0,$30
8AB0                        .BYTE $CC,$F3,$03,$03,$00,$C3,$30,$CC,$00,$F0,$F0,$CC,$33,$0C,$CC,$00
8AC0                        .BYTE $33,$C0,$C0,$CF,$33,$0C,$F0,$F0,$CC,$33,$0C,$03,$00,$30,$F3,$30
8AD0                        .BYTE $30,$33,$3C,$CC,$CC,$00,$16,$BF,$FE,$94,$15,$55,$55,$54,$00,$00
8AE0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
8AF0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8B00                        .BYTE $BE,$AA,$BE,$00,$2F,$EA,$BF,$F8,$0B,$EA,$AB,$E0,$02,$FE,$AB,$FF
8B10                        .BYTE $80,$2A,$AB,$EA,$A8,$0B,$EA,$AB,$E0,$02,$FE,$AB,$FF,$80,$BF,$AA
8B20                        .BYTE $AF,$E0,$00,$00,$BF,$80,$0B,$E0,$2F,$80,$02,$F8,$0B,$F8,$00,$BE
8B30                        .BYTE $02,$F8,$00,$2F,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$F8,$00
8B40                        .BYTE $2F,$80,$BF,$80,$0B,$E0,$0A,$FA,$00,$00,$00,$00,$00,$00,$03,$EA
8B50                        .BYTE $EA,$F0,$00,$00,$00,$00,$00,$00,$00,$00,$00,$0F,$AA,$AA,$BC,$0E
8B60                        .BYTE $AC,$AA,$FA,$8E,$AF,$AA,$AA,$AA,$AA,$AA,$AB,$C0,$00,$FC,$00,$55
8B70                        .BYTE $45,$54,$55,$45,$50,$05,$51,$41,$15,$45,$50,$FF,$CC,$03,$FF,$3C
8B80                        .BYTE $33,$FC,$FF,$00,$F0,$3F,$FC,$3F,$CC,$3C,$CF,$F3,$FC,$C3,$F0,$FF
8B90                        .BYTE $CF,$F3,$F0,$FF,$CF,$F3,$F0,$30,$FF,$3F,$F3,$FC,$FF,$3F,$0C,$3C
8BA0                        .BYTE $FF,$3F,$0F,$F3,$FC,$FC,$FF,$3C,$33,$FC,$FF,$3F,$3F,$F0,$FF,$30
8BB0                        .BYTE $CF,$F3,$03,$03,$FC,$FC,$3F,$CF,$F0,$3F,$C0,$CF,$F3,$FC,$C0,$00
8BC0                        .BYTE $3F,$CF,$FC,$FF,$3F,$0C,$3F,$C0,$CF,$F3,$FC,$FF,$00,$30,$F3,$30
8BD0                        .BYTE $30,$33,$FC,$FF,$CF,$F0,$16,$AA,$AA,$94,$16,$AA,$AA,$94,$00,$00
8BE0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
8BF0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8C00                        .BYTE $BF,$FF,$FE,$00,$2F,$FF,$FF,$F8,$0B,$FF,$FF,$E0,$02,$FF,$FF,$FF
8C10                        .BYTE $80,$AF,$FF,$FF,$FA,$0B,$FF,$FF,$E0,$02,$FF,$FF,$FE,$80,$BF,$FF
8C20                        .BYTE $FF,$E0,$00,$00,$BF,$80,$0B,$E0,$2F,$80,$02,$F8,$0B,$F8,$00,$BE
8C30                        .BYTE $02,$F8,$00,$2F,$80,$00,$0B,$F8,$00,$0B,$F8,$00,$BE,$02,$F8,$00
8C40                        .BYTE $2F,$80,$BF,$80,$0B,$E0,$02,$A8,$00,$00,$00,$00,$00,$00,$00,$FA
8C50                        .BYTE $EB,$C0,$00,$00,$00,$00,$00,$00,$00,$00,$00,$03,$EA,$AA,$BC,$0E
8C60                        .BYTE $AC,$AA,$FA,$AE,$BF,$AA,$AA,$AA,$AA,$AA,$AF,$00,$03,$FF,$00,$00
8C70                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8C80                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8C90                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8CA0                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8CB0                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8CC0                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8CD0                        .BYTE $00,$00,$00,$00,$00,$00,$16,$AA,$AA,$94,$16,$AA,$AA,$94,$00,$00
8CE0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
8CF0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8D00                        .BYTE $BF,$FF,$FE,$00,$2F,$FF,$FF,$F8,$0B,$FF,$FF,$E0,$02,$FF,$FF,$FF
8D10                        .BYTE $80,$BF,$FF,$FF,$FE,$0B,$FF,$FF,$E0,$02,$FF,$FF,$FE,$80,$BF,$FF
8D20                        .BYTE $FF,$E0,$00,$00,$BF,$80,$2B,$E0,$2F,$80,$02,$F8,$0B,$F8,$02,$BE
8D30                        .BYTE $02,$F8,$00,$2F,$80,$00,$0B,$F8,$00,$0B,$F8,$02,$BE,$02,$F8,$00
8D40                        .BYTE $2F,$80,$BF,$80,$0B,$E0,$00,$00,$00,$00,$00,$00,$00,$00,$00,$3E
8D50                        .BYTE $EF,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$FA,$AA,$BC,$0E
8D60                        .BYTE $AC,$AA,$FE,$AA,$BF,$AA,$AA,$AA,$AA,$AA,$AF,$00,$0F,$BB,$C0,$00
8D70                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8D80                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8D90                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8DA0                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8DB0                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8DC0                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8DD0                        .BYTE $00,$00,$00,$00,$00,$00,$15,$55,$55,$54,$16,$BF,$FE,$94,$00,$00
8DE0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
8DF0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8E00                        .BYTE $AF,$FF,$FA,$00,$2B,$FF,$FF,$E8,$0A,$FF,$FF,$A0,$02,$BF,$FF,$FE
8E10                        .BYTE $80,$AF,$FF,$FF,$FA,$0A,$FF,$FF,$A0,$02,$BF,$FF,$FE,$80,$AF,$FF
8E20                        .BYTE $FF,$A0,$00,$00,$BF,$AA,$AF,$E0,$2F,$80,$02,$F8,$0B,$FA,$AA,$FE
8E30                        .BYTE $02,$F8,$00,$2F,$80,$00,$0B,$F8,$00,$0B,$FA,$AA,$FE,$02,$F8,$00
8E40                        .BYTE $2F,$80,$BF,$80,$0B,$E0,$00,$00,$00,$00,$00,$00,$00,$00,$00,$0F
8E50                        .BYTE $FC,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$3E,$AA,$BC,$0E
8E60                        .BYTE $AC,$AA,$FA,$AE,$AF,$AA,$FF,$FE,$AB,$FA,$BC,$00,$3E,$BA,$F0,$00
8E70                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
```

```
8E80                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8E90                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8EA0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8EB0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8EC0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8ED0                    .BYTE $00,$00,$00,$00,$00,$00,$15,$55,$55,$54,$16,$BF,$FE,$94,$00,$00
8EE0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
8EF0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

8F00                    .BYTE $2A,$AA,$A8,$00,$0A,$AA,$AA,$A0,$02,$AA,$AA,$80,$00,$AA,$AA,$AA
8F10                    .BYTE $00,$2A,$AA,$AA,$A8,$02,$AA,$AA,$80,$00,$AA,$AA,$AA,$00,$2A,$AA
8F20                    .BYTE $AA,$80,$00,$00,$BF,$FF,$FF,$E0,$2F,$80,$02,$F8,$0B,$FF,$FF,$FE
8F30                    .BYTE $02,$F8,$00,$2F,$80,$00,$0B,$F8,$00,$0B,$FF,$FF,$FE,$02,$F8,$00
8F40                    .BYTE $2F,$80,$BF,$80,$0B,$E0,$00,$00,$00,$00,$00,$00,$00,$00,$00,$03
8F50                    .BYTE $F0,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$0F,$FA,$BC,$0E
8F60                    .BYTE $AC,$AA,$FA,$8E,$AF,$EA,$BC,$3E,$AB,$FF,$FC,$00,$FA,$BA,$BC,$00
8F70                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8F80                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8F90                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8FA0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8FB0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8FC0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
8FD0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$16,$BC,$3E,$94,$00,$00
8FE0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
8FF0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

#######################################################################################
;Integrated much of the disassembly from Dan Boris & "Scotty" ($9000-$9CC6).
9000     4C 1D F3       JMP INIT_$F31D

;Start a new game
GAMESTRT:
9003     A5 6C          LDA NUMPLAYERS                              ;Copy number of players
9005     85 62          STA CURPLAYERS                              ;
9007     A2 FF          LDX $FF                                     ;Reset stack pointer
9009     9A             TXS                                         ;
900A     20 56 E3       JSR $E356                                   ;Turn off sound
900D     20 30 D5       JSR $D530                                   ;Setup hardware
9010     20 86 DC       JSR $DC86                                   ;Setup display list
9013     20 20 D4       JSR RESETSC_$D420                           ;Init scoring
9016     20 C3 91       JSR $91C3                                   ;Setup skill level data
9019     A5 E3          LDA $E3
901B     D0 03          BNE $9020
901D     20 BA D6       JSR MCSEND_$D6BA
9020     20 CD D6       JSR $D6CD
9023     A9 00          LDA #$00
9025     85 E1          STA $E1
9027     85 4F          STA $4F
9029     85 E9          STA $E9
902B     85 5C          STA $5C
902D     85 5E          STA $5E
902F     A9 00          LDA #$00
9031     85 67          STA $67
9033     A9 0F          LDA #$0F
9035     85 65          STA $65
9037     A5 E3          LDA $E3
9039     D0 04          BNE $903F
903B     A9 36          LDA #$36
903D     85 66          STA $66
903F     AD 82 02       LDA $0282                                   ;Read console switches
9042     85 5D          STA $5D                                     ;Save
9044     A5 75          LDA $75                                     ;Player 1 starts with 5 lives.

9046     85 E0          STA $E0                                     ;
9048     A9 01          LDA #$01
904A     85 4E          STA $4E
904C     85 E8          STA $E8                                     ;Set player 1 level to 1
904E     85 D5          STA $D5
9050     A9 00          LDA #$00
9052     85 61          STA $61
9054     20 55 D2       JSR $D255
9057     20 26 D3       JSR $D326
905A     20 E4 D2       JSR $D2E4
```

```
905D     A5 62         LDA CURPLAYERS
905F     F0 1B         BEQ $907C
9061     A5 75         LDA $75                                     ;Player 2 starts with 5 lives.
9063     85 E1         STA $E1                                     ;
9065     A9 01         LDA #$01
9067     85 4F         STA $4F
9069     85 E9         STA $E9                                     ;Set player 2 level to 1
906B     A9 01         LDA #$01
906D     85 61         STA $61
906F     20 55 D2      JSR $D255
9072     20 26 D3      JSR $D326
9075     20 E4 D2      JSR $D2E4
9078     A9 00         LDA #$00
907A     85 61         STA $61

;Initialize number of each sprite on first screen

907C     A9 1A         LDA #$1A                                    ;Set pointer to $EF1A, which
is level 1's data
907E     85 A0         STA TEMP0                                   ;
9080     A9 EF         LDA #$EF                                    ;
9082     85 A1         STA TEMP1                                   ;
9084     A0 10         LDY $10                                     ;Copy 16 bytes
9086     B1 A0         LDA (TEMP0),Y                               ;
9088     99 06 19      STA OBJS_PER_LEVEL_$1906,Y                  ;Into $1906
908B     99 16 19      STA OBJS_PER_WAVE_OTHER_PLAYER_$1916,Y      ;into $1916
908E     88            DEY                                         ;
908F     10 F5         BPL $9086                                   ;

9091     A5 E5         LDA $E5
9093     45 C6         EOR $C6
9095     85 C5         STA $C5
9097     A5 62         LDA CURPLAYERS
9099     D0 07         BNE $90A2
909B     A9 00         LDA #$00
909D     85 61         STA $61
909F     4C A6 90      JMP $90A6
90A2     A9 01         LDA #$01
90A4     85 61         STA $61
90A6     20 D6 D4      JSR $D4D6

90A9     A6 61         LDX $61                                     ;Get player number
90AB     B5 E0         LDA $E0,X                                   ;Get number of lives
90AD     10 46         BPL $90F5                                   ;> -1
90AF     20 D6 D4      JSR $D4D6
90B2     A6 61         LDX $61
90B4     B5 E0         LDA $E0,X
90B6     10 3D         BPL $90F5
90B8     20 5A DD      JSR $DD5A
90BB     A5 E3         LDA $E3
90BD     D0 30         BNE $90EF
90BF     A9 01         LDA #$01
90C1     85 67         STA $67
90C3     20 70 F5      JSR $F570
90C6     A9 00         LDA #$00
90C8     85 E5         STA $E5
90CA     85 E6         STA $E6

90CC     20 34 DB      JSR $DB34
90CF     A5 0C         LDA $0C
90D1     10 04         BPL $90D7
90D3     A5 0D         LDA $0D
90D5     30 03         BMI $90DA
90D7     4C 03 90      JMP GAMESTRT_$9003

90DA     20 38 F5      JSR $F538
90DD     20 5D D3      JSR $D35D
90E0     A5 E5         LDA $E5
90E2     C9 F0         CMP #$F0
90E4     90 E6         BCC $90CC
90E6     20 52 F6      JSR $F652
90E9     20 45 F5      JSR $F545
90EC     20 5A DD      JSR $DD5A
```

```
90EF     20 91 D6       JSR $D691
90F2     4C 95 F3       JMP $F395

90F5     20 5A DD       JSR $DD5A
90F8     20 55 D2       JSR $D255
90FB     20 E4 D2       JSR $D2E4
90FE     A5 62          LDA CURPLAYERS
9100     F0 1D          BEQ $911F
9102     20 8A F5       JSR $F58A
9105     A9 40          LDA #$40
9107     85 CA          STA $CA
9109     85 67          STA $67
910B     20 38 F5       JSR $F538
910E     20 5D D3       JSR $D35D
9111     20 34 DB       JSR $DB34
9114     A5 CA          LDA $CA
9116     D0 F3          BNE $910B
9118     A9 00          LDA #$00
911A     85 67          STA $67
911C     20 5A DD       JSR $DD5A
911F     20 26 D3       JSR $D326
9122     20 30 D6       JSR $D630
9125     4C 54 B0       JMP $B054
9128     A5 E4          LDA $E4
912A     D0 FC          BNE $9128
912C     A5 E4          LDA $E4
912E     F0 FC          BEQ $912C
9130     20 48 92       JSR $9248
9133     20 56 E3       JSR $E356
9136     20 30 D6       JSR $D630
9139     20 55 91       JSR $9155
913C     20 34 DB       JSR $DB34
913F     20 5D D3       JSR $D35D
9142     20 33 D7       JSR $D733
9145     20 9E D7       JSR $D79E
9148     20 E3 91       JSR $91E3
914B     A5 C9          LDA CRELEFT
914D     D0 03          BNE $9152
914F     4C 00 D0       JMP $D000
9152     4C 39 91       JMP $9139
9155     A5 CA          LDA $CA
9157     29 7F          AND #$7F
9159     D0 12          BNE $916D
915B     A9 00          LDA #$00
915D     85 CB          STA $CB
915F     A5 CC          LDA $CC
9161     C9 04          CMP #$04
9163     90 02          BCC $9167
9165     C6 CC          DEC $CC
9167     A5 EB          LDA $EB
9169     D0 02          BNE $916D
916B     E6 EC          INC $EC
916D     A5 CA          LDA $CA
916F     29 3F          AND #$3F
9171     D0 08          BNE $917B
9173     A5 EB          LDA $EB
9175     F0 04          BEQ $917B
9177     C6 EB          DEC $EB
9179     10 47          BPL $91C2
917B     A5 EB          LDA $EB
917D     D0 43          BNE $91C2
917F     A9 06          LDA #$06
9181     85 EA          STA $EA
9183     0A             ASL A
9184     85 60          STA $60
9186     A5 ED          LDA $ED
9188     CD 0D 19       CMP $190D
918B     B0 35          BCS $91C2
918D     AD 0B 19       LDA $190B                                     ;Get Spheroid count
9190     F0 30          BEQ $91C2                                     ;Exit if zero (goes directly
to rts)
9192     85 A3          STA TEMP3                                     ;Store in zero page to save
cycles
```

```
9194      A6 D8          LDX $D8                                      ;Get index of Spheroids to add
to $1E8C (sprite type table)
9196      BD 8C 1E       LDA SPRITE_TYPE_$1E8C,X                      ;Get sprite type
9199      29 1F          AND #$1F                                     ;Mask it off
919B      C9 06          CMP #$06                                     ;Spheroid?
919D      D0 1E          BNE $91BD                                    ;Branch if not
919F      BD 91 1F       LDA SPRITE_STATE_$1F91,X                     ;Sprite state table
91A2      C9 01          CMP #$01                                     ;Is it enabled?
91A4      D0 17          BNE $91BD                                    ;Branch if it isnt'
91A6      BD 30 1D       LDA MISCTBL_$1D30,X                          ;Get sprite image for Spheroid
91A9      F0 12          BEQ $91BD                                    ;
91AB      20 A8 D3       JSR RANDOM_$D3A8                             ;Get a random number into $C5
and accumulator
91AE      C5 EC          CMP $EC                                      ;
91B0      B0 0B          BCS $91BD                                    ; if more than 236
91B2      20 77 BA       JSR $BA77
91B5      30 0B          BMI $91C2
91B7      DE 30 1D       DEC MISCTBL_$1D30,X
91BA      4C C2 91       JMP $91C2
91BD      E8             INX
91BE      C6 A3          DEC TEMP3                                    ;Dec temp Spheroid count
91C0      D0 D4          BNE $9196
91C2      60             RTS
;
;Setup skill level data
;
91C3      A4 64          LDY SKILL                                    ;Get skill level
91C5      B9 62 ED       LDA $ED62,Y
91C8      85 72          STA $72
91CA      B9 67 ED       LDA $ED67,Y
91CD      85 71          STA $71
91CF      B9 6C ED       LDA $ED6C,Y
91D2      85 6F          STA $6F
91D4      85 70          STA $70
91D6      85 6E          STA $6E
91D8      18             CLC
91D9      69 28          ADC #$28
91DB      85 73          STA $73
91DD      B9 71 ED       LDA $ED71,Y
91E0      85 75          STA $75
91E2      60             RTS

91E3      A6 55          LDX $55                                      ;Sprite number
91E5      E0 53          CPX #$53                                     ;Last sprite?
91E7      B0 1E          BCS $9207                                    ;Branch if it is
91E9      BD 8C 1E       LDA SPRITE_TYPE_$1E8C,X                      ;Get sprite type
91EC      F0 0E          BEQ $91FC                                    ;Branch if it the player
91EE      C5 76          CMP $76                                      ;
91F0      F0 0A          BEQ $91FC
91F2      BD E8 1F       LDA $1FE8,X
91F5      F0 05          BEQ $91FC
91F7      DE 7D 1B       DEC MTTBL,X
91FA      30 21          BMI $921D

OBJCONT:
91FC      A5 54          LDA $54
91FE      D0 13          BNE $9213
9200      A5 E4          LDA $E4
9202      F0 0F          BEQ $9213
9204      E8             INX                                          ;Next sprite
9205      D0 02          BNE $9209
9207      A2 01          LDX $01
9209      E4 55          CPX $55
920B      D0 D8          BNE $91E5
920D      A5 E5          LDA $E5
920F      C5 E7          CMP $E7
9211      F0 FA          BEQ $920D
9213      86 55          STX $55
9215      A5 E5          LDA $E5
9217      85 E7          STA $E7
9219      20 AA DE       JSR $DEAA
921C      60             RTS
```

```
*
*********************************************
*                                           *
*         MOVE    -- CALLED BY CHKOBJ       *
*                                           *
*********************************************
*
MOVE:                                                            ;X IS OBJECT INDEX
921D     BD 8C 1E      LDA SPRITE_TYPE_$1E8C,X                   ;Get sprite type
9220     29 1F         AND #$1F                                  ;Mask off type

*         WE NOW HAVE A NUMBER FROM 1 TO $E DENOTING OBJECT TYPE

9222     38            SEC                                       ;
9223     E9 01         SBC #$01                                  ;Subtract one to bypass player
type
9225     0A            ASL A                                     ;MULTIPLY BY 2, HAVE 0 TO $1A
9226     A8            TAY                                       ;
9227     B9 E5 EB      LDA $EBE5,Y                               ;LOAD TEMP0 AND TEMP1
922A     85 A0         STA TEMP0                                 ; WITH ADDRESS OF OBJECT
HANDLING
922C     B9 E6 EB      LDA $EBE6,Y                               ; ROUTINE LOOKED UP IN MOVTBL
922F     85 A1         STA TEMP1                                 ;JUMP TO CORRECT ROUTINE TO
HANDLE
9231     6C A0 00      JMP ($00A0)                               ; EACH TYPE OF OBJECT
;
; Get free object index for a missile
;
; Expects
; $EF = index
; if $EF is positive (0-127) it is an index for a slot where missiles can be stored
; if $EF is minus (128-255) this function exits
;
; Returns
; A is 0..127 if a free slot has been found, in this case A = object index of free slot
; If A is #$80 this means no free slots available.
; $EF might be changed also
;
GET_MISSILE_SLOT_$9234:
9234     A6 EF         LDX $EF
9236     30 0E         BMI $9246
9238     BD 91 1F      LDA SPRITE_STATE_$1F91,X
923B     F0 04         BEQ $9241                                 ;Dead
923D     A2 80         LDX $80
923F     30 05         BMI $9246
9241     BD 30 1D      LDA MISCTBL_$1D30,X
9244     85 EF         STA $EF
9246     8A            TXA
9247     60            RTS

9248     20 30 D6      JSR $D630
924B     20 E4 D2      JSR $D2E4
924E     A6 61         LDX $61                                   ;Get player number
9250     B5 E8         LDA $E8,X                                 ;Get players current level
9252     29 0F         AND #$0F                                  ;Mask off bottom 4 bits
9254     F0 04         BEQ $925A
9256     C9 05         CMP #$05
9258     D0 08         BNE $9262

925A     20 56 E3      JSR $E356
925D     A9 14         LDA #$14
925F     20 95 E3      JSR DOTUNE_$E395
9262     A9 00         LDA #$00
9264     85 79         STA $79
9266     85 78         STA $78
9268     85 77         STA $77
926A     85 55         STA $55
926C     A4 79         LDY $79
926E     A9 01         LDA #$01                                  ;Flag screen as being drawn
9270     85 E2         STA $E2
9272     B9 44 ED      LDA $ED44,Y
9275     85 76         STA $76
9277     84 79         STY $79
```

```
9279      A9 00          LDA #$00
927B      85 77          STA $77
927D      85 78          STA $78
927F      20 34 DB       JSR $DB34
9282      A5 CA          LDA $CA
9284      29 01          AND #$01
9286      F0 03          BEQ $928B
9288      20 03 DE       JSR $DE03
928B      20 E3 91       JSR $91E3
928E      20 5D D3       JSR $D35D
9291      A5 78          LDA $78
9293      F0 EA          BEQ $927F
9295      A4 79          LDY $79
9297      C8             INY
9298      C0 0A          CPY #$0A
929A      90 D6          BCC $9272
929C      20 D0 92       JSR $92D0
929F      A9 00          LDA #$00
92A1      85 78          STA $78
92A3      85 76          STA $76
92A5      20 AA DE       JSR $DEAA
92A8      20 34 DB       JSR $DB34
92AB      20 11 93       JSR $9311
92AE      20 E3 91       JSR $91E3
92B1      20 5D D3       JSR $D35D
92B4      A5 78          LDA $78
92B6      F0 F0          BEQ $92A8
92B8      A9 01          LDA #$01
92BA      8D 91 1F       STA SPRITE_STATE_$1F91
92BD      A2 00          LDX $00
92BF      20 36 E1       JSR $E136
92C2      20 AA DE       JSR $DEAA
92C5      A5 52          LDA $52
92C7      C5 53          CMP $53
92C9      D0 F7          BNE $92C2
92CB      A9 00          LDA #$00                              ;Screen finished being drawn
92CD      85 E2          STA $E2
92CF      60             RTS

92D0      A2 53          LDX $53
92D2      86 79          STX $79
92D4      A0 03          LDY $03
92D6      B9 4E ED       LDA $ED4E,Y
92D9      9D CF 1A       STA SPRITE_X,X
92DC      18             CLC
92DD      69 0D          ADC #$0D
92DF      9D E3 1E       STA SPRITE_X_EXTENT,X
92E2      B9 52 ED       LDA $ED52,Y
92E5      9D 26 1B       STA SPRITE_Y,X
92E8      18             CLC
92E9      69 0C          ADC #$0C
92EB      9D 3A 1F       STA SPRITE_Y_EXTENT,X
92EE      B9 56 ED       LDA $ED56,Y
92F1      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
92F4      A9 00          LDA #$00
92F6      9D D9 1C       STA SATBL,X
92F9      9D 8C 1E       STA SPRITE_TYPE_$1E8C,X
92FC      A9 01          LDA #$01
92FE      9D 91 1F       STA SPRITE_STATE_$1F91,X
9301      84 77          STY $77
9303      86 79          STX $79
9305      20 36 E1       JSR $E136
9308      A4 77          LDY $77
930A      A6 79          LDX $79
930C      E8             INX
930D      88             DEY
930E      10 C6          BPL $92D6
9310      60             RTS

9311      A2 53          LDX $53
9313      86 79          STX $79
9315      A0 03          LDY $03
9317      BD CF 1A       LDA SPRITE_X,X
```

```
931A     18             CLC
931B     79 5A ED       ADC $ED5A,Y
931E     85 BE          STA XINTEND_BE
9320     9D CF 1A       STA SPRITE_X,X
9323     C0 03          CPY #$03
9325     D0 09          BNE $9330
9327     BD CF 1A       LDA SPRITE_X,X
932A     C9 4B          CMP #$4B
932C     90 34          BCC $9362
932E     F0 32          BEQ $9362
9330     BD E3 1E       LDA SPRITE_X_EXTENT,X
9333     18             CLC
9334     79 5A ED       ADC $ED5A,Y
9337     85 C0          STA XXINTEND_C0
9339     9D E3 1E       STA SPRITE_X_EXTENT,X
933C     BD 26 1B       LDA SPRITE_Y,X
933F     18             CLC
9340     79 5E ED       ADC $ED5E,Y
9343     85 BF          STA YINTEND_BF
9345     9D 26 1B       STA SPRITE_Y,X
9348     BD 3A 1F       LDA SPRITE_Y_EXTENT,X
934B     18             CLC
934C     79 5E ED       ADC $ED5E,Y
934F     85 C1          STA YYINTEND_C1
9351     9D 3A 1F       STA SPRITE_Y_EXTENT,X
9354     84 77          STY $77
9356     86 79          STX $79
9358     20 AF E1       JSR $E1AF
935B     A4 77          LDY $77
935D     A6 79          LDX $79
935F     4C 89 93       JMP $9389
9362     A2 53          LDX $53
9364     A0 03          LDY $03
9366     A9 00          LDA #$00
9368     9D 91 1F       STA SPRITE_STATE_$1F91,X
936B     84 77          STY $77
936D     86 79          STX $79
936F     20 AF E1       JSR $E1AF
9372     A6 79          LDX $79
9374     A4 77          LDY $77
9376     A9 0F          LDA #$0F
9378     9D 8C 1E       STA SPRITE_TYPE_$1E8C,X
937B     A9 00          LDA #$00
937D     9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
9380     E8             INX
9381     88             DEY
9382     10 E2          BPL $9366
9384     A9 01          LDA #$01
9386     85 78          STA $78
9388     60             RTS
9389     E8             INX
938A     88             DEY
938B     30 03          BMI $9390
938D     4C 17 93       JMP $9317
9390     60             RTS

******************************************************
******************************************************
*                                                    *
*                                                    *
*    ROBOTRON     6-JULY-83                          *
*                 5-AUGUST-83            2:40        *
*                24-AUGUST-83            9:15        *
*                                                    *
*                                                    *
*         RSUBR.S    - ROBOTRON SUBROUTINES          *
*                                                    *
******************************************************

******************************************
******************************************
*                                        *
*     SPECIFIC OBJECT MOVE ROUTINES      *
*                                         *
*                                         *
*  ALL MUST END WITH A     JMP OBJCONT    *
*                                         *
*******************************************

*************************************************************************************
*
*         USAGE OF OBJECT DATA TABLES
*                 - STANDARD ENTRIES FOR ALL OBJECTS
*                 ( FOR THE OTHER ENTRIES, SEE INDIVIDUAL MOVE ROUTINES )
*
*         XTBL    - X POSITION
*         YTBL    - Y POSITION
*         XEXTBL  - X EXTENT (ABSOLUTE)
*         YEXTBL  - Y EXTENT (ABSOLUTE)
*         MTTBL   - NUMBER OF FRAMES UNTIL NEXT MOVE
*         DLPHTBL - DISPLAY LIST HIGH BYTE POINTER
*         DLPLTBL - DISPLAY LIST LOW BYTE POINTER
*         DL2PTBL - OFFSET BETWEEN 2 DISPLAY LIST ENTRIES, 0 IF IN 1 ZONE
*         STTBL   - USED BY UNLOADER, LOWER 2 BITS ARE COMMAND TO UNLOADER:
*           LOWER 2 BITS OF STTBL       MEANING
*             00     -   CREATURE IS DEAD AND GONE - UNLOAD IT PERMANENTLY
*                         ( CREATURE CURRENTLY IS IN DL - REMOVE IT FOREVER )
*             01     -   CREATURE IS ALIVE AND MOVING - DO NORMAL LOAD/UNLOAD
*                         WITH COMPLETE ZONE-CROSSING CHECKING
*             10     -   CREATURE IS NEW - IT DOES NOT EXIST IN DL AND
*                         MUST BE ADDED (IN 1 OR 2 ZONES)
*             11     -   CREATURE IS IN THE PROCESS OF DYING - IT DOES NOT MOVE
*                         SO IT IS SAFE TO ASSUME THAT IT DOES NOT CROSS ZONES.
*                         THIS COMMAND MEANS TO LOOK UP NEW STAMP, PALETTE AND
*                         WIDTH DATA, BUT NOT TO CHECK FOR ZONE CROSSINGS.
*                         THE INTEND VARIABLES ARE NOT SET, SO TAKE ALL DATA
*                         NECESSARY FROM THE OBJECT DATA TABLES.  THIS WILL
*                         PROBABLY BE A SIMPLE CALL TO GETSTAMP...
*
*******************

*******************************************
*                                         *
*         GMOV    -- MOVE GRUNTS          *
*                                         *
*******************************************
*
*         USE OF OBJECT DATA TABLE ENTRIES:
*
*         DXTBL   - 0
*         DYTBL   - UNUSED
*         DTTBL   - UNUSED
*         SATBL   - STEP - 0,1,2,3 OR 4 - 9 FOR DYING
*         MISCTBL - UNUSED
*         CRTBL   - 1
*
***********
*
GMOV:
9391     A5 E2          LDA $E2                                 ;Game State
9393     D0 0C          BNE $93A1                               ;Branch if not in play mode
9395     BD 91 1F       LDA SPRITE_STATE_$1F91,X                ;LOAD THIS GRUNT'S CURRENT
STATUS
9398     29 03          AND #MASK2                              ;
939A     D0 08          BNE GMOV01_$93A4                        ;WILL BE 0 IF GRUNT IS DEAD AND
GONE
939C     A9 00          LDA #NULLCODE                           ;STTBL IS 0, SO NULL OUT CRTBL

939E     9D 8C 1E       STA CRTBL,X
93A1     4C FC 91       JMP OBJCONT_$91FC                       ;PROCESS NEXT OBJECT

GMOV01:
93A4     29 02          AND #$02                                ;GET ONLY BIT 1 - GRUNT 'DYING'
FLAG
93A6     F0 03          BEQ $93AB                               ;Grunt is not dying
93A8     4C 48 94       JMP $9448                               ;GRUNT IS DYING - DON'T MOVE
```

```
93AB     AD 15 19     LDA $1915                        ;Get number of Electrodes
(Grunts die with
                                                       ; contact with Electrodes

which die also.)
93AE     F0 35        BEQ $93E5                        ; if 0, skip checking for them
93B0     A4 DF        LDY $DF                          ;Get index of first sprite
after grunts
93B2     B9 8C 1E     LDA CRTBL,Y                      ;Get sprite type
93B5     29 1F        AND #$1F                         ;Mask it
93B7     C9 10        CMP #$10                         ;Is it an Electrode?
93B9     D0 25        BNE $93E0                        ;Branch if it isn't
93BB     B9 91 1F     LDA SPRITE_STATE_$1F91,Y         ;Get sprites state
93BE     29 03        AND #$03                         ;Mask it
93C0     C9 01        CMP #$01                         ;Is it enabled?
93C2     D0 1C        BNE $93E0                        ;Branch if it isn't
93C4     20 5D DC     JSR $DC5D                        ;Check if sprite overlap
93C7     A5 B7        LDA FRMCNT                       ;Get result
93C9     F0 15        BEQ $93E0                        ;Branch if they don't overlap
93CB     84 B5        STY TEMP21                       ;Save Y
93CD     A9 01        LDA #$01                         ;
93CF     20 DB D6     JSR SCORING_$D6DB                ;Score+100 points
93D2     A4 B5        LDY TEMP21                       ;Restore Y
93D4     86 B5        STX TEMP21                       ; Preserve X in temp variable
93D6     98           TYA                              ; A = electrode index
93D7     AA           TAX                              ; X = electrode index
93D8     20 0E D7     JSR $D70E                        ;Mark electrode as dead
93DB     A6 B5        LDX TEMP21                       ;Restore x from temp variable
93DD     4C 48 94     JMP $9448                        ;Kill off grunt also


93E0     C8           INY                              ;Next Electrode to check for
collision
93E1     C4 D6        CPY $D6                          ;
93E3     90 CD        BCC $93B2                        ;Branch if not done with
Electrodes

*          GRUNT IS ALIVE AND WELL

93E5     BD CF 1A     LDA XTBL,X                       ;LOAD GRUNT X POSITION
93E8     CD CF 1A     CMP SPRITE_X                     ;Compare to player's X
93EB     B0 0F        BCS GMOV1_$93FC                  ;If >= player X
93ED     69 02        ADC #GSTEPX                      ;If < player X, add 2 to Grunt
XPos,
93EF     85 BE        STA XINTEND_BE                   ; move Grunt right.
93F1     BD E3 1E     LDA XEXTBL,X                     ;Get Grunt X Extent
93F4     18           CLC                              ;
93F5     69 02        ADC #GSTEPX                      ;Add 2 to that also, so that
Grunt XPos and
                                                       ; Grunt XExtent have been moved

by same amount
93F7     85 C0        STA XXINTEND_C0                  ;STORE FOR LOADER
93F9     4C 08 94     JMP GMOV2_$9408

GMOV1:
93FC     E9 02        SBC #GSTEPX                      ;Subtract 2 from Grunts
current sprite X
93FE     85 BE        STA XINTEND_BE                   ; move Grunt left.
9400     BD E3 1E     LDA SPRITE_X_EXTENT,X            ;Subtract 2 from Grunt X Extent
also
9403     38           SEC
9404     E9 02        SBC #GSTEPX                      ;MOVE THE OTHER EDGE OF IT
9406     85 C0        STA XXINTEND_C0                  ;And store result in $C0

GMOV2:
9408     BD 26 1B     LDA YTBL,X                       ;LOAD GRUNT Y POSITION
940B     CD 26 1B     CMP SPRITE_Y                     ;Compare to player's Y
940E     B0 0F        BCS GMOV3_$941F                  ;>=
9410     69 04        ADC #GSTEPY                      ;Add 4 to Grunt YPos,
9412     85 BF        STA YINTEND_BF                   ; move Grunt down.
9414     BD 3A 1F     LDA SPRITE_Y_EXTENT,X            ;Add 4 to sprite Y extent
9417     18           CLC
9418     69 04        ADC #GSTEPY                      ;MOVE THE OTHER EDGE OF IT
```

```
941A      85 C1          STA YYINTEND_C1                              ;And store in $C1.  All
parameters set up
                                                                      ; for call to D1B4 (in

playfield check)
941C      4C 2B 94       JMP GMOV4_$942B

GMOV3:
941F      E9 04          SBC #GSTEPY                                  ;Subtract 4 from Grunt YPos,
9421      85 BF          STA YINTEND_BF                               ; move Grunt up.
9423      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
9426      38             SEC
9427      E9 04          SBC #GSTEPY                                  ;Subtract 4 from Grunt Y Extent
9429      85 C1          STA YYINTEND_C1                              ; and store in $C1

GMOV4:
942B      20 B4 D1       JSR CHKINTBD_$D1B4                           ;Ensure that sprite is in the
playfield
942E      A9 03          LDA #$03
9430      85 A4          STA MTTBL,X                                  ;RESET # OF FRAMES UNTIL NEXT
MOVE

9432      20 05 D4       JSR RANDPM_$D405                             ;Get a random number and mask
it with #$03
9435      18             CLC
9436      65 CC          ADC $CC                                      ;Add to compute frames to wait
before
                                                                      ; Grunt can move again


9438      9D 7D 1B       STA MTTBL,X
943B      DE D9 1C       DEC SATBL,X                                  ;Next animation
943E      10 3C          BPL GMOV5_$947C
9440      A9 03          LDA #$03                                     ;We've reached the end of the
anim
                                                                      ; sequence, reset anim frame
9442      9D D9 1C       STA SATBL,X                                  ;NEW ANIMATION STEP
9445      4C 7C 94       JMP GMOV5_$947C                              ;OK, NOW TRY TO LOAD THIS GRUNT

GDYING:             ;GRUNT IS DYING - DON'T MOVE IT, USE DIFFERENT ANIMATION
9448      BD D9 1C       LDA SATBL,X                                  ;From original source code see
GDYING
                                                                      ; label.  Grunt is dying,

don't move it
944B      C9 04          CMP #$04                                     ;4th anim frame of death?
944D      10 12          BPL GDYING1_$9461                            ;>=4th

*          THIS GRUNT JUST DIED SINCE ITS LAST MOVE
*          START GRUNT DEATH SOUND

944F      A9 02          LDA #SCREDIE                                 ;Play sound #2 (Generic
Explosion)
9451      20 95 E3       JSR DOTUNE_$E395                             ;Call DoTune
9454      A9 00          LDA #$00                                     ;Dying direction is 0
9456      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
9459      A9 03          LDA #$03                                     ;One before lowest death
animation
945B      9D D9 1C       STA SATBL,X
945E      4C 65 94       JMP GDYING_$9465                             ;Jump to gdying2

GDYING1:
9461      C9 09          CMP #$09                                     ;9th anim frame of death
(highest)
9463      F0 0B          BEQ $9470                                    ;Yes, so really kill this
Grunt.

GDYING2:
9465      FE D9 1C       INC SATBL,X                                  ;This is gdying2
9468      A9 01          LDA #$01                                     ;GSPEED FOR A DYING GRUNT
(shamelessly
                                                                      ; copied from original source

code)
946A      9D 7D 1B       STA MTTBL,X
946D      4C 76 94       JMP $9476
9470      20 1C D7       JSR $D71C                                    ;Call routine that marks
```

```
sprite as
                                                                    ; completely dead - sets sprite
state
                                                                    ;  to 0 and sprite type to 0
9473      4C FC 91       JMP OBJCONT_$91FC                          ;PROCESS NEXT OBJECT
9476      20 AF E1       JSR $E1AF
9479      4C FC 91       JMP OBJCONT_$91FC                          ;PROCESS NEXT OBJECT

GMOV5:
947C      20 AF E1       JSR $E1AF
947F      AD 3E 21       LDA $213E
9482      D0 F8          BNE $947C
9484      A5 BE          LDA XINTEND_BE
9486      9D CF 1A       STA SPRITE_X,X
9489      A5 C0          LDA XXINTEND_C0
948B      9D E3 1E       STA SPRITE_X_EXTENT,X
948E      A5 BF          LDA YINTEND_BF
9490      9D 26 1B       STA SPRITE_Y,X
9493      A5 C1          LDA YYINTEND_C1
9495      9D 3A 1F       STA SPRITE_Y_EXTENT,X
9498      A9 09          LDA #$09
949A      20 45 E4       JSR $E445
949D      4C FC 91       JMP OBJCONT_$91FC                          ;PROCESS NEXT OBJECT
;
; FAMILY AI HANDLER - THIS IS CALLED FOR MIKEY, DADDY, MOMMY
;
94A0      BD 91 1F       LDA SPRITE_STATE_$1F91,X                   ;Read the state of the human
94A3      29 02          AND #$02                                   ;Dying?
94A5      F0 03          BEQ $94AA                                  ;No
94A7      4C 35 95       JMP FDYING_$9535                           ;Human is dying

94AA      A5 E2          LDA $E2                                    ;Read game state
94AC      F0 03          BEQ $94B1                                  ;We're in game mode
94AE      4C EA 94       JMP FOK_$94EA                              ;We're not in game mode, so
don't bother

; check for collisions with Hulk
94B1      AD 0A 19       LDA $190A                                  ;Get Hulk count
94B4      F0 34          BEQ FOK_$94EA                              ;If 0, we've nothing to process
(go to FOK in original source)
94B6      A4 D7          LDY $D7                                    ;Get index of first Hulk

FHCL:
94B8      B9 91 1F       LDA SPRITE_STATE_$1F91,Y                   ;Get it's state
94BB      29 03          AND #$03                                   ;Mask it
94BD      F0 26          BEQ FHCN_$94E5                             ;Branch if it's not active
94BF      20 5D DC       JSR $DC5D                                  ;Check for collision
94C2      A5 B7          LDA FRMCNT                                 ;Get result
94C4      F0 1F          BEQ FHCN_$94E5                             ;Branch if no collision

*          THE FAMILY MEMBER HAS BEEN SEVERELY KILLED
*          ENTER THE FAMILY DYING SOUND INTO THE SOUND QUEUE

94C6      A9 03          LDA #$03                                   ;Play Family Death Sound
94C8      20 95 E3       JSR DOTUNE_$E395

*          NOW SET FAMILY ANIMATION TO #0 (SKULL) WITH HIGH BIT SET.
*          SET THE FAMILY CODE TO BE A MOMMY - #MOCODE
*          ALSO SET THE 'DYING' BIT IN STTBL
*          ALSO SET THE DIRECTION TO 8
*          THEN IMMEDIATELY GO TO THE FAMILY JUST-DYING ROUTINE

94CB      A9 00          LDA #$00
94CD      9D D9 1C       STA SATBL,X                                ;Set animation frame to 0
94D0      A9 02          LDA #$02                                   ;
94D2      9D 8C 1E       STA SPRITE_TYPE_$1E8C,X                    ;Set sprite type to Mommy, but
will render as skull
94D5      A9 02          LDA #$02                                   ;
94D7      1D 91 1F       ORA SPRITE_STATE_$1F91,X                   ;Set sprite state (add dying
flag bit)
94DA      9D 91 1F       STA SPRITE_STATE_$1F91,X                   ;
94DD      A9 08          LDA #$08                                   ;Set facing direction (8 is a
special flag meaning DYING)
```

```
94DF      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X                    ;
94E2      4C 3C 95       JMP FJUSTDIE_$953C                            ;

FHCN:
94E5      C8             INY                                           ;Next Hulk
94E6      C4 D8          CPY SPTR                                      ;Done all Hulks?
94E8      90 CE          BCC FHCL_$94B8                                ;Branch if not

; Post collision detection with Hulks, if we get here human is alive
*
******************************************
*                                        *
*         FMOV    -- MOVE FAMILY         *
*                                        *
******************************************
*
*         USE OF OBJECT DATA TABLE ENTRIES:
*
*         DXTBL   - DIRECTION OF MOTION (0-7) OR 8 IF SKULL OR POINTS
*         DYTBL   - NOT USED
*         DTTBL   - #FRAMES FOR DIRECTION CHANGE
*         SATBL   - ANIMATION (0-4) OR (0-5) FOR SKULL AND POINTS
*         MISCTBL - UNUSED
*         CRTBL   - MOMMY (2), DADDY (3), AND MIKEY (4)
*
**********
*
*         FAMILY IS ALIVE AND WELL, SO MAKE THEM MOVE

FOK:
94EA      DE 82 1C       DEC DTTBL,X                                   ;COUNT FOR DIRECTION CHANGE
94ED      10 12          BPL FMOV1                                     ;JUMP PAST IF GO STRAIGHT
94EF      20 A8 D3       JSR RANDOM                                    ;IF TIME TO TURN:  GET A #
94F2      29 1F          AND #MASK5
94F4      69 0A          ADC #10                                       ;MINIMUM FAMILY DIR TIMER
94F6      9D 82 1C       STA DTTBL,X                                   ;USE IT AS THE NEW TIMER FOR
DIRECTION
94F9      20 A8 D3       JSR RANDOM                                    ;PICK A NUMBER FOR DIRECTION
94FC      29 07          AND #$07                                      ;CHANGE IT INTO A NUMBER 0 TO
7
94FE      9D D4 1B       STA DXTBL,X                                   ;SAVE THE NEW DIRECTION

FMOV1:
9501      BD CF 1A       LDA XTBL,X                                    ;LOAD HUMAN POSITION
9504      BC D4 1B       LDY DXTBL,X                                   ;GET THE DIRECTION
9507      18             CLC
9508      79 1D EC       ADC XDIRTBL,Y                                 ;ADD X STEP FOR THE DIRECTION
950B      85 BE          STA XINTEND                                   ;STORE NEW POSITION
950D      BD E3 1E       LDA XEXTBL,X
9510      18             CLC
9511      79 1D EC       ADC XDIRTBL,Y                                 ;ADD X STEP FOR THE DIRECTION
9514      85 C0          STA XXINTEND                                  ;STORE NEW EXTENT
9516      BD 26 1B       LDA YTBL,X
9519      18             CLC
951A      79 25 EC       ADC YDIRTBL,Y                                 ;ADD Y STEP.  X REG WAS THE
SAME
951D      85 BF          STA YINTEND                                   ;STORE NEW Y POS
951F      BD 3A 1F       LDA YEXTBL,X
9522      18             CLC
9523      79 25 EC       ADC YDIRTBL,Y                                 ;ADD Y STEP.  X REG WAS THE
SAME
9526      85 C1          STA YXINTEND                                  ;STORE NEW Y EXTENT

*         CHANGE ANIMATION STEP

; OK, now $BE = adjusted X, $C0 = adjusted X extent, $BF = adjusted Y, $C1 = adjusted Y extent
9528      DE D9 1C       DEC SATBL,X                                   ;GET CURRENT STEP IN ANIMATION
952B      10 2D          BPL FMOV5_$955A                               ;OK IF NON-NEGATIVE
952D      A9 03          LDA #$03                                      ;HIGHEST FAMILY ANIMATION STEP
952F      9D D9 1C       STA SATBL,X                                   ; Reset to 3rd frame
9532      4C 5A 95       JMP FMOV5_$955A                               ;NOW TRY TO LOAD THIS FAMILY

; Comes here if the human is dying
```

```
;
FDYING:                 ;A FAMILY MEMBER IS DYING
9535     BD D9 1C       LDA SATBL,X
9538     29 80          AND #$80                                     ;If bit 7 is set the person has
just died
953A     F0 10          BEQ $954C                                    ;FAMILY HAS BEEN DYING - NOW
REMOVE IT

*          NOW RESET HIGH BIT OF SATBL (LEAVING CORRECT ANIMATION STEP),
*          AND SET MTTBL TO A LARGE VALUE, AND LOAD THIS OBJECT

FJUSTDIE:
953C     BD D9 1C       LDA SATBL,X                                  ;GET STEP WITH HI-BIT SET
953F     29 7F          AND #$7F                                     ;RESET HIGH BIT
9541     9D D9 1C       STA SATBL,X
9544     A9 3C          LDA #FDIEWAIT
9546     9D 7D 1B       STA MTTBL,X                                  // Must show skull for 60
frames?

*          NOW GO AHEAD AND DO A SPECIAL LOAD...

9549     4C 54 95       JMP FDYING2_$9554                            ;GO AHEAD AND LOAD USING TABLE
DATA

FDYING1:
954C     20 1E D7       JSR $D71E                                    ;Kill this sprite permanently
954F     C6 5B          DEC $5B                                      ;Decrement family on screen
counter
9551     4C FC 91       JMP OBJCONT_$91FC                            ;PROCESS NEXT OBJECT

FDYING2:
*          DO A SPECIAL LOAD HERE WHICH DOES NOT NEED THE INTEND
*                  VARIABLES TO BE SET... CORRECT DATA IS IN TABLES
*          FROM THIS ENTRY POINT THE UNLOADER SHOULD FETCH NEW STAMP
*     DATA USING GETSTAMP BUT SHOULD ASSUME THAT NO ZONE CHANGES WILL BE MADE
*          ON TO NEXT OBJECT
*                       ;JMP OBJCONT

9554     20 AF E1       JSR $E1AF
9557     4C FC 91       JMP OBJCONT_$91FC                            ;PROCESS NEXT OBJECT

; Set the human in a random direction for a random number of moves
FMOV5:
955A     20 AF E1       JSR $E1AF
955D     AD 3E 21       LDA $213E
9560     F0 18          BEQ FMOV6_$957A
9562     20 A8 D3       JSR RANDOM_$D3A8                             ;Get a random rumber in $C5 and
accumulator
9565     29 1F          AND #$1F
9567     69 0A          ADC #$0A
9569     9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X
956C     20 A8 D3       JSR RANDOM_$D3A8                             ;Get a random number in $C5 and
accumulator
956F     29 01          AND #$01
9571     18             CLC
9572     69 02          ADC #$02
9574     9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
9577     4C 01 95       JMP FMOV1_$9501

FMOV6:
957A     A5 BE          LDA XINTEND_BE
957C     9D CF 1A       STA SPRITE_X,X                               ;Write sprite X position
957F     A5 C0          LDA XXINTEND_C0
9581     9D E3 1E       STA SPRITE_X_EXTENT,X                        ;Write sprite Y position
9584     A5 BF          LDA YINTEND_BF
9586     9D 26 1B       STA SPRITE_Y,X
9589     A5 C1          LDA YYINTEND_C1
958B     9D 3A 1F       STA SPRITE_Y_EXTENT,X
958E     20 53 D1       JSR $D153
9591     A5 A4          LDA TEMP4
9593     F0 05          BEQ $959A
9595     A9 00          LDA #$00
9597     9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X
```

```
959A      A5 CD          LDA $CD
959C      9D 7D 1B       STA MTTBL,X
959F      AD 15 19       LDA $1915
95A2      F0 1F          BEQ $95C3
95A4      A4 DF          LDY $DF
95A6      B9 91 1F       LDA SPRITE_STATE_$1F91,Y
95A9      29 03          AND #$03
95AB      C9 01          CMP #$01
95AD      D0 07          BNE $95B6
95AF      20 5D DC       JSR $DC5D
95B2      A5 B7          LDA FRMCNT
95B4      D0 08          BNE $95BE
95B6      C8             INY
95B7      C4 D6          CPY $D6
95B9      90 EB          BCC $95A6
95BB      4C FC 91       JMP OBJCONT_$91FC                             ;PROCESS NEXT OBJECT
95BE      A9 00          LDA #$00
95C0      9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X

95C3      4C FC 91       JMP OBJCONT_$91FC                             ;PROCESS NEXT OBJECT
;
; HULK AI MANHANDLER
;
95C6      A5 E2          LDA $E2                                       ;Are we in game or attract
mode?
95C8      F0 03          BEQ $95CD                                     ;Game mode
95CA      4C FC 91       JMP OBJCONT_$91FC                             ;PROCESS NEXT OBJECT
95CD      AD 15 19       LDA $1915
95D0      F0 2E          BEQ $9600
95D2      A4 DF          LDY $DF                                       ;Get index of first electrode
95D4      B9 8C 1E       LDA SPRITE_TYPE_$1E8C,Y
95D7      29 1F          AND #$1F
95D9      C9 10          CMP #$10                                      ;An electrode? Hulks flatten
electrodes!!!!!!
95DB      D0 1E          BNE $95FB
95DD      B9 91 1F       LDA SPRITE_STATE_$1F91,Y                      ;Get the electrode's sprite
state
95E0      29 03          AND #$03
95E2      C9 01          CMP #$01                                      ;Is the electrode active?
95E4      D0 15          BNE $95FB                                     ;No, it must be in a dying state
95E6      20 5D DC       JSR $DC5D                                     ;Check if they overlap
95E9      A5 B7          LDA FRMCNT                                    ; $B7 = 1 if overlap
95EB      F0 0E          BEQ $95FB                                     ;No overlap
95ED      A5 10          LDA $10                                       ;???? pointless opcode, a is
overwritten
95EF      86 A1          STX TEMP1                                     ;Save Hulk's sprite index in A1
95F1      98             TYA                                           ;A = electrode index
95F2      AA             TAX                                           ;X = electrode index
95F3      20 0E D7       JSR $D70E
95F6      A6 A1          LDX TEMP1                                     ;Restore sprite index
95F8      4C 00 96       JMP $9600
95FB      C8             INY                                           ;Next electrode
95FC      C4 D6          CPY $D6                                       ;Have we reached the end of the
                                                                       ; electrodes? $D6= index of
first human
95FE      90 D4          BCC $95D4

*
******************************************
*                                        *
*         HMOV    -- MOVE HULKS          *
*                                        *
******************************************
*
*         USE OF OBJECT DATA TABLE ENTRIES:
*
*         DXTBL   - DIRECTION (0-3)
*         DYTBL   - UNUSED
*         DTTBL   - # OF FRAMES TO DIRECTION CHANGE
*         SATBL   - ANIMATION (0-3)
*         MISCTBL - DIRECTION TO JUMP AFTER BEING SHOT
*         CRTBL   - 5
*
```

```
**********************************************************************
*
*         THIS IS THE ROUTINE FOR A DUMB HULK THAT WALKS RANDOMLY
*         IT CAN LATER BE MODIFIED SO THAT THEY ARE OCCASIONALLY SMART
*         RIGHT NOW IT ABOUT THE SAME AS THE FAMILY ROUTINE
*
*         MAKE HULK JUMP IN 4-BIT DIRECTION OF MISCTBL
*
HMOV:
9600      BD 30 1D       LDA MISCTBL_$1D30,X                          ;LOAD DIRECTION TO JUMP
9603      F0 38          BEQ HMOV01_$963D                             ;IF DIR IS 0, MOVE NORMALLY
9605      A8             TAY
9606      B9 3D EC       LDA XDIRTBL4_$EC3D,Y                         ;Get the Hulk's x step
9609      0A             ASL A                                        ;Multiply it by 2
960A      18             CLC
960B      79 3D EC       ADC XDIRTBL4_$EC3D,Y                         ;Add it to itself to make it *
3
960E      85 B7          STA FRMCNT                                   ;Stuff result in a temporary
zero page variable
9610      18             CLC
9611      7D CF 1A       ADC SPRITE_X,X
9614      85 BE          STA XINTEND_BE
9616      BD E3 1E       LDA SPRITE_X_EXTENT,X
9619      18             CLC
961A      65 B7          ADC FRMCNT                                   ;Add on precalc'd x step * 3
961C      85 C0          STA XXINTEND_C0
961E      BD 26 1B       LDA SPRITE_Y,X
9621      18             CLC
9622      79 4D EC       ADC YDIRTBL4_$EC4D,Y                         ;Add YDIRTBL4
9625      85 BF          STA YINTEND_BF
9627      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
962A      18             CLC
962B      79 4D EC       ADC YDIRTBL4_$EC4D,Y                         ;Add YDIRTBL4
962E      85 C1          STA YYINTEND_C1
9630      A9 00          LDA #$00
9632      9D 30 1D       STA MISCTBL_$1D30,X
9635      A9 03          LDA #$03
9637      9D 7D 1B       STA MTTBL,X
963A      4C AE 96       JMP HMOV2_$96AE

HMOV01:
963D      DE 82 1C       DEC DTTBL,X                                  ;COUNT FOR DIRECTION CHANGE
9640      10 3A          BPL HMOV1_$967C                              ;JUMP PAST IF GO STRAIGHT
9642      20 A8 D3       JSR RANDOM_$D3A8                             ;IF TIME TO TURN:  GET A #
9645      29 0F          AND #MASK4
9647      69 00          ADC #$00                                     ;MINIMUM DIR TIMER INITIAL
VALUE
9649      9D 82 1C       STA DTTBL,X                                  ;USE IT AS THE NEW TIMER FOR
DIRECTION
964C      BD 2B 1C       LDA SPRITE_DELTA_Y_$1C2B,X
964F      30 0E          BMI $965F
9651      F0 0C          BEQ $965F
9653      A0 01          LDY $01
9655      84 B7          STY FRMCNT
9657      88             DEY                                          ;Reduce Y to 0.  This will
consider
                                                                      ; player's coords in the
following subcall
9658      20 6E BD       JSR PICK_DIRECTION_BD6E                      ;Get a direction
965B      C9 04          CMP #$04                                     ;Hulk can only go N, S, E, W,
which
                                                                      ; corresponds to 0-3
965D      90 05          BCC $9664
965F      20 A8 D3       JSR RANDOM_$D3A8                             ;PICK A NUMBER FOR DIRECTION
9662      29 03          AND #$03                                     ;CHANGE IT INTO A NUMBER 0 TO
3
9664      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X                   ;STORE NEW DIRECTION
9667      20 35 D1       JSR GETEXTEN_$D135                           ;GET THE NEW EXTENTS
966A      BD CF 1A       LDA XTBL,X                                   ;GET X POS
966D      18             CLC
966E      65 AB          ADC TEMP11                                   ;ADD X SIZE
9670      9D E3 1E       STA XEXTBL,X
9673      BD 26 1B       LDA YTBL,X                                   ;GET Y POS
```

```
9676      18              CLC
9677      65 AC           ADC TEMP12                                  ;ADD Y SIZE
9679      9D 3A 1F        STA YEXTBL,X                                ;RENEW EXTENT

HMOV1:
967C      BC D4 1B        LDY XTBL,X                                  ;LOAD HULK POSITION
967F      B9 1D EC        LDA DXTBL,Y                                 ;GET THE DIRECTION
9682      0A              ASL A
9683      18              CLC
9684      79 1D EC        ADC XDIRTBL,Y                               ;ADD X STEP FOR THE DIRECTION
9687      85 B7           STA FRMCNT
9689      18              CLC
968A      7D CF 1A        ADC XDIRTBL,X                               ;HULK STEPS
968D      85 BE           STA XINTEND                                 ;STORE NEW POSITION
968F      BD E3 1E        LDA XEXTBL,X                                ;GET X EXTENT
9692      18              CLC
9693      65 B7           ADC XDIRTBL,Y                               ;STEP
9695      85 C0           STA XXINTEND_C0
9697      BD 26 1B        LDA YTBL,X
969A      18              CLC
969B      79 25 EC        ADC YDIRTBL,Y                               ;ADD Y STEP.  X REG WAS THE
SAME
969E      85 BF           STA YINTEND                                 ;STORE NEW Y POS
96A0      BD 3A 1F        LDA YEXTBL,X                                ;GET Y EXTENT
96A3      18              CLC
96A4      79 25 EC        ADC YDIRTBL,Y                               ;MOVE THE OTHER EDGE
96A7      85 C1           STA YXINTEND
96A9      A5 CE           LDA $CE                                     ;Read number of frames to wait
(this is
96AB      9D 7D 1B        STA MTTBL,X                                 ; set by skill level)

*          CHANGE ANIMATION STEP

HMOV2:
96AE      DE D9 1C        DEC SATBL,X                                 ;DECREMENT THE ANIMATION
96B1      10 05           BPL HMOV2_$96B8                             ;OK IF NON-NEGATIVE
96B3      A9 03           LDA #$03                                    ;HIGHEST HULK ANIMATION STEP
96B5      9D D9 1C        STA SATBL,X                                 ;NEW ANIMATION STEP

HMOV2
96B8      20 B4 D1        JSR CHKINTBD_$D1B4                          ;Ensure that sprite is in the
playfield
                                                                      ;$BE = X of object $BF = Y of
object $C0 = X
                                                                      ;Extent $C1 = Y extent
96BB      A5 A4           LDA TEMP4
96BD      F0 05           BEQ $96C4
96BF      A9 00           LDA #$00
96C1      9D 82 1C        STA MOVES_B4_DIR_CHANGE_$1C82,X
96C4      20 AF E1        JSR $E1AF
96C7      AD 3E 21        LDA $213E
96CA      F0 15           BEQ $96E1
96CC      20 A8 D3        JSR RANDOM_$D3A8
96CF      29 1F           AND #$1F
96D1      69 00           ADC #$00
96D3      9D 82 1C        STA MOVES_B4_DIR_CHANGE_$1C82,X
96D6      20 A8 D3        JSR RANDOM_$D3A8
96D9      29 01           AND #$01
96DB      18              CLC
96DC      69 02           ADC #$02
96DE      4C 64 96        JMP $9664
96E1      A5 BE           LDA XINTEND_BE
96E3      9D CF 1A        STA SPRITE_X,X
96E6      A5 C0           LDA XXINTEND_C0
96E8      9D E3 1E        STA SPRITE_X_EXTENT,X
96EB      A5 BF           LDA YINTEND_BF
96ED      9D 26 1B        STA SPRITE_Y,X
96F0      A5 C1           LDA YYINTEND_C1
96F2      9D 3A 1F        STA SPRITE_Y_EXTENT,X
96F5      4C FC 91        JMP OBJCONT_$91FC                           ;PROCESS NEXT OBJECT

*
******************************************
```

```
*                                         *
*          SMOV   -- MOVE SPHEROIDS       *
*                                         *
*******************************************
*
*          USE OF OBJECT DATA TABLE ENTRIES:
*
*          DXTBL   - DELTA X
*          DYTBL   - DELTA Y
*          DTTBL   - # OF BIRTHS REMAINING
*          SATBL   - ANIMATION STEP (0-5)
*          MISCTBL - # OF MOVES UNTIL BIRTH
*          CRTBL   - 6
*
***********
*
SMOV:
96F8      BD 91 1F       LDA SPRITE_STATE_$1F91,X                      ;Get sprite state
96FB      D0 03          BNE $9700                                     ; if active (nonzero)
96FD      4C FC 91       JMP OBJCONT_$91FC                             ;PROCESS NEXT OBJECT
9700      C9 01          CMP #$01                                      ;Is this Spheroid active or
dying
                                                                       ; (1 = active)

9702      F0 24          BEQ $9728                                     ; if active
9704      BD D9 1C       LDA SATBL,X                                   ;OK, this Spheroid's dying
here
9707      C9 06          CMP #$06
9709      D0 06          BNE $9711
970B      20 1E D7       JSR $D71E                                     ;Kill this Spheroid
permanently
970E      4C FC 91       JMP OBJCONT_$91FC                             ;PROCESS NEXT OBJECT
9711      A9 06          LDA #$06
9713      9D D9 1C       STA SATBL,X
9716      A9 60          LDA #$60
9718      9D 7D 1B       STA MTTBL,X
971B      A9 0B          LDA #$0B                                      ;Play Death Noise
971D      20 95 E3       JSR DOTUNE_$E395
9720      C6 C9          DEC CRELEFT                                   ;Reduce enemies on screen
counter
9722      20 AF E1       JSR $E1AF
9725      4C FC 91       JMP OBJCONT_$91FC                             ;PROCESS NEXT OBJECT

; if we get here the Spheroid is alive
9728      20 28 BA       JSR ALTER_DELTAS_BA28                         ;Alter the Spheroid's deltas,
if required,
                                                                       ; to change its angle of

movement
972B      18             CLC
972C      BD CF 1A       LDA SPRITE_X,X                                ;Do the usual "is in bounds of
screen
                                                                       ; stuff" documented elsewhere

in this code
972F      65 B1          ADC TEMP17
9731      85 BE          STA XINTEND_BE
9733      18             CLC
9734      BD E3 1E       LDA SPRITE_X_EXTENT,X
9737      65 B1          ADC TEMP17
9739      85 C0          STA XXINTEND_C0
973B      C9 9C          CMP #$9C
973D      B0 09          BCS $9748
973F      A5 BE          LDA XINTEND_BE
9741      18             CLC
9742      69 10          ADC #$10
9744      C9 12          CMP #$12
9746      B0 0F          BCS $9757
9748      BD CF 1A       LDA SPRITE_X,X
974B      85 BE          STA XINTEND_BE
974D      BD E3 1E       LDA SPRITE_X_EXTENT,X
9750      85 C0          STA XXINTEND_C0
9752      BD 30 1D       LDA MISCTBL_$1D30,X
9755      F0 29          BEQ $9780
9757      18             CLC
```

```
9758      BD 26 1B       LDA SPRITE_Y,X
975B      65 B2          ADC TEMP18
975D      85 BF          STA YINTEND_BF
975F      18             CLC
9760      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
9763      65 B2          ADC TEMP18
9765      85 C1          STA YYINTEND_C1
9767      C9 BC          CMP #$BC
9769      B0 06          BCS $9771
976B      A5 BF          LDA YINTEND_BF
976D      C9 12          CMP #$12
976F      B0 15          BCS $9786
9771      BD 26 1B       LDA SPRITE_Y,X
9774      85 BF          STA YINTEND_BF
9776      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
9779      85 C1          STA YYINTEND_C1
977B      BD 30 1D       LDA MISCTBL_$1D30,X
977E      D0 06          BNE $9786
9780      20 1C D7       JSR $D71C                                  ;Mark this as dead
9783      4C FC 91       JMP OBJCONT_$91FC                          ;PROCESS NEXT OBJECT

9786      A5 E2          LDA $E2                                    ;Check play state
9788      D0 5A          BNE $97E4                                  ;Screen being drawn, don't
bother
978A      BD 82 1C       LDA MOVES_B4_DIR_CHANGE_$1C82,X
978D      38             SEC
978E      E9 01          SBC #$01
9790      9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X
9793      10 1A          BPL $97AF                                  ;Non zero, so keep the Spheroid
moving
                                                                    ; in its current direction

; if we get here, the Spheroid needs to change direction
9795      A9 01          LDA #$01                                   ;We want a number in range of
-1 to 1
9797      85 A4          STA TEMP4
9799      20 05 D4       JSR RANDPM_$D405                           ;Get number
979C      85 B3          STA TEMP19                                 ;Save in $B3
979E      20 05 D4       JSR RANDPM_$D405                           ;We want another number in
range of -1 to 1
97A1      85 B4          STA TEMP20
97A3      20 A8 D3       JSR RANDOM_$D3A8                           ;Get number
97A6      29 0F          AND #$0F
97A8      C9 08          CMP #$08
97AA      10 F7          BPL $97A3                                  ;Get another number, this one
isn't good enough
97AC      9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X
97AF      A5 B1          LDA TEMP17                                 ;$B1 and $B2 were set by the
call ALTER_DELTAS_BA28
97B1      18             CLC                                        ; and $B3 and $B4 were set above
97B2      65 B3          ADC TEMP19
97B4      85 B1          STA TEMP17
97B6      30 08          BMI $97C0
97B8      A9 05          LDA #$05
97BA      C5 B1          CMP TEMP17
97BC      90 08          BCC $97C6
97BE      B0 08          BCS $97C8
97C0      A9 FA          LDA #$FA
97C2      C5 B1          CMP TEMP17
97C4      90 02          BCC $97C8
97C6      85 B1          STA TEMP17
97C8      A5 B2          LDA TEMP18
97CA      18             CLC
97CB      65 B4          ADC TEMP20
97CD      85 B2          STA TEMP18
97CF      30 08          BMI $97D9
97D1      A9 05          LDA #$05
97D3      C5 B2          CMP TEMP18
97D5      90 08          BCC $97DF
97D7      B0 08          BCS $97E1
97D9      A9 FA          LDA #$FA
97DB      C5 B2          CMP TEMP18
97DD      90 02          BCC $97E1
```

```
97DF      85 B2          STA TEMP18
97E1      20 03 BA       JSR SET_OBJECT_DELTAXY_$BA03                 ;Set new deltas for object
97E4      A5 CF          LDA $CF                                      ;Get Spheroid speed
97E6      9D 7D 1B       STA MTTBL,X                                  ;Save in frames before move
variable
97E9      FE D9 1C       INC SATBL,X
97EC      BD D9 1C       LDA SATBL,X
97EF      C5 EA          CMP $EA
97F1      90 05          BCC $97F8
97F3      A9 00          LDA #$00
97F5      9D D9 1C       STA SATBL,X
;
; Seems to be "draw object and set X,Y and extents" function
;
97F8      20 AF E1       JSR $E1AF
97FB      AD 3E 21       LDA $213E                                    ;Success?
97FE      F0 0D          BEQ $980D                                    ;Yes
9800      BD 26 1B       LDA SPRITE_Y,X
9803      85 BF          STA YINTEND_BF
9805      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
9808      85 C1          STA YYINTEND_C1
980A      4C F8 97       JMP $97F8
980D      A5 BE          LDA XINTEND_BE
980F      9D CF 1A       STA SPRITE_X,X
9812      A5 C0          LDA XXINTEND_C0
9814      9D E3 1E       STA SPRITE_X_EXTENT,X
9817      A5 BF          LDA YINTEND_BF
9819      9D 26 1B       STA SPRITE_Y,X
981C      A5 C1          LDA YYINTEND_C1
981E      9D 3A 1F       STA SPRITE_Y_EXTENT,X
9821      4C FC 91       JMP OBJCONT_$91FC                            ;PROCESS NEXT OBJECT

******************************************
*
******************************************
*                                        *
*         QMOV   -- MOVE QUARKS          *
*                                        *
******************************************
*
*         USE OF OBJECT DATA TABLE ENTRIES:
*
*         DXTBL   - DIRECTION  ( 4 - 7 )
*         DYTBL   - DIRECTION CHANGE TIMER
*         DTTBL   - # OF BIRTHS REMAINING
*         SATBL   - ANIMATION STEP (0-5)
*         MISCTBL - # OF MOVES UNTIL BIRTH
*         CRTBL   - 7
*
**********
*
9824      A5 E2          LDA $E2                                      ;Can we do anything?
9826      F0 03          BEQ $982B
9828      4C 0E 99       JMP $990E
982B      BD 91 1F       LDA SPRITE_STATE_$1F91,X                     ;Read sprite state
982E      D0 03          BNE $9833                                    ;If non zero, our Quark might
be alive
9830      4C FC 91       JMP OBJCONT_$91FC                            ;PROCESS NEXT OBJECT
9833      C9 01          CMP #$01                                     ;Alive?
9835      F0 1F          BEQ $9856                                    ;Yes

; if we get here the Quark is dying
9837      BD D9 1C       LDA SATBL,X
983A      C9 0C          CMP #$0C                                     ;Are we at the last Quark death
anim frame?
983C      D0 06          BNE $9844                                    ;No
983E      20 1E D7       JSR $D71E                                    ;Yes, kill this Quark off
9841      4C FC 91       JMP OBJCONT_$91FC                            ;PROCESS NEXT OBJECT
9844      A9 0C          LDA #$0C
9846      9D D9 1C       STA SATBL,X
9849      A9 60          LDA #$60
984B      9D 7D 1B       STA MTTBL,X
984E      C6 C9          DEC CRELEFT                                  ;Decrease enemy count
```

```
9850      20 AF E1       JSR $E1AF
9853      4C FC 91       JMP OBJCONT_$91FC                            ;PROCESS NEXT OBJECT

; if we come here, we have a live Quark!
; Quarks give birth to tanks.
;
9856      A5 EB          LDA $EB
9858      D0 76          BNE $98D0
985A      BD 82 1C       LDA MOVES_B4_DIR_CHANGE_$1C82,X              ;If 0, then Quark can change
direction
985D      F0 71          BEQ $98D0                                    ;Yes, its 0
985F      BD 26 1B       LDA SPRITE_Y,X
9862      C9 AD          CMP TEMP13
9864      B0 6A          BCS $98D0

; do we want to give birth to a Tank?
9866      20 A8 D3       JSR RANDOM_$D3A8                             ;Get a random number
9869      29 3F          AND #$3F                                     ;Mask off
986B      C5 EC          CMP $EC                                      ;Is the result > our threshold
for
                                                                      ; creating a tank?
986D      B0 61          BCS $98D0                                    ;Yes, so don't create a tank

; give birth to a tank (maybe) - depends if there's a slot free for the tank or not
986F      86 A2          STX TEMP2                                    ;Save current object index in
temp var
9871      20 34 92       JSR GET_MISSILE_SLOT_$9234                   ;Get a free slot for our "born"
tank to use
9874      A6 A2          LDX TEMP2                                    ;Restore
9876      A8             TAY                                          ;A = slot index from function,
move to Y
                                                                      ; for indexing
9877      30 57          BMI $98D0                                    ;We don't have any free slots

; if we get here, we're going to create a tank
9879      A9 01          LDA #$01                                     ;Make active
987B      99 91 1F       STA SPRITE_STATE_$1F91,Y
987E      BD CF 1A       LDA SPRITE_X,X                               ;Tank X coord = Quark X coord
9881      99 CF 1A       STA SPRITE_X,Y
9884      18             CLC
9885      69 09          ADC #$09

9887      99 E3 1E       STA SPRITE_X_EXTENT,Y                        ;Do the usual extent stuff
988A      BD 26 1B       LDA SPRITE_Y,X
988D      99 26 1B       STA SPRITE_Y,Y                               ;Tank Y coord = Quark Y coord
9890      18             CLC
9891      69 0F          ADC #$0F
9893      99 3A 1F       STA SPRITE_Y_EXTENT,Y
9896      A9 09          LDA #$09                                     ;And lo, a Tank was born into
this world
9898      99 8C 1E       STA SPRITE_TYPE_$1E8C,Y
989B      84 A4          STY TEMP4                                    ;Save Y and X into temp
variables
989D      86 A5          STX TEMP5
989F      98             TYA
98A0      AA             TAX
98A1      20 36 E1       JSR $E136                                    ;Call magic function which I
have no idea
                                                                      ; does, except set $213E to 0
on success
98A4      A4 A4          LDY TEMP4
98A6      A6 A5          LDX TEMP5                                    ;Restore x and y from temp
variables
98A8      AD 3E 21       LDA $213E                                    ;Check success flag
98AB      D0 20          BNE $98CD                                    ;Failed to create this tank

; if we get here, the tank is almost ready to be added to the playfield
98AD      A9 00          LDA #$00

98AF      99 D9 1C       STA SATBL,X
98B2      99 D4 1B       STA SPRITE_DELTA_X_$1BD4,Y
98B5      99 2B 1C       STA SPRITE_DELTA_Y_$1C2B,Y
98B8      A9 03          LDA #$03
```

```
98BA      99 7D 1B       STA MTTBL,Y
98BD      EE 0E 19       INC $190E                                  ;OK, we've spawned a tank
98C0      E6 C9          INC CRELEFT                                ;Increment enemy count
98C2      A9 11          LDA #$11                                   ;Play Tank Birth Sound
98C4      20 95 E3       JSR DOTUNE_$E395
98C7      DE 82 1C       DEC MOVES_B4_DIR_CHANGE_$1C82,X
98CA      4C D0 98       JMP $98D0

; tank couldn't be created
98CD      20 FB BA       JSR RECORD_OPEN_SLOT_$BAFB                 ;Record this open slot

; but keep the Quark moving anyway
QMOV:
98D0      18             CLC
98D1      BC D4 1B       LDY SPRITE_DELTA_X_$1BD4,X                 ;Get Quark direction
98D4      BD CF 1A       LDA SPRITE_X,X
98D7      79 2D EC       ADC QUARKXDIRTBL_$EC2D,Y                   ;MOVE ACCORDING TO THE
DIRECTION
98DA      85 BE          STA XINTEND_BE
98DC      BD E3 1E       LDA SPRITE_X_EXTENT,X
98DF      18             CLC
98E0      79 2D EC       ADC QUARKXDIRTBL_$EC2D,Y
98E3      85 C0          STA XXINTEND_C0
98E5      BD 26 1B       LDA SPRITE_Y,X
98E8      18             CLC
98E9      79 35 EC       ADC QUARKYDIRTBL_$EC35,Y
98EC      85 BF          STA YINTEND_BF
98EE      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
98F1      18             CLC
98F2      79 35 EC       ADC QUARKYDIRTBL_$EC35,Y
98F5      85 C1          STA YYINTEND_C1
98F7      20 B4 D1       JSR CHKINTBD_$D1B4                         ;Ensure that sprite is in the
playfield.
98FA      A5 A4          LDA TEMP4
98FC      F0 10          BEQ $990E
98FE      BD 82 1C       LDA MOVES_B4_DIR_CHANGE_$1C82,X
9901      D0 06          BNE $9909
9903      20 1C D7       JSR $D71C                                  ;Kill this Quark completely
9906      4C FC 91       JMP OBJCONT_$91FC                          ;PROCESS NEXT OBJECT
9909      A9 00          LDA #$00
990B      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
990E      A9 03          LDA #$03
9910      9D 7D 1B       STA MTTBL,X
9913      FE D9 1C       INC SATBL,X
9916      BD D9 1C       LDA SATBL,X
9919      C5 60          CMP $60
991B      90 05          BCC $9922
991D      A9 00          LDA #$00
991F      9D D9 1C       STA SATBL,X
9922      A5 E2          LDA $E2
9924      D0 1A          BNE QMOV6_$9940
9926      DE 2B 1C       DEC SPRITE_DELTA_Y_$1C2B,X
9929      10 12          BPL QMOV5_$993D
992B      20 A8 D3       JSR RANDOM_$D3A8
992E      29 07          AND #$07
9930      09 04          ORA #$04
9932      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
9935      20 A8 D3       JSR RANDOM_$D3A8
9938      29 3F          AND #$3F
993A      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X

QMOV5:
993D      4C F8 97       JMP $97F8

QMOV6:
9940      BD CF 1A       LDA SPRITE_X,X
9943      85 BE          STA XINTEND_BE
9945      BD E3 1E       LDA SPRITE_X_EXTENT,X
9948      85 C0          STA XXINTEND_C0
994A      BD 26 1B       LDA SPRITE_Y,X
994D      85 BF          STA YINTEND_BF
994F      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
9952      85 C1          STA YYINTEND_C1
```

```
9954      4C F8 97       JMP $97F8
;
; ENFORCER AI HANDLER
;
9957      BD 91 1F       LDA SPRITE_STATE_$1F91,X
995A      D0 03          BNE $995F                                  ;Non-zero, he's alive!
995C      4C FC 91       JMP OBJCONT_$91FC                          ;It's zero, this guy is dead,
process
                                                                    ; next object

; OK, Enforcer's alive, what do we do now
995F      BD 30 1D       LDA MISCTBL_$1D30,X                        ;
9962      9D 7D 1B       STA MTTBL,X
9965      BD D9 1C       LDA SATBL,X                                ;Get sprite animation frame
9968      C9 04          CMP #$04                                   ;We on 4th frame (as in, its
alive)?
996A      F0 27          BEQ $9993                                  ;Yes, Enforcer is alive
996C      90 05          BCC $9973                                  ;Less than 4, the Enforcer is
dying

; if we get here, the Enforcer is dying
996E      A9 01          LDA #$01                                   ;OK, wait 1 frame before doing
anything, OK?
9970      9D 7D 1B       STA MTTBL,X
9973      FE D9 1C       INC SATBL,X                                ;Bump to next animation frame
9976      BD D9 1C       LDA SATBL,X                                ;Get animation frame
9979      C9 08          CMP #$08                                   ;8th frame?
997B      90 2C          BCC $99A9
997D      C6 ED          DEC $ED                                    ;
997F      BD 8C 1E       LDA SPRITE_TYPE_$1E8C,X                    ;Get sprite type and divide by
32
9982      4A             LSR A
9983      4A             LSR A
9984      4A             LSR A
9985      4A             LSR A
9986      4A             LSR A
9987      A8             TAY                                        ;Convert the result to an index
(no idea why)
9988      A9 FF          LDA #$FF
998A      99 26 19       STA $1926,Y
998D      20 1C D7       JSR $D71C                                  ;Kill this Enforcer
9990      4C FC 91       JMP OBJCONT_$91FC                          ;PROCESS NEXT OBJECT

; if we get here the Enforcer is *possibly* alive, let's check its state
9993      BD 91 1F       LDA SPRITE_STATE_$1F91,X                   ;
9996      C9 03          CMP #$03                                   ;Dying?
9998      D0 29          BNE $99C3                                  ;Nope, still alive
999A      A9 05          LDA #$05
999C      9D D9 1C       STA SATBL,X
999F      A9 01          LDA #$01
99A1      9D 7D 1B       STA MTTBL,X
99A4      A9 02          LDA #$02                                   ;Play Generic Explosion Sound
99A6      20 95 E3       JSR DOTUNE_$E395
99A9      BD 26 1B       LDA SPRITE_Y,X
99AC      85 BF          STA YINTEND_BF
99AE      BD CF 1A       LDA SPRITE_X,X
99B1      85 BE          STA XINTEND_BE
99B3      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
99B6      85 C1          STA YYINTEND_C1
99B8      BD E3 1E       LDA SPRITE_X_EXTENT,X
99BB      85 C0          STA XXINTEND_C0
99BD      20 AF E1       JSR $E1AF                                  ;Mark Enforcer as truly dead
99C0      4C FC 91       JMP OBJCONT_$91FC                          ;PROCESS NEXT OBJECT

; If we get here, the Enforcer is alive
99C3      DE 82 1C       DEC MOVES_B4_DIR_CHANGE_$1C82,X            ;Is it time for the Enforcer to
change
                                                                          ;  direction yet?
99C6      D0 56          BNE $9A1E                                  ;No
99C8      20 A8 D3       JSR RANDOM_$D3A8                           ;Time to change direction. Get
a random #
99CB      29 1F          AND #$1F                                   ;From 0..31
99CD      4A             LSR A                                      ;Divide by 2
```

```
99CE      D0 1E         BNE $99EE                                  ;If it's not 0, then
99D0      A9 07         LDA #$07                                   ; specify number range from -7
to +7
                                                                   ; (it's a mask)

99D2      85 A4         STA TEMP4
99D4      20 05 D4      JSR RANDPM_$D405                           ;Call function
99D7      9D D4 1B      STA SPRITE_DELTA_X_$1BD4,X                 ;Set sprite delta X (X
increment) to result
99DA      20 05 D4      JSR RANDPM_$D405                           ;And call function again
99DD      9D 2B 1C      STA SPRITE_DELTA_Y_$1C2B,X                 ;Set sprite delta Y
99E0      20 A8 D3      JSR RANDOM_$D3A8                           ;Get a random number

99E3      F0 FB         BEQ $99E0                                  ;If zero, get another random
number!!!
99E5      29 0F         AND #$0F                                   ;Mask off lower 4 bits
99E7      69 04         ADC #$04                                   ;Add 4
99E9      9D 82 1C      STA MOVES_B4_DIR_CHANGE_$1C82,X            ;And that's how many moves
before we
                                                                         ; change direction!!
99EC      D0 30         BNE $9A1E                                  ;A cheap JMP equivalent
99EE      A9 01         LDA #$01
99F0      85 B7         STA FRMCNT
99F2      AD CF 1A      LDA SPRITE_X

99F5      85 B8         STA TEMPX
99F7      AD 26 1B      LDA SPRITE_Y
99FA      85 B9         STA TEMPY
99FC      20 DC BC      JSR COMPUTE_DELTAS_$BCDC                   ;Call function that computes
differences
                                                                   ; between x coords of objects
(in this case,
                                                                   ; Enforcer and player)
99FF      A5 B1         LDA TEMP17
9A01      9D D4 1B      STA SPRITE_DELTA_X_$1BD4,X                 ;Save X delta
9A04      A5 B2         LDA TEMP18
9A06      9D 2B 1C      STA SPRITE_DELTA_Y_$1C2B,X                 ;Save Y delta
9A09      A5 AA         LDA TEMP10
9A0B      9D 30 1D      STA MISCTBL_$1D30,X
9A0E      20 A8 D3      JSR RANDOM_$D3A8                           ;Get a random number
9A11      C9 18         CMP #$18

9A13      B0 F9         BCS $9A0E                                  ;If 24 or more, get another one
9A15      C9 00         CMP #$00
9A17      D0 02         BNE $9A1B                                  ;If not 0
9A19      A9 50         LDA #$50
9A1B      9D 82 1C      STA MOVES_B4_DIR_CHANGE_$1C82,X            ;Save number to moves before
direction
                                                                         ; change
9A1E      BC D4 1B      LDY SPRITE_DELTA_X_$1BD4,X                 ;Get X delta into Y register
9A21      98            TYA
9A22      18            CLC
9A23      7D CF 1A      ADC SPRITE_X,X                             ;Add to current X coordinate
9A26      85 BE         STA XINTEND_BE                             ;Store in intended X temp
variable
9A28      98            TYA
9A29      18            CLC
9A2A      7D E3 1E      ADC SPRITE_X_EXTENT,X                      ;Add to current X extent coord
9A2D      85 C0         STA XXINTEND_C0
9A2F      BC 2B 1C      LDY SPRITE_DELTA_Y_$1C2B,X                 ;Get Y delta into Y register
9A32      98            TYA
9A33      18            CLC
9A34      7D 26 1B      ADC SPRITE_Y,X
9A37      85 BF         STA YINTEND_BF
9A39      98            TYA
9A3A      18            CLC
9A3B      7D 3A 1F      ADC SPRITE_Y_EXTENT,X
9A3E      85 C1         STA YYINTEND_C1
9A40      20 B4 D1      JSR CHKINTBD_$D1B4                         ;Ensure that sprite is in the
playfield
9A43      20 AF E1      JSR $E1AF                                  ;Draw the Enforcer
9A46      AD 3E 21      LDA $213E                                  ;Could it be drawn?
9A49      F0 0D         BEQ $9A58                                  ;Yes
```

```
; if we got here, then we've had an issue with the y-coordinate when drawing our sprite
; so re-draw using old Y coord
9A4B      BD 26 1B       LDA SPRITE_Y,X

9A4E      85 BF          STA YINTEND_BF
9A50      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
9A53      85 C1          STA YYINTEND_C1
9A55      4C 43 9A       JMP $9A43

; when we get here, the new X, Y coordinates and extents of the Enforcer are in temp variables
; so save them back to the Enforcer
9A58      A5 BF          LDA YINTEND_BF
9A5A      9D 26 1B       STA SPRITE_Y,X
9A5D      A5 C1          LDA YYINTEND_C1
9A5F      9D 3A 1F       STA SPRITE_Y_EXTENT,X
9A62      A5 BE          LDA XINTEND_BE
9A64      9D CF 1A       STA SPRITE_X,x
9A67      A5 C0          LDA XXINTEND_C0
9A69      9D E3 1E       STA SPRITE_X_EXTENT,X
9A6C      BD 30 1D       LDA MISCTBL_$1D30,X
9A6F      9D 7D 1B       STA MTTBL,X
9A72      86 B7          STX FRMCNT                                  ;Save current object index in
$b7
9A74      BD 8C 1E       lda SPRITE_TYPE_$1E8C,X
9A77      4A             LSR A
9A78      4A             LSR A
9A79      4A             LSR A
9A7A      4A             LSR A
9A7B      4A             LSR A
9A7C      AA             TAX
9A7D      DE 26 19       DEC $1926,X
9A80      10 13          BPL $9A95
9A82      A5 EE          LDA $EE
9A84      C9 10          CMP #$10
9A86      B0 0D          BCS $9A95
9A88      A5 71          LDA $71
9A8A      9D 26 19       STA $1926,X
9A8D      A6 B7          LDX FRMCNT                                  ;Restore current object index
from $b7
9A8F      20 BE B5       JSR $B5BE

9A92      4C FC 91       JMP OBJCONT_$91FC                           ;PROCESS NEXT OBJECT
9A95      A6 B7          LDX FRMCNT
9A97      4C FC 91       JMP OBJCONT_$91FC                           ;PROCESS NEXT OBJECT
;
; BRAIN AI HANDLER
;
9A9A      A5 E2          LDA $E2                                     ;Read play state
9A9C      D0 0C          BNE $9AAA
9A9E      BD 91 1F       LDA SPRITE_STATE_$1F91,X
9AA1      29 03          AND #$03
9AA3      D0 08          BNE $9AAD
9AA5      A9 00          LDA #$00
9AA7      9D 8C 1E       STA SPRITE_TYPE_$1E8C,X                     ;Save zero to this sprite type,
it's
                                                                     ; permadead
9AAA      4C FC 91       JMP OBJCONT_$91FC                           ;PROCESS NEXT OBJECT

9AAD      29 02          AND #$02                                    ;Dying?
9AAF      F0 03          BEQ $9AB4                                   ;No
9AB1      4C 13 9C       JMP $BRAINDEATH_$9C13                       ;OK, this Brain IS dying, so we
gotta do
                                                                     ; something.
; if we get here, the Brain is alive.
9AB4      DE 82 1C       DEC MOVES_B4_DIR_CHANGE_$1C82,X
9AB7      30 03          BMI $9ABC
9AB9      4C A5 9B       JMP $9BA5
9ABC      A9 03          LDA #$03
9ABE      9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X

9AC1      BC 2B 1C       LDY BRAIN_TARGET_INDEX_$1C2B,X              ;Do we have a target?
9AC4      F0 37          BEQ $9AFD                                   ;No
9AC6      B9 8C 1E       LDA SPRITE_TYPE_$1E8C,Y                     ;Read type of target
```

```
9AC9      29 1F          AND #$1F
9ACB      F0 0C          BEQ $9AD9
9ACD      C9 02          CMP #$02                                      ;Mommy?
9ACF      F0 2C          BEQ $9AFD
9AD1      C9 04          CMP #$04                                      ;Mikie?
9AD3      F0 28          BEQ $9AFD
9AD5      C9 03          CMP #$03                                      ;Daddy?
9AD7      F0 24          BEQ $9AFD
9AD9      A5 5B          LDA $5B                                       ;OK, if we don't have a family
target, we
                                                                       ; check number of family on
screen.
9ADB      D0 04          BNE $9AE1                                     ;We have some family, find a
family member
9ADD      A0 00          LDY $00                                       ;Set target index to player.
9ADF      F0 18          BEQ $9AF9
9AE1      C8             INY                                           ;Bump to next family member
9AE2      C4 D7          CPY $D7                                       ;Have we run out of family
members (ie our
                                                                       ; y index == (last family
member + 1) aka
                                                                       ; start of first Hulk on
screen).
9AE4      90 02          BCC $9AE8
9AE6      A4 D6          LDY $D6                                       ;OK, we'll have to start our
search for a
                                                                       ; valid family member from the
start then.
                                                                       ; Get index of first family
member.
9AE8      B9 8C 1E       LDA SPRITE_TYPE_$1E8C,Y                       ;What type of sprite are you,
sir/madam?
9AEB      29 1F          AND #$1F
9AED      C9 02          CMP #$02                                      ;Mommy?
9AEF      F0 08          BEQ $9AF9
9AF1      C9 04          CMP #$04                                      ;Mikey?
9AF3      F0 04          BEQ $9AF9
9AF5      C9 03          CMP #$03                                      ;Daddy?
9AF7      D0 E8          BNE $9AE1                                     ;Keep scanning til we find a
suitable
                                                                       ; family member.
9AF9      98             TYA
9AFA      9D 2B 1C       STA BRAIN_TARGET_INDEX_$1C2B,X                ;Set our new target
9AFD      A9 00          LDA #$00
9AFF      85 B7          STA FRMCNT                                    ;Find fastest path flag
9B01      20 6E BD       JSR PICK_DIRECTION_$BD6E
9B04      C9 0F          CMP #$0F                                      ;If direction is #$0F, means
"target reached"
9B06      F0 06          BEQ $9B0E
9B08      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X                    ;Set direction
9B0B      4C A5 9B       JMP $9BA5                                     ;Continue

; no direction change for the Brain required,
so let's see if we have any juicy humans to program!!!
9B0E      B9 8C 1E       LDA SPRITE_TYPE_$1E8C,Y                       ;Get target type
9B11      29 1F          AND #$1F                                      ;It's a valid sprite type, yes?
(as in,
                                                                       ; it's non-zero).
9B13      D0 03          BNE $9B18
9B15      4C A5 9B       JMP $9BA5                                     ;It's 0, so do nothing
9B18      B9 91 1F       LDA SPRITE_STATE_$1F91,Y                      ;How's our target's health?
9B1B      29 03          AND #$03
9B1D      C9 01          CMP #$01                                      ;Active?
9B1F      F0 03          BEQ $9B24                                     ;Yes
9B21      4C A5 9B       JMP $9BA5                                     ;Nope, the target's dead or
dying,
                                                                       ; so let's go a walk

; OK, where is our target in relation to our Brain. x = Brain index, y = target index
; The Brain will stand still looking at its target while programming takes place
9B24      BD CF 1A       LDA SPRITE_X,X
9B27      D9 CF 1A       CMP SPRITE_X,Y
9B2A      B0 04          BCS $9B30                                     ;Brain X > target X, so face
```

```
WEST.
9B2C      A9 02        LDA #$02                                    ;2 = East in DXTBL
9B2E      D0 02        BNE $9B32
9B30      A9 03        LDA #$03                                    ;3 = West in DXTBL
9B32      DD D4 1B     CMP SPRITE_DELTA_X_$1BD4,X                  ;We already going in this
direction?
9B35      F0 06        BEQ $9B3D                                   ;Yes, so no need to change, we
start programming
9B37      9D D4 1B     STA SPRITE_DELTA_X_$1BD4,X                  ;Otherwise, change direction
9B3A      4C 8B 9B     JMP $9B8B

; Begin programming that Prog!!!!!
9B3D      A9 0A        LDA #$0A
9B3F      9D D9 1C     STA SATBL,X
9B42      A9 7F        LDA #$7F                                    ;127 frames of programming the
Prog
9B44      9D 7D 1B     STA MTTBL,X
9B47      A9 0F        LDA #$0F                                    ;Start "Human being
programmed" sound -
                                                                   ; we're creating a Prog!!!!!!!
9B49      20 95 E3     JSR DOTUNE_$E395
9B4C      C6 5B        DEC $5B                                     ;Dec number of family on screen
counter
9B4E      A9 0B        LDA #$0B                                    ;Change type of sprite to
"Prog"
9B50      99 8C 1E     STA SPRITE_TYPE_$1E8C,Y
9B53      A9 04        LDA #$04
9B55      99 2B 1C     STA SPRITE_DELTA_Y_$1C2B,Y
9B58      99 D4 1B     STA SPRITE_DELTA_X_$1BD4,Y
9B5B      A9 00        LDA #$00
9B5D      99 D9 1C     STA SATBL,Y
9B60      A9 7F        LDA #$7F
9B62      99 7D 1B     STA MTTBL,Y                                 ;127 frames of jumping up and
down!!
9B65      A9 01        LDA #$01
9B67      99 82 1C     STA MOVES_B4_DIR_CHANGE_$1C82,Y
9B6A      8A           TXA
9B6B      99 30 1D     STA MISCTBL_$1D30,Y                         ;Save index of the programming
"Brain" in MISCTBL
9B6E      86 A0        STX TEMP0                                   ;Save X in temp variable
9B70      98           TYA
9B71      AA           TAX                                         ;X now is Prog index
9B72      BD CF 1A     LDA SPRITE_X,X
9B75      85 BE        STA XINTEND_BE
9B77      BD E3 1E     LDA SPRITE_X_EXTENT,X
9B7A      85 C0        STA XXINTEND_C0
9B7C      BD 26 1B     LDA SPRITE_Y,X
9B7F      85 BF        STA YINTEND_BF
9B81      BD 3A 1F     LDA SPRITE_Y_EXTENT,X
9B84      85 C1        STA YYINTEND_C1
9B86      20 AF E1     JSR $E1AF                                   ;Draw the Prog

9B89      A6 A0        LDX TEMP0                                   ;Restore X from temp variable.
X now is
                                                                   ;Brain index.
                                                                   ;And of course, draw the Brain.
9B8B      BD CF 1A     LDA SPRITE_X,X
9B8E      85 BE        STA XINTEND_BE
9B90      BD E3 1E     LDA SPRITE_X_EXTENT,X
9B93      85 C0        STA XXINTEND_C0
9B95      BD 26 1B     LDA SPRITE_Y,X
9B98      85 BF        STA YINTEND_BF
9B9A      BD 3A 1F     LDA SPRITE_Y_EXTENT,X
9B9D      85 C1        STA YYINTEND_C1
9B9F      20 AF E1     JSR $E1AF
9BA2      4C FC 91     JMP OBJCONT_$91FC                           ;PROCESS NEXT OBJECT

; if we get here, the Brain is alive, not programming, and would like to move and/or
;   fire a Cruise Missile

9BA5      A5 CB        LDA $CB                                     ;This is either 0 or 1, it seems
to flip
                                                                   ; between it.
```

```
9BA7      D0 13          BNE $9BBC
9BA9      DE 30 1D       DEC BRAIN_CRUISE_COUNTDOWN_$1D30,X           ;Count down before firing
Cruise
9BAC      D0 0E          BNE $9BBC                                    ;If not 0, don't fire Cruise
9BAE      20 A8 D3       JSR RANDOM_$D3A8                             ;OK, we're going to fire a
Cruise, but
                                                                      ; first reload countdown
9BB1      29 7F          AND #$7F                                     ;Get a number between 0 and
127, and add
                                                                      ; 3 to it.
9BB3      18             CLC
9BB4      69 03          ADC #$03
9BB6      9D 30 1D       STA BRAIN_CRUISE_COUNTDOWN_$1D30,X           ;Set count down
9BB9      20 81 B9       JSR $B981                                    ;Fire Cruise Missile

; Move Brain
9BBC      BC D4 1B       LDY SPRITE_DELTA_X_$1BD4,X                   ;Get direction (0-7) into y
9BBF      BD CF 1A       LDA SPRITE_X,X                               ;Get current sprite X
coordinate
9BC2      18             CLC
9BC3      79 1D EC       ADC XDIRTBL_$EC1D,Y                          ;Compute new sprite X
coordinate
9BC6      85 BE          STA XINTEND_BE                               ;And store it in intended X
coord
                                                                      ; (where sprite would *like* to
move to)
9BC8      BD E3 1E       LDA SPRITE_X_EXTENT,X
9BCB      18             CLC
9BCC      79 1D EC       ADC XDIRTBL_$EC1D,Y
9BCF      85 C0          STA XXINTEND_C0
9BD1      BD 26 1B       LDA SPRITE_Y,X
9BD4      18             CLC
9BD5      79 25 EC       ADC YDIRTBL_$EC25,Y                          ;Compute new sprite Y
coordinate
9BD8      85 BF          STA YINTEND_BF                               ; and store it in intended Y
coord
                                                                      ; (where sprite would *like* to
move to)
9BDA      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
9BDD      18             CLC
9BDE      79 25 EC       ADC YDIRTBL_$EC25,Y
9BE1      85 C1          STA YYINTEND_C1
9BE3      A5 D2          LDA $D2                                      ;Get Brain move delay
9BE5      9D 7D 1B       STA MTTBL,X
9BE8      BD D9 1C       LDA SATBL,X
9BEB      C9 0A          CMP #$0A
9BED      D0 04          BNE $9BF3
9BEF      A9 03          LDA #$03
9BF1      D0 07          BNE $9BFA
9BF3      DE D9 1C       DEC SATBL,X
9BF6      10 15          BPL $9C0D
9BF8      A9 03          LDA #$03
9BFA      9D D9 1C       STA SATBL,X
9BFD      20 35 D1       JSR $D135                                    ;Get width and height of
current frame
9C00      A5 BE          LDA XINTEND_BE
9C02      18             CLC
9C03      65 AB          ADC TEMP11                                   ;Add on computed width
9C05      85 C0          STA XXINTEND_C0
9C07      A5 BF          LDA YINTEND_BF
9C09      65 AC          ADC TEMP12                                   ;Add on computed height
9C0B      85 C1          STA YYINTEND_C1
9C0D      20 B4 D1       JSR CHKINTBD_$D1B4                           ;Ensure that sprite is in
playfield
9C10      4C F8 97       JMP $97F8                                    ;Draw the Brain and we're done
really


; The Brain's been killed.  So, the brain dies, and if it's Prog'ing a family member,
;   so does the Prog.

BRAINDEATH
9C13      BD D9 1C       LDA SATBL,X
```

```
9C16      C9 0A          CMP #$0A                                     ;Are we showing the
programming anim?
9C18      D0 34          BNE $9C4E                                    ;No

; We WERE programming but not now. How's our Prog getting on? If it's still alive, kill it
9C1A      BC 2B 1C       LDY BRAIN_TARGET_INDEX_$1C2B,X               ;Get index of family member
being Prog'd
9C1D      B9 91 1F       LDA SPRITE_STATE_$1F91,Y                     ;How's it doing?
9C20      F0 27          BEQ $9C49                                    ;Dead!!
9C22      C9 03          CMP #$03
9C24      F0 23          BEQ $9C49
9C26      A9 03          LDA #$03                                     ;Play Family Death Sound
9C28      20 95 E3       JSR DOTUNE_$E395                             ;
9C2B      A9 80          LDA #$80
9C2D      99 D9 1C       STA SATBL,Y
9C30      A9 02          LDA #$02                                     ;Mommy (also skull)
9C32      99 8C 1E       STA SPRITE_TYPE_$1E8C,Y
9C35      A9 02          LDA #$02                                     ;Dying flag
9C37      19 91 1F       ORA SPRITE_STATE_$1F91,Y
9C3A      99 91 1F       STA SPRITE_STATE_$1F91,Y                     ;Set dying flag
9C3D      A9 08          LDA #$08
9C3F      99 D4 1B       STA SPRITE_DELTA_X_$1BD4,Y
9C42      A9 01          LDA #$01
9C44      99 7D 1B       STA MTTBL,Y
9C47      E6 5B          INC $5B                                      ;Increment number of family on
screen
                                                                      ; *temporarily*
9C49      A9 00          LDA #$00
9C4B      9D D9 1C       STA SATBL,X
9C4E      B9 D9 1C       LDA SATBL,X
9C51      C9 04          CMP #$04
9C53      10 0D          BPL GDYING1_$9C62
9C55      A9 02          LDA #$02                                     ;Play Generic Explosion Sound
9C57      20 95 E3       JSR DOTUNE_$E395
9C5A      A9 03          LDA #$03                                     ;ONE BEFORE LOWEST DEATH
ANIMATION
9C5C      9D D9 1C       STA SATBL,X
9C5F      4C 66 9C       JMP GDYING2                                  ;CONTINUE: ADVANCE STEP AND
LOAD

GDYING1                  ;THIS GRUNT DIED AT LEAST A FRAME AGO
9C62      C9 09          CMP #$09                                     ;HIGHEST DEATH ANIMATION
9C64      F0 0B          BEQ GDIE_$9C71                               ;REALLY KILL THIS GRUNT

GDYING2:
9C66      FE D9 1C       INC SATBL,X                                  ;ADVANCE DEATH ANIMATION
9C69      A9 01          LDA #$01                                     ;GSPEED FOR A DYING GRUNT
9C6B      9D 7D 1B       STA MTTBL,X
9C6E      4C 77 9C       JMP GMOVD0_$9C77                             ;CONTINUE: TRY TO LOAD THIS
GRUNT

GDIE:
9C71      20 1C D7       JSR $D71C                                    ;MARK THIS BRAIN AS TRULY DEAD
9C74      4C FC 91       JMP OBJCONT_$91FC                            ;PROCESS NEXT OBJECT

GMOVD0:
9C77      20 AF E1       JSR $E1AF                                    ;DRAW THIS BRAIN
9C7A      4C FC 91       JMP OBJCONT_$91FC                            ;PROCESS NEXT OBJECT

; Electrode
9C7D      BD 91 1F       LDA SPRITE_STATE_$1F91,X
9C80      29 03          AND #$03
9C82      D0 08          BNE $9C8C
9C84      A9 00          LDA #$00
9C86      9D 8C 1E       STA SPRITE_TYPE_$1E8C,X                     ;Kill Electrode off
9C89      4C FC 91       JMP OBJCONT_$91FC                           ;PROCESS NEXT OBJECT
9C8C      29 02          AND #$02
9C8E      D0 08          BNE $9C98
9C90      A9 7F          LDA #$7F
9C92      9D 7D 1B       STA MTTBL,X
9C95      4C FC 91       OBJCONT_$91FC                               ;PROCESS NEXT OBJECT
9C98      BD D9 1C       LDA SATBL,X
9C9B      C9 01          CMP #$01
```

```
9C9D      10 0D          BPL $9CAC
9C9F      A9 02          LDA #$02
9CA1      20 95 E3       JSR DOTUNE_$E395                            ;Play Generic Explosion Sound
9CA4      A9 00          LDA #$00
9CA6      9D D9 1C       STA SATBL,X
9CA9      4C B0 9C       JMP $9CB0
9CAC      C9 02          CMP #$02
9CAE      F0 0B          BEQ $9CBB                                   ;If 2 frames have elapsed then
this
                                                                     ; electrode's dead and its time
to go
9CB0      FE D9 1C       INC SATBL,X
9CB3      A9 01          LDA #$01
9CB5      9D 7D 1B       STA MTTBL,X
9CB8      4C C1 9C       JMP $9CC1
9CBB      20 1E D7       JSR $D71E                                   ;Mark this sprite as truly dead
9CBE      4C FC 91       JMP OBJCONT_$91FC                           ;PROCESS NEXT OBJECT
9CC1      20 AF E1       JSR $E1AF
9CC4      4C FC 91       JMP OBJCONT_$91FC                           ;PROCESS NEXT OBJECT

9CC7                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9CD0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9CE0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9CF0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

9D00                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9D10                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9D20                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9D30                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9D40                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9D50                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9D60                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9D70                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9D80                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9D90                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9DA0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9DB0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9DC0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9DD0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9DE0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9DF0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

9E00                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9E10                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9E20                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9E30                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9E40                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9E50                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9E60                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9E70                                                                 .BYTE
```

```
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9E80                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9E90                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9EA0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9EB0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9EC0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9ED0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9EE0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9EF0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

9F00                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9F10                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9F20                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9F30                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9F40                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9F50                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9F60                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9F70                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9F80                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9F90                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9FA0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9FB0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9FC0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9FD0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9FE0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
9FF0                                                                     .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

;GRAPHICS DATA ($A000-$AFFF)
A000                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A010                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A020                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A030                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A040                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A050                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A060                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$03,$B0,$00,$0B
A070                    .BYTE $F8,$00,$0F,$EC,$00,$3F,$BF,$00,$3F,$BF,$00,$3E,$FE,$00,$3B,$FB
A080                    .BYTE $00,$2F,$EF,$00,$3F,$BF,$00,$03,$B0,$00,$00,$C0,$00,$00,$00,$00
A090                    .BYTE $C0,$C0,$AA,$80,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A0A0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A0B0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A0C0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A0D0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$FF,$FF,$FF
A0E0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
A0F0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

A100                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A110                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A120                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A130                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A140                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A150                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A160                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$02,$20,$00,$0E
```

```
A170                         .BYTE $EC,$00,$3E,$FB,$00,$80,$00,$80,$80,$00,$80,$C0,$00,$C0,$C0,$00
A180                         .BYTE $C0,$C0,$00,$C0,$00,$00,$00,$00,$00,$00,$00,$C0,$00,$00,$00,$00
A190                         .BYTE $F3,$C0,$80,$80,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A1A0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A1B0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A1C0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A1D0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$FF,$FF,$FF
A1E0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
A1F0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

A200                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A210                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A220                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A230                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A240                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A250                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A260                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$02,$60,$00,$09
A270                         .BYTE $08,$00,$0A,$A8,$00,$FB,$FB,$C0,$C0,$00,$C0,$80,$00,$C0,$C0,$00
A280                         .BYTE $C0,$C0,$00,$80,$C3,$30,$C0,$03,$30,$00,$00,$C0,$00,$00,$00,$00
A290                         .BYTE $C0,$C0,$A2,$80,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A2A0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A2B0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A2C0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A2D0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$FF,$FF,$FF
A2E0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
A2F0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

A300                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A310                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A320                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A330                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A340                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A350                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A360                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$02,$A0,$00,$09
A370                         .BYTE $18,$00,$09,$48,$00,$2A,$AA,$00,$FB,$FB,$C0,$FE,$FE,$C0,$BF,$BF
A380                         .BYTE $80,$EF,$EF,$C0,$00,$00,$00,$00,$80,$00,$00,$80,$00,$00,$00,$00
A390                         .BYTE $F3,$C0,$A2,$80,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A3A0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A3B0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A3C0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A3D0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$FF,$FF,$FF
A3E0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
A3F0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

A400                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A410                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A420                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A430                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A440                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A450                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A460                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$80,$00,$0A
A470                         .BYTE $A8,$00,$09,$48,$00,$25,$46,$00,$2A,$AA,$00,$2A,$AA,$00,$2A,$AA
A480                         .BYTE $00,$2A,$AA,$00,$2A,$AA,$00,$02,$A0,$00,$00,$80,$00,$00,$80,$00
A490                         .BYTE $C0,$C0,$80,$80,$FF,$AA,$AA,$FF,$00,$00,$00,$00,$00,$00,$00,$00
A4A0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$01,$00,$00,$00,$00,$00
A4B0                         .BYTE $00,$00,$00,$00,$00,$00,$01,$00,$00,$00,$00,$00,$00,$00,$00,$00
A4C0                         .BYTE $01,$00,$00,$00,$00,$00,$00,$00,$01,$00,$00,$00,$00,$00,$00,$00
A4D0                         .BYTE $01,$00,$01,$00,$01,$00,$00,$00,$00,$00,$00,$00,$00,$FF,$FF,$FF
A4E0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
A4F0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

A500                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A510                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A520                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A530                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A540                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A550                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A560                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A570                         .BYTE $80,$00,$08,$18,$00,$25,$46,$00,$20,$02,$00,$20,$02,$00,$20,$02
A580                         .BYTE $00,$20,$02,$00,$00,$00,$00,$00,$00,$00,$00,$40,$00,$00,$40,$00
A590                         .BYTE $A2,$80,$80,$80,$30,$82,$82,$0C,$FF,$C0,$AA,$80,$AA,$80,$2A,$00
A5A0                         .BYTE $08,$00,$00,$00,$00,$00,$00,$00,$00,$10,$01,$00,$10,$00,$00,$00
A5B0                         .BYTE $00,$00,$00,$00,$00,$10,$01,$00,$10,$00,$00,$00,$00,$00,$00,$10
```

```
A5C0                         .BYTE $01,$00,$10,$00,$00,$00,$00,$00,$01,$10,$10,$10,$00,$10,$00,$10
A5D0                         .BYTE $11,$00,$01,$00,$01,$00,$10,$00,$10,$00,$00,$00,$00,$FF,$FF,$FF
A5E0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
A5F0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

A600                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A610                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A620                         .BYTE $00,$00,$00,$FC,$FF,$F0,$FF,$FC,$00,$00,$00,$00,$00,$00,$00,$00
A630                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A640                         .BYTE $00,$00,$00,$00,$00,$00,$00,$03,$00,$0D,$C0,$1F,$D0,$3F,$F0,$3F
A650                         .BYTE $F0,$3F,$F0,$05,$40,$45,$44,$00,$00,$00,$00,$00,$00,$00,$00,$00
A660                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$02
A670                         .BYTE $A0,$00,$09,$48,$00,$20,$46,$00,$25,$46,$00,$25,$56,$00,$24,$56
A680                         .BYTE $00,$25,$16,$00,$25,$46,$00,$01,$40,$00,$00,$40,$00,$00,$40,$00
A690                         .BYTE $88,$80,$80,$80,$F3,$8A,$A2,$CF,$EA,$C0,$88,$80,$00,$00,$00,$00
A6A0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$10,$01,$00,$10,$00,$00,$00
A6B0                         .BYTE $00,$00,$00,$00,$00,$10,$01,$00,$10,$00,$00,$00,$00,$00,$00,$10
A6C0                         .BYTE $01,$00,$10,$00,$00,$00,$00,$00,$01,$10,$10,$10,$00,$10,$00,$10
A6D0                         .BYTE $11,$00,$01,$00,$01,$00,$10,$00,$10,$00,$00,$00,$00,$FF,$FF,$FF
A6E0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
A6F0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

A700                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A710                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$03,$00,$03,$00,$03
A720                         .BYTE $00,$FF,$C0,$FC,$3F,$F0,$C0,$00,$00,$00,$00,$00,$00,$00,$00,$00
A730                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A740                         .BYTE $00,$00,$00,$00,$00,$00,$00,$01,$00,$01,$00,$01,$00,$07,$40,$0F
A750                         .BYTE $C0,$00,$00,$45,$44,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A760                         .BYTE $15,$40,$00,$50,$50,$00,$40,$10,$00,$00,$00,$00,$00,$00,$00,$00
A770                         .BYTE $00,$00,$0A,$A8,$00,$25,$02,$00,$20,$46,$00,$21,$12,$00,$24,$42
A780                         .BYTE $00,$24,$46,$00,$00,$00,$00,$00,$40,$00,$00,$40,$00,$00,$80,$00
A790                         .BYTE $A2,$80,$80,$80,$30,$82,$A2,$CF,$FB,$C0,$A2,$80,$A2,$80,$22,$00
A7A0                         .BYTE $08,$00,$00,$00,$00,$00,$00,$00,$00,$40,$01,$00,$04,$00,$00,$00
A7B0                         .BYTE $00,$00,$00,$00,$00,$40,$01,$00,$04,$00,$00,$00,$00,$00,$00,$40
A7C0                         .BYTE $01,$00,$04,$00,$00,$00,$00,$00,$01,$40,$04,$40,$00,$40,$00,$40
A7D0                         .BYTE $05,$00,$01,$00,$01,$00,$04,$00,$04,$00,$00,$00,$00,$FF,$FF,$FF
A7E0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
A7F0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

A800                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A810                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$33,$30,$33,$30,$0F
A820                         .BYTE $C0,$FF,$C0,$FC,$3F,$F0,$CF,$FC,$00,$00,$00,$00,$03,$00,$00,$00
A830                         .BYTE $03,$00,$00,$00,$03,$00,$00,$3F,$C0,$00,$00,$0F,$FC,$0F,$FC,$00
A840                         .BYTE $00,$00,$00,$00,$00,$00,$00,$03,$00,$0D,$C0,$35,$70,$01,$00,$01
A850                         .BYTE $00,$C5,$4C,$00,$00,$04,$40,$00,$00,$00,$00,$00,$00,$00,$00,$00
A860                         .BYTE $50,$10,$00,$40,$10,$00,$10,$40,$00,$00,$00,$00,$00,$00,$00,$00
A870                         .BYTE $00,$00,$00,$00,$00,$2A,$AA,$00,$20,$46,$00,$24,$06,$00,$24,$42
A880                         .BYTE $00,$24,$46,$00,$20,$46,$00,$00,$40,$00,$00,$40,$00,$00,$80,$00
A890                         .BYTE $A2,$80,$A2,$80,$30,$82,$82,$0C,$FB,$C0,$A2,$80,$00,$00,$00,$00
A8A0                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$40,$01,$00,$04,$00,$00,$00
A8B0                         .BYTE $00,$00,$00,$00,$00,$40,$01,$00,$04,$00,$00,$00,$00,$00,$00,$40
A8C0                         .BYTE $01,$00,$04,$00,$00,$00,$00,$00,$01,$40,$04,$40,$00,$40,$00,$40
A8D0                         .BYTE $05,$00,$01,$00,$01,$00,$04,$00,$04,$00,$00,$00,$00,$FF,$FF,$FF
A8E0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
A8F0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

A900                         .BYTE $00,$54,$54,$54,$54,$04,$54,$54,$10,$54,$54,$A8,$A8,$A8,$A8,$08
A910                         .BYTE $A8,$A8,$20,$A8,$A8,$44,$00,$00,$00,$00,$00,$3F,$F0,$0C,$C0,$3F
A920                         .BYTE $F0,$FF,$C0,$FC,$0F,$F0,$CC,$0C,$FF,$FF,$FF,$FC,$0F,$C0,$00,$00
A930                         .BYTE $0C,$C0,$00,$00,$0F,$C0,$30,$0F,$C0,$03,$C0,$0C,$0C,$0C,$0C,$00
A940                         .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$03,$00,$01,$00,$75,$74,$C5
A950                         .BYTE $4C,$05,$40,$05,$40,$00,$00,$00,$00,$00,$00,$00,$00,$15,$40,$00
A960                         .BYTE $44,$50,$00,$10,$40,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
A970                         .BYTE $00,$00,$00,$B0,$00,$00,$80,$00,$25,$02,$00,$25,$16,$00,$20,$16
A980                         .BYTE $00,$21,$52,$00,$00,$00,$00,$00,$00,$00,$00,$40,$00,$00,$C0,$00
A990                         .BYTE $80,$80,$A2,$80,$30,$82,$82,$0C,$EA,$C0,$80,$80,$80,$80,$22,$00
A9A0                         .BYTE $08,$00,$08,$00,$01,$00,$01,$54,$01,$00,$01,$00,$01,$00,$55,$00
A9B0                         .BYTE $01,$00,$01,$54,$01,$00,$01,$00,$01,$00,$55,$00,$01,$00,$01,$54
A9C0                         .BYTE $01,$54,$01,$54,$55,$54,$01,$54,$01,$00,$01,$00,$55,$00,$01,$00
A9D0                         .BYTE $01,$00,$55,$00,$01,$00,$55,$00,$01,$00,$55,$00,$00,$FF,$FF,$FF
A9E0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
A9F0                         .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
```

```
AA00                        .BYTE $00,$44,$10,$40,$04,$04,$04,$44,$10,$44,$04,$88,$20,$80,$08,$08
AA10                        .BYTE $08,$88,$20,$88,$08,$10,$FF,$CC,$C3,$0F,$C0,$0F,$C0,$33,$30,$3F
AA20                        .BYTE $F0,$FF,$C0,$FC,$0F,$F0,$CC,$CC,$57,$57,$57,$F4,$0F,$C0,$03,$00
AA30                        .BYTE $33,$30,$03,$00,$0F,$C0,$30,$0F,$C0,$03,$C0,$0C,$CC,$0C,$CC,$00
AA40                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$0D,$C0,$0F,$C0,$35
AA50                        .BYTE $70,$00,$00,$00,$00,$00,$00,$00,$00,$00,$05,$00,$00,$10,$40,$00
AA60                        .BYTE $44,$10,$00,$04,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
AA70                        .BYTE $00,$00,$00,$A0,$00,$02,$A0,$00,$2A,$AA,$00,$2A,$AA,$00,$2A,$AA
AA80                        .BYTE $00,$2A,$AA,$00,$2A,$AA,$00,$02,$A0,$00,$00,$80,$00,$00,$00,$00
AA90                        .BYTE $80,$80,$88,$80,$30,$82,$82,$0C,$EA,$C0,$80,$80,$00,$00,$00,$00
AAA0                        .BYTE $00,$00,$00,$00,$01,$00,$01,$54,$01,$00,$01,$00,$01,$00,$55,$00
AAB0                        .BYTE $01,$00,$01,$54,$01,$00,$01,$00,$01,$00,$55,$00,$01,$00,$01,$54
AAC0                        .BYTE $01,$54,$01,$54,$55,$54,$01,$54,$01,$00,$01,$00,$55,$00,$01,$00
AAD0                        .BYTE $01,$00,$55,$00,$01,$00,$55,$00,$01,$00,$55,$00,$00,$FF,$FF,$FF
AAE0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
AAF0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

AB00                        .BYTE $00,$44,$10,$40,$04,$04,$04,$44,$10,$44,$04,$88,$20,$80,$08,$08
AB10                        .BYTE $08,$88,$20,$88,$08,$54,$CC,$CC,$C3,$0C,$00,$FF,$FC,$CC,$CC,$FF
AB20                        .BYTE $FC,$FF,$C0,$FC,$03,$F0,$CC,$CC,$7F,$77,$77,$F4,$3F,$F0,$0F,$C0
AB30                        .BYTE $0C,$C0,$0C,$C0,$0F,$C0,$30,$03,$C0,$00,$C0,$0C,$CC,$0C,$C0,$07
AB40                        .BYTE $57,$57,$C0,$00,$00,$00,$00,$00,$00,$00,$00,$03,$00,$35,$70,$C5
AB50                        .BYTE $4C,$05,$40,$05,$40,$00,$00,$05,$00,$00,$05,$00,$00,$15,$40,$00
AB60                        .BYTE $45,$10,$00,$15,$40,$00,$50,$50,$00,$FE,$AA,$A0,$00,$00,$00,$00
AB70                        .BYTE $00,$00,$00,$00,$00,$02,$20,$00,$00,$80,$00,$00,$80,$00,$00,$80
AB80                        .BYTE $00,$00,$80,$00,$00,$00,$00,$00,$80,$00,$00,$80,$00,$00,$00,$00
AB90                        .BYTE $80,$80,$F3,$C0,$F3,$8A,$A2,$CF,$EA,$C0,$80,$80,$80,$80,$22,$00
ABA0                        .BYTE $08,$00,$08,$00,$01,$40,$01,$00,$01,$00,$01,$00,$01,$00,$01,$00
ABB0                        .BYTE $05,$00,$00,$40,$00,$40,$00,$40,$00,$40,$00,$40,$04,$40,$00,$00
ABC0                        .BYTE $00,$00,$00,$00,$00,$00,$04,$00,$00,$00,$00,$00,$00,$00,$04,$00
ABD0                        .BYTE $00,$00,$00,$00,$04,$00,$00,$00,$04,$00,$04,$00,$14,$FF,$FF,$FF
ABE0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
ABF0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

AC00                        .BYTE $00,$44,$10,$40,$04,$54,$04,$44,$10,$44,$54,$88,$20,$80,$08,$A8
AC10                        .BYTE $08,$88,$20,$88,$A8,$10,$CC,$CC,$CC,$CC,$00,$0F,$C0,$33,$30,$3F
AC20                        .BYTE $F0,$FF,$C0,$FC,$03,$F0,$CC,$CC,$57,$77,$57,$54,$0F,$C0,$03,$00
AC30                        .BYTE $33,$30,$03,$00,$0F,$C0,$30,$00,$C0,$00,$00,$0C,$CC,$0C,$C0,$0F
AC40                        .BYTE $77,$77,$C0,$0F,$77,$77,$C0,$00,$00,$00,$00,$00,$00,$0D,$C0,$0F
AC50                        .BYTE $C0,$CF,$CC,$00,$00,$00,$00,$00,$00,$00,$05,$00,$00,$10,$40,$00
AC60                        .BYTE $41,$10,$00,$01,$00,$00,$00,$00,$00,$32,$22,$20,$00,$00,$00,$00
AC70                        .BYTE $00,$00,$00,$00,$00,$02,$E0,$00,$0A,$A8,$00,$0A,$A8,$00,$0A,$A8
AC80                        .BYTE $00,$0A,$A8,$00,$0A,$A8,$00,$02,$A0,$00,$00,$80,$00,$00,$00,$00
AC90                        .BYTE $80,$80,$C0,$C0,$30,$82,$82,$0C,$EA,$C0,$80,$80,$00,$00,$00,$00
ACA0                        .BYTE $00,$00,$00,$00,$01,$40,$01,$00,$01,$00,$01,$00,$01,$00,$01,$00
ACB0                        .BYTE $05,$00,$00,$40,$00,$40,$00,$40,$00,$40,$00,$40,$04,$40,$00,$00
ACC0                        .BYTE $00,$00,$00,$00,$00,$00,$04,$00,$00,$00,$00,$00,$00,$00,$04,$00
ACD0                        .BYTE $00,$00,$00,$00,$04,$00,$00,$00,$04,$00,$04,$00,$7D,$FF,$FF,$FF
ACE0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
ACF0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

AD00                        .BYTE $00,$44,$10,$54,$14,$44,$54,$54,$04,$54,$44,$88,$20,$A8,$28,$88
AD10                        .BYTE $A8,$A8,$08,$A8,$88,$54,$CC,$CF,$CC,$CF,$00,$3F,$F0,$0C,$C0,$3F
AD20                        .BYTE $F0,$FF,$C0,$FC,$00,$F0,$CF,$CC,$F7,$77,$77,$74,$0F,$C0,$00,$00
AD30                        .BYTE $0C,$C0,$00,$00,$0F,$C0,$00,$00,$C0,$00,$00,$CF,$CC,$0F,$C0,$07
AD40                        .BYTE $77,$57,$40,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$03,$00,$35
AD50                        .BYTE $70,$00,$00,$45,$44,$04,$40,$00,$00,$00,$00,$00,$00,$15,$40,$00
AD60                        .BYTE $51,$10,$00,$10,$40,$00,$00,$00,$00,$32,$22,$20,$00,$00,$00,$00
AD70                        .BYTE $00,$00,$00,$00,$00,$02,$A0,$00,$08,$08,$00,$08,$08,$00,$08,$08
AD80                        .BYTE $00,$08,$08,$00,$00,$00,$00,$00,$00,$00,$00,$80,$00,$00,$00,$00
AD90                        .BYTE $A2,$80,$F3,$C0,$F3,$8A,$A2,$CF,$FB,$C0,$A2,$80,$A2,$80,$22,$00
ADA0                        .BYTE $08,$00,$00,$00,$01,$10,$01,$00,$01,$00,$01,$00,$01,$00,$01,$00
ADB0                        .BYTE $11,$00,$00,$10,$00,$10,$00,$10,$00,$10,$00,$10,$10,$10,$00,$00
ADC0                        .BYTE $00,$00,$00,$00,$00,$00,$10,$00,$00,$00,$00,$00,$00,$00,$10,$00
ADD0                        .BYTE $00,$00,$00,$00,$10,$00,$00,$00,$10,$00,$10,$00,$7D,$FF,$FF,$FF
ADE0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
ADF0                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

AE00                        .BYTE $00,$44,$10,$04,$04,$44,$40,$40,$04,$44,$44,$88,$20,$08,$08,$88
AE10                        .BYTE $80,$80,$08,$88,$88,$DC,$C0,$CC,$CC,$CC,$00,$33,$30,$33,$30,$0F
AE20                        .BYTE $C0,$FF,$C0,$FC,$00,$F0,$C0,$0C,$57,$57,$57,$7C,$03,$00,$00,$00
AE30                        .BYTE $03,$00,$00,$00,$03,$00,$00,$00,$00,$00,$00,$C0,$0C,$00,$00,$00
AE40                        .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$0D
```

```
AE50                    .BYTE $C0,$0D,$C0,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
AE60                    .BYTE $40,$50,$00,$40,$10,$00,$10,$40,$00,$32,$22,$20,$00,$00,$00,$00
AE70                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$0B,$38,$00,$0B,$38,$00,$0B,$08
AE80                    .BYTE $00,$08,$08,$00,$0B,$38,$00,$03,$30,$00,$00,$C0,$00,$00,$00,$00
AE90                    .BYTE $A2,$80,$C0,$C0,$30,$82,$82,$0C,$FB,$C0,$A2,$80,$00,$00,$00,$00
AEA0                    .BYTE $00,$00,$00,$00,$01,$10,$01,$00,$01,$00,$01,$00,$01,$00,$01,$00
AEB0                    .BYTE $11,$00,$00,$10,$00,$10,$00,$10,$00,$10,$00,$10,$10,$10,$00,$00
AEC0                    .BYTE $00,$00,$00,$00,$00,$00,$10,$00,$00,$00,$00,$00,$00,$00,$10,$00
AED0                    .BYTE $00,$00,$00,$00,$10,$00,$00,$00,$10,$00,$10,$00,$7D,$FF,$FF,$FF
AEE0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
AEF0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

AF00                    .BYTE $00,$54,$50,$54,$54,$40,$50,$50,$54,$54,$54,$A8,$A0,$A8,$A8,$80
AF10                    .BYTE $A0,$A0,$A8,$A8,$A8,$FC,$C0,$CF,$CC,$CF,$C0,$03,$00,$03,$00,$03
AF20                    .BYTE $00,$FF,$C0,$FC,$00,$30,$FF,$FC,$FF,$FF,$FF,$FC,$00,$00,$00,$00
AF30                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$FF,$FC,$00,$00,$00
AF40                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$03
AF50                    .BYTE $00,$03,$00,$0F,$C0,$45,$44,$00,$00,$00,$00,$00,$00,$00,$00,$00
AF60                    .BYTE $15,$40,$00,$50,$50,$00,$40,$10,$00,$F2,$AA,$A0,$00,$00,$00,$00
AF70                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$0A,$A8,$00,$0A,$A8,$00,$0A,$A8
AF80                    .BYTE $00,$0A,$A8,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
AF90                    .BYTE $AA,$80,$F3,$C0,$FF,$AA,$AA,$FF,$FF,$C0,$AA,$80,$AA,$80,$2A,$00
AFA0                    .BYTE $08,$00,$00,$00,$01,$00,$01,$00,$01,$00,$01,$00,$01,$00,$01,$00
AFB0                    .BYTE $01,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
AFC0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
AFD0                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$14,$FF,$FF,$FF
AFE0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
AFF0                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

***********************************************************
***********************************************************
*                                                         *
*                                                         *
*    ROBOTRON    29-JULY-83                               *
*                 2-AUGUST-83           12:00             *
*                22-AUGUST-83            8:10             *
*                                                         *
*                                                         *
*         RWAVE.S    - ROBOTRON WAVE-RELATED ROUTINES     *
*                                                         *
***********************************************************

*
*********************************************************************************
*                                                                               *
*     WAVESTRT-- SUBROUTINE TO INITIALIZE THINGS BEFORE EACH WAVE               *
*                                                                               *
*             NOTE:  THERE WILL HAVE TO BE AN ENTRY POINT INTO THIS             *
*     ROUTINE THAT WILL RESTART A WAVE WHEN THE MC DIES AND THEN RETURNS.       *
*     THIS POINT SHOULD BE AFTER THE INC OF WAVENUM AND AFTER THE LOAD          *
*     OF THE WAVE-START NUMBERS FROM ROM.                                       *
*                                                                               *
*********************************************************************************
*

;Integrated some of disassembly from Dan Boris & "Scotty" ($B000-$B08A).
WAVESTRT:
B000     E6 D5          INC WAVENUM                                 ;A NEW WAVE
B002     F8             SED
B003     A6 61          LDX $61                                     ;GET PLAYER NUMBER
B005     18             CLC                                         ;
B006     B5 E8          LDA $E8,X                                   ;GERT PLAYERS CURRENT LEVEL
B008     69 01          ADC #$01                                    ;INCREMENT TO NEXT LEVEL
B00A     95 E8          STA $E8,X                                   ;SAVE IT
B00C     D8             CLD                                         ;

*         INITIALIZE NUMBERS OF EACH OBJECT:
*         GET THE BLOCK OF STARTNUM NUMBERS FROM WAVETBL AND MOVE IT INTO
*           THE RAM LOCATIONS STARTING WITH GNUM.
*         HERE A Z80 WOULD BE NICE

B00D     A9 00          LDA #$00
B00F     85 A2          STA TEMP2                 ;TEMP2 WILL BE USED FOR HI-BYTE OF WAVENUM*16
```

```
B011      A5 D5          LDA WAVENUM                              ;GET CURRENT LEVEL, 1 TO 255

*         TURN WAVE NUMBERS OVER 40 INTO THE RANGE 20 - 40
WS1:
B013      C9 29          CMP #41                                  ;LESS THAN 41?
B015      90 08          BCC WS2                                  ;BRANCH IF LESS THAN 40
B017      38             SEC
B018      E9 14          SBC #20                                  ;CHOP OFF 20
B01A      85 D5          STA WAVENUM                              ;
B01C      4C 13 B0       JMP WS1                                  ;AND TRY AGAIN

*         NOW ACCUMULATOR IS A WAVE # 1 - 40.  MULTIPLY IT BY 16 TO GET WAVETBL OFFSET

WS2:
B01F      0A             ASL A                                    ;A = A * 16
B020      26 A2          ROL TEMP2                                ;Accumulator carries into
$A2, which will be
                                                                  ; used to compute high byte of
level data.
B022      0A             ASL A
B023      26 A2          ROL TEMP2
B025      0A             ASL A
B026      26 A2          ROL TEMP2
B028      0A             ASL A                                    ;NOW ACC HAS LOW BYTE OF
PRODUCT
B029      26 A2          ROL TEMP2                                ;TEMP2 HAS HIGH BYTE OF
PRODUCT

*         ADD OFFSET (JUST COMPUTED) TO #WAVETBL TO FIND BLOCK BASE ADDRESS

B02B      18             CLC
B02C      69 0A          ADC #L(WAVETBL-$10)                      ;WAVETBL STARTS WITH WAVE 1,
NOT 0
B02E      85 BA          STA TEMP0                                ;LOW BYTE OF BLOCK BASE
ADDRESS
B030      A5 A2          LDA TEMP2                                ;ADD HIGH BYTE
B032      69 EF          ADC #H(WAVETBL-$10)
B034      85 BB          STA TEMP1                                ;HIGH BYTE OF BLOCK BASE
ADDRESS

B036      E6 6E          INC $6E                                  ; INCREMENT A SKILL VARIABLE

B038      A5 6E          LDA $6E
B03A      C5 73          CMP $73
B03C      90 07          BCC $B045
B03E      A5 73          LDA $73
B040      38             SEC
B041      E9 14          SBC #$14                                 ;GO BACK 20 LEVELS TOO
B043      85 6E          STA $6E

*         NOW (TEMP0) IS BLOCK BASE ADDRESS.  LOOP THRU NUMBERS TO MOVE WITH Y

B045      A0 0F          LDY #STARTNUM                            ;NUMBER OF NUMBERS TO MOVE

WLOOP1:
B047      B1 BA          LDA (TEMP0),Y                            ;GET A NUMBER FROM WAVETBL IN
ROM
B049      99 06 19       STA GNUM,Y                               ;STORE IT IN RAM
B04C      88             DEY
B04D      10 F8          BPL WLOOP1_$B047                         ;Y NON-NEGATIVE - MORE TO MOVE

**********NOW THE RAM VARIABLES ARE LOADED WITH START-OF-WAVE NUMBERS

*         ZERO OUT THE OBJECT DATA TABLES
*                 LOOP FROM XTBL TO XTBL+MAXOBJS*NUMTBLS
*         THIS WILL WIPE OUT THE ENTIRE TOP PAGE OF THE TABLES, EVEN
*                 ABOVE XTBL+MAXOBJS*NUMTBLS

B04F      C8             INY
B050      84 6D          STY $6D
B052      F0 04          BEQ $B058
B054      A9 01          LDA #$01
B056      85 6D          STA $6D
```

```
WSZLOOP:
B058      A9 00          LDA #$00                                  ;ZERO OUT OBJECT DATA TABLES
LOOP
B05A      A0 57          LDY $57
B05C      99 91 1F       STA SPRITE_STATE_$1F91,Y                  ;Clear sprite enable table
B05F      99 8C 1E       STA SPRITE_TYPE_$1E8C,Y                   ;Clear sprite type table
B062      99 E8 1F       STA $1FE8,Y
B065      99 87 1D       STA $1D87,Y
B068      99 DE 1D       STA $1DDE,Y
B06B      99 35 1E       STA $1E35,Y
B06E      88             DEY
B06F      10 EB          BPL $B05C

*          NOW TEMP1 (HI BYTE OF INDEX) IS ABOVE UPPER LIMIT, SO WE ARE DONE


*          SET UP EACH TYPE OF OBJECT INDIVIDUALLY:
*              SET OBJECT DATA TABLE SEGMENT POINTER (FPTR, HPTR, ETC.)
*              CREATE SPECIFIED NUMBER OF OBJECTS IN OBJECT DATA TABLES
*                       ( IF START-OF-WAVE NUMBER IS 0, CREATE 1 OBJECT
*                         SO OTHER LOOPS WON'T GET SCREWED UP )
*              SET ALL OBJECT DATA TABLE VARIABLES
*              SET ALL VARIABLES GLOBAL TO A CERTAIN OBJECT TYPE (I.E. GSPEED)
*
*          X IS A RUNNING POINTER INTO THE OBJECT DATA TABLES

B071      A5 72          LDA $72
B073      85 EC          STA $EC
B075      A9 FF          LDA #$FF
B077      A0 07          LDY $07

B079      99 26 19       STA $1926,Y
B07C      88             DEY
B07D      10 FA          BPL $B079
B07F      A2 01          LDX $01                                   ;Start with sprite #1

*          INITIALIZE NUMBER OF CREATURES LEFT

B081      A9 00          LDA #$00
B083      85 C9          STA CRELEFT
*
**********GRUNTS
*          NO NEED TO SET POINTER - OBJECT 1 IS ALWAYS THE FIRST GRUNT

B085      AC 06 19       LDY GNUM                                  ;LOOP THRU ALL GRUNTS
B088      D0 03          BNE WSGGO                                 ;AT LEAST 1 GRUNT - DISTRIBUTE
GRUNTS
B08A      4C D3 B0       JMP WSGCONT                               ;GO TO GLOBAL GRUNT VARIABLE
SETUP

WSGGO:
B08D      88             DEY                                       ;Y INDEXES UNTIL NEGATIVE -
FIX FENCEPOST ERROR

WSGLOOP:
B08E      20 D8 D3       JSR RANDXYBX_$D3D8                        ;GET A VALID GRUNT POSITION
B091      A5 C3          LDA RANDOMX
B093      9D CF 1A       STA XTBL,X                                ;GRUNT XPOS
B096      18             CLC
B097      69 07          ADC #GWID                                 ;COMPUTE EXTENT, X EXTENT =
XPOS + $07
B099      9D E3 1E       STA XEXTBL,X                              ;GRUNT X EXTENT
B09C      A5 C4          LDA RANDOMY
B09E      9D 26 1B       STA YTBL,X                                ;GRUNT YPOS
B0A1      18             CLC
B0A2      69 0C          ADC #GHEIGHT                              ;COMPUTE EXTENT, Y EXTENT =
YPOS + $0C
B0A4      9D 3A 1F       STA YEXTBL,X                              ;GRUNT Y EXTENT
B0A7      98             TYA                                       ;USE GRUNT # AS SEED TO GET
GOOD DISTR
B0A8      29 0F          AND #MASK3                                ;GET A NUMBER 0 - 7
B0AA      18             CLC
```

```
B0AB      69 08          ADC #WSWAIT
B0AD      9D 7D 1B       STA MTTBL,X                              ;NUMBER OF FRAMES UNTIL MOVE
B0B0      20 14 D4       JSR RAND2                                ;GET A NUMBER, 0 - 2
B0B3      18             CLC
B0B4      69 01          ADC #$01                                 ;NUMBER, 1 - 3
B0B6      9D D9 1C       STA SATBL,X                              ;GRUNT ANIMATION STEP
B0B9      A9 00          LDA #$00
BOBB      9D D4 1B       STA DXTBL,X                              ;GRUNT DX (DIR) MUST ALWAYS BE
0
BOBE      A9 01          LDA #GCODE
B0C0      9D 8C 1E       STA CRTBL,X                              ;SET SPRITE TYPE TO GRUNT
B0C3      A9 01          LDA #$01
B0C5      9D 91 1F       STA $1F91,X
B0C8      A9 00          LDA #$00
B0CA      9D E8 1F       STA $1FE8,X

*          DLPHTBL,DLPLTBL AND DL2PTBL WILL BE SET UP BY THE LOAD AT THE END OF
*                  THE WAVESTRT ROUTINE
*          DONE WITH THIS GRUNT, ON TO NEXT...

B0CD      E6 C9          INC CRELEFT                              ;ANOTHER LIVING CREATURE IS
CREATED
B0CF      E8             INX                                      ;NEXT SPRITE LOCATION
B0D0      88             DEY                                      ;COUNT DOWN UNTIL GRUNTS DONE
B0D1      10 BB          BPL WSGLOOP_B08E                         ;MORE GRUNTS TO SET UP

*          SET$GSPEED - NUMBER OF FRAMES BETWEEN GRUNT MOVES AT START
*                OF WAVE    - THIS VARIABLE WILL BE CHANGED DURING THE WAVE
WSGCONT:
B0D3      A4 6E          LDY $6E
B0D5      B9 76 ED       LDA $ED76,Y
B0D8      A8             TAY                                      ;PUT WAVE NUMBER IN Y
B0D9      A5 6D          LDA GSPTBL-1,Y      ;LOAD STARTING GSPEED - USE -1 BECAUSE NO WAVE 0
B0DB      0A             ASL A
B0DC      85 CC          STA GSPEED
B0DE      98             TYA
B0DF      38             SEC
B0E0      E5 CC          SBC GSPEED
B0E2      85 CC          STA GSPEED
B0E4      86 DF          STX $DF                                  ;SAVE SPRITE INDEX OF END OF
GRUNTS

;Disassembly of $B0E6-$B149 compliments of Dan Boris & "Scotty"
;Electrodes

B0E6      AC 15 19       LDY $1915                                ;Get number of electrodes
B0E9      D0 03          BNE $B0EE                                ;Branch if there are any
B0EB      4C 4B B1       JMP $B14B                                ;Jump if not

B0EE      84 A3          STY TEMP3                                ;Save number of electrodes
B0F0      AC 14 19       LDY $1914                                ;Get type of electrode for this
screen
B0F3      B9 34 ED       LDA $ED34,Y                              ;Get electrode width from
table
B0F6      85 A0          STA TEMP0                                ;
B0F8      B9 3C ED       LDA $ED3C,Y                              ;Get electrode height from
table
B0FB      85 A1          STA TEMP1
B0FD      A4 A3          LDY TEMP3

B0FF      20 D8 D3       JSR RANDXYBX_$D3D8                       ;Get random screen position
B102      A5 C3          LDA $C3                                  ;Get random X-pos
B104      9D CF 1A       STA SPRITE_X,X                           ;Store in Sprite table
B107      18             CLC                                      ;
B108      65 A0          ADC TEMP0                                ;Add electrode width
B10A      9D E3 1E       STA SPRITE_X_EXTENT,X                    ;Store in sprite table
B10D      A5 C4          LDA $C4                                  ;Get random Y-pos
B10F      9D 26 1B       STA SPRITE_Y,X                           ;Store in sprite table
B112      18             CLC                                      ;
B113      65 A1          ADC TEMP1                                ;Add electrode height
B115      9D 3A 1F       STA SPRITE_Y_EXTENT,X                    ;Store in sprite table
B118      8A             TXA                                      ;
B119      A8             TAY                                      ;
```

```
B11A     88             DEY                                         ;Put previous sprite index
into Y
B11B     F0 0A          BEQ $B127                                   ;skip if we are on the first
sprite

B11D     20 5D DC       JSR $DC5D                                   ;see if this sprite overlaps a
previous one
B120     A5 B7          LDA FRMCNT                                  ;get result
B122     D0 DB          BNE $B0FF                                   ;If they overlap, try again
B124     88             DEY                                         ;goto next previous sprite
B125     D0 F6          BNE $B11D                                   ;Branch if we are not done
B127     AD 14 19       LDA $1914                                   ;Get electrode style
B12A     9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X                  ;Store in sprite table
B12D     A9 10          LDA #$10                                    ;
B12F     9D 8C 1E       STA SPRITE_TYPE_$1E8C,X                     ;Set sprite type to Electrode
B132     A9 01          LDA #$01                                    ;
B134     9D 91 1F       STA SPRITE_STATE_$1F91,X                    ;Enable sprite
B137     A9 00          LDA #$00                                    ;
B139     9D D9 1C       STA SATBL,X                                 ;Set animation frame
B13C     A9 00          LDA #$00
B13E     9D E8 1F       STA $1FE8,X
B141     A9 7F          LDA #$7F
B143     9D 7D 1B       STA MTTBL,X
B146     E8             INX                                         ;Next sprite position
B147     C6 A3          DEC TEMP3                                   ;Decrement number of
Electrodes
B149     D0 B4          BNE $B0FF                                   ;Branch if not done

**********FAMILY
;Setup Human Family - Human AI Handler routine is at $94A0.

;Integrated most of the disassembly from Dan Boris & "Scotty" ($B14B-$B1AB).
WSG1:                                                               ;DONE WITH GRUNT SETUP
B14B     86 D6          STX FPTR                                    ;SET POINTER TO START OF
FAMILY
                                                                    ;Save index of first Human
B14D     18             CLC                                               ;
B14E     AD 09 19       LDA $1909                                   ;Get number of Mikeys
B151     6D 08 19       ADC $1908                                   ;add to number of Daddies
B154     6D 07 19       ADC $1907                                   ;add to number of Mommies
B157     85 5B          STA $5B                                           ;Save
B159     AD 07 19       LDA MONUM                                   ;Get number of Mommies
B15C     D0 0D          BNE WSFLOOP_B16B                            ;Branch if there are any
B15E     AD 08 19       LDA DNUM                                    ;Get number of Daddies
B161     D0 08          BNE WSFLOOP_B16B                            ;Branch if there are any
B163     AD 09 19       LDA MINUM                                   ;Get number of Mikeys
B166     D0 03          BNE WSFLOOP_B16B                            ;Branch if there are any
B168     4C FF B1       JMP WSFCONT_B1FF                            ;No humans, done

WSFLOOP:
B16B     A9 02          LDA #MOCODE                                 ;TEMP2 IS FAMILY FINISHED FLAG
- MO,
B16D     85 A2          STA TEMP2                                   ; D AND MI ROUTINES DEC THIS IF
NO
                                                                    ; MO, D OR MI TO SET UP.  IF IT
GOES
                                                                    ; NEGATIVE, WE ARE DONE
B16F     CE 07 19       DCE MONUM                                   ;Decrement number of Mommies
B172     10 05          BPL ADDMOMMY_$B179                          ;Branch if there are still
some left
B174     C6 A2          DEC TEMP2                                   ;No Mommies left
B176     4C 81 B1       JMP WSFCHKD_$B181

ADDMOMMY:
B179     A9 02          LDA #DCODE                                  ;
B17B     9D 8C 1E       STA CRTBL,X                                 ;Set sprite type to Mommy
B17E     20 AC B1       JSR ADDFAM_$B1AC                            ;Put Mommy in sprite table

WSFCHKD:
B181     CE 08 19       DEC DNUM                                    ;Decrement number of Daddies
B184     10 05          BPL ADDDADDY_$B18B                          ;Branch if there are still
some left
B186     C6 A2          DEC TEMP2                                   ;No Daddies left
```

```
B188      4C 93 B1       JMP WSFCHKMI_$B193

ADDDADDY:
B18B      A9 03          LDA #DCODE                                  ;
B18D      9D 8C 1E       STA CRTBL,X                                 ;Set sprite type to Daddy
B190      20 AC B1       JSR ADDFAM_$B1AC                            ;put Daddy in sprite table

WSFCHKMI:
B193      CE 09 19       DEC MINUM                                   ;Decrement number of Mikeys
B196      10 05          BPL ADDMIKEY_$B19D                          ;Branch if there are still
some left
B198      C6 A2          DEC TEMP2                                   ;No Mikeys left
B19A      4C A5 B1       JMP WSF1_$B1A5

ADDMIKEY:
B19D      A9 04          LDA #MICODE
B19F      9D 8C 1E       STA CRTBL,X                                 ;Set sprite type to Mikey
B1A2      20 AC B1       JSR ADDFAM_$B1AC                            ;Put Mikey in sprite table

WSF1:
B1A5      A5 A2          LDA TEMP2                                   ;Done with all humans?
B1A7      10 C2          BPL WSFLOOP_$B16B                           ;Branch if not
B1A9      4C FF B1       JMP WSFCONT_$B1FF                           ;Done

**********************************************************************************
*         SUBROUTINE TO PUT A FAMILY MEMBER INTO OBJECT DATA TABLES WITH STATE
*           VARIABLES SET UP.  ASSUME CRTBL,X ALREADY LOADED WITH CORRECT CODE
*
;Integrated most of the disassembly from Dan Boris & "Scotty" ($B000-$B08A).
ADDFAM:
B1AC      20 BD D3       JSR RANDXY_$D3BD                            ;Get random screen position
B1AF      A5 C3          LDA RANDOMX                                 ;Get X-pos
B1B1      9D CF 1A       CTA XTBL,X                                  ;Put it in sprite table
B1B4      18             CLC                                         ;
B1B5      69 04          ADC #FWID                                   ;X-pos = X-pos + 4 (width)
B1B7      9D E3 1E       STA XEXTBL,X                                ;Store in sprite table
B1BA      A5 C4          LDA RANDOMY                                 ;Get Y-pos
B1BC      9D 26 1B       STA YEXTBL,X                                ;Put it in sprite table
B1BF      18             CLC                                         ;
B1C0      69 0B          ADC #FHEIGHT                                ;Y-pos = Y-pos + 11 (height)
B1C2      9D 3A 1F       STA YEXTBL,X                                ;Put it in sprite table
B1C5      A4 DF          LDY $DF                                     ;Get sprite index of
electrodes
B1C7      C4 D6          CPY $D6                                     ;Compare with current sprite
index
B1C9      F0 0C          BEQ $B1D7                                   ;If they are equal, then we've
reached
                                                                     ; the end of our buffer
B1CB      20 5D DC       JSR $DC5D                                   ;See if sprite overlaps any
electrodes
B1CE      A5 B7          LDA FRMCNT                                  ;Get result
B1D0      D0 DA          BNE $B1AC                                   ;Try again if it overlaps
B1D2      C8             INY                                         ;Next electrode
B1D3      C4 D6          CPY $D6                                     ;Done with electrodes yet
B1D5      90 F4          BCC $B1CB                                   ;Branch if not

B1D7      20 A8 D3       JSR RANDOM                                  ;Get random number
B1DA      29 07          AND #$07                                    ;Number from 0 - 7
B1DC      9D 82 1C       STA MTTBL,X                                 ;NUMBER OF FRAMES UNTIL MOVE
B1DF      8A             TXA
B1E0      29 0F          AND #$0F
B1E2      9D 7D 1B       STA MTTBL,X
B1E5      20 14 D4       JSR RAND2                                   ;Get random number from 0 to 2
B1E8      9D D9 1C       STA SATBL,X                                 ;Sprite animation frame
B1EB      20 A8 D3       JSR RANDOM_$D3A8                            ;Get random number
B1EE      29 07          AND #$07                                    ;Number from 0 -7
B1F0      9D D4 1B       STA DXTBL,X                                 ;STORE IN DIRECTION HUMAN IS
FACING
B1F3      A9 01          LDA #$01                                    ;
B1F5      9D 91 1F       STA SPRITE_STATE_$1F91,X                    ;Enabled sprite
B1F8      A9 00          LDA #$00
B1FA      9D E8 1F       STA $1FE8,X
B1FD      E8             INX                                         ;Next human
```

```
B1FE      60              RTS

WSFCONT:
B1FF      A9 0A           LDA #$0A                                        ;WE WANT TO USE A TABLE LOOKUP
B201      85 CD           STA FSPEED                                      ;FAMILY SPEED
B203      A9 00           LDA #$00
B205      85 D4           STA FAMLEVEL                                    ;RESET SCORE LEVEL FOR PICKING
UP FAMILY

**********HULKS
;Integrated most of the disassembly from Dan Boris & "Scotty" ($B207-$B276).
;Setup Hulks - Hulk AI Handler routine is at $95C6.
B207      86 D7           STX HPTR                                        ;SET POINTER TO START OF HULKS
B209      AC 0A 19        LDY HNUM                                        ;Get number of Hulks
B20C      D0 03           BNE WSHGO_$B211                                 ;Branch if there are some
B20E      4C 78 B2        JMP WSHCONT_$B278                               ;Otherwise if we have no
Hulks, do our Spheroids

WSHGO:
B211      A4 6E           LDY $6E                                         ;Read skill
B213      B9 BC ED        LDA $EDBC,Y
B216      85 CE           STA $CE
B218      B9 02 EE        LDA $EE02,Y
B21B      85 A0           STA TEMP0
B21D      AC 0A 19        LDY HNUM                                        ;Get number of Hulks
B220      88              DEY                                             ; Y MUST DECREMENT UNTIL
NEGATIVE

WSHLOOP:
B221      20 D8 D3        JSR RANDXYBX_$D3D8                              ;Get random screen position
B224      A5 C3           LDA RANDOMX                                     ;Get result
B226      9D CF 1A        STA XTBL,X                                      ;Write X position to sprite
table
B229      18              CLC                                             ;
B22A      69 07           ADC #HWID                                       ;Width
B22C      9D E3 1E        STA XEXTBL,X                                    ;Store in sprite table
B22F      A5 C4           LDA RANDOMY                                     ;Get Y component from random
screen pos
B231      9D 26 1B        STA YTBL,X                                      ;Write Y position to sprite
table
B234      18              CLC                                             ;
B235      69 0E           ADC #HHEIGHT                                    ;Height
B237      9D 3A 1F        STA YEXTBL,X                                    ;Store in sprite table
B23A      98              TYA                                             ; number of Hulks into acc
B23B      29 0F           AND #MASK3
B23D      18              CLC
B23E      69 08           ADC #WSWAIT                                     ;Stagger time when Hulks move,
so that
                                                                          ;  they all don't move same
time
B240      9D 7D 1B        STA MTTBL,X                                     ;NUMBER OF FRAMES UNTIL MOVE
B243      20 14 D4        JSR RAND2_$D414                                 ;Random number between 0 and 2
to
                                                                          ;  determine what sprite
frame to start
B246      18              CLC                                             ;  with use
B247      69 01           ADC #$01                                        ;Add 1 to frame index
B249      9D D9 1C        STA SATBL,X                                     ; Animation frame
B24C      20 A8 D3        JSR RANDOM_$D3A8                                ;Random
B24F      29 07           AND #MASK3                                      ;
B251      9D 82 1C        STA DTTBL,X                                     ; Move 0-7 times in the
direction
                                                                          ;   selected before changing
direction
B254      29 03           AND #MASK2                                      ; Produces a "random"
direction
B256      9D D4 1B        STA DXTBL,X
B259      A9 05           LDA #HCODE                                      ; Hulk
B25B      9D 8C 1E        STA CRTBL,X                                     ;HULK OBJECT CODE
B25E      A9 00           LDA #HCODE
B260      9D 30 1D        STA CRTBL,X
B263      C6 A0           DEC TEMP0
B265      A5 A0           LDA TEMP0
```

```
B267      9D 2B 1C        STA SPRITE_DELTA_Y_$1C2B,X                    ;But its not setting a y
direction here
B26A      A9 01           LDA #$01                                      ;Make Hulk active
B26C      9D 91 1F        STA SPRITE_STATE_$1F91,X
B26F      A9 00           LDA #$00
B271      9D E8 1F        STA $1FE8,X
B274      E8              INX
B275      88              DEY
B276      10 A9           BPL WSHLOOP                                   ;MORE HULKS TO SET UP

;Integrated most of the disassembly from Dan Boris & "Scotty" ($B278-$B43C).
**********SPHEROIDS
WSHCONT:
B278      86 D8           STX SPTR                                      ;SET POINTER TO START OF
SPHEROIDS
B27A      A4 6E           LDY $6E
B27C      B9 8E EE        LDA $EE8E,Y
B27F      85 CF           STA $CF
B281      B9 48 EE        LDA $EE48,Y
B284      85 EB           STA $EB
B286      A9 01           LDA #$01
B288      85 A4           STA TEMP4
B28A      20 05 D4        JSR RANDPM_$D405
B28D      A8              TAY                                           ;LOOK UP IN SQBTTBL
B28E      A5 6D           LDA $6D
B290      F0 05           BEQ $B297
B292      C0 01           CPY #$01
B294      D0 01           BNE $B297
B296      88              DEY
B297      98              TYA
B298      18              CLC
B299      65 EB           ADC $EB
B29B      85 EB           STA $EB
B29D      A9 04           LDA #$04
B29F      85 EA           STA $EA
B2A1      0A              ASL A
B2A2      85 60           STA $60
B2A4      A9 00           LDA #$00
B2A6      85 7D           STA $7D

**********SPHEROIDS
;Setup Spheroids -  Spheroid AI Handler routine resides at $96F8.

B2A8      AC 0B 19        LDY SNUM                                      ;Get Spheroid count
B2AB      D0 03           BNE WSSGO                                     ;If we have some
B2AD      4C 21 B3        JMP WSSCONT_$B321

WSSGO:
B2B0      88              DEY

WSQLOOP:
B2B1      20 D8 D3        JSR RANDXYBX_$D3D8                            ;Get random screen coords
B2B4      A5 C4           LDA $C4
B2B6      9D 26 1B        STA SPRITE_Y,X
B2B9      18              CLC
B2BA      69 0D           ADC #$0D
B2BC      9D 3A 1F        STA SPRITE_Y_EXTENT,X                         ;Y Extent = YPos + #$0D
B2BF      8A              TXA
B2C0      29 01           AND #$01
B2C2      F0 04           BEQ $B2C8
B2C4      A9 02           LDA #$02
B2C6      D0 02           BNE $B2CA
B2C8      A9 94           LDA #$94
B2CA      9D CF 1A        STA SPRITE_X,X
B2CD      18              CLC
B2CE      69 09           ADC #$09                                      ;X Extent = XPos + 9
B2D0      9D E3 1E        STA SPRITE_X_EXTENT,X
B2D3      A9 06           LDA #$06                                      ;Spheroid
B2D5      9D 8C 1E        STA SPRITE_TYPE_$1E8C,X                       ;Save to sprite type
B2D8      20 A8 D3        JSR RANDOM_$D3A8
B2DB      29 03           AND #$03
B2DD      18              CLC
B2DE      69 03           ADC #$03
```

```
B2E0      9D 30 1D        STA MISCTBL_$1D30,X
B2E3      A9 00           LDA #SCODE
B2E5      85 B1           STA TEMP17                                  ;X Delta for
SET_OBJECT_DELTAXY
B2E7      85 B2           STA TEMP18                                  ;Y Delta
B2E9      A9 03           LDA #MAXSSPD
B2EB      85 A4           STA TEMP4
B2ED      20 05 D4        JSR RANDPM                                  ;X Delta "randomness" factor
for SET_OBJECT_DELTAXY
B2F0      85 B3           STA TEMP19
B2F2      20 05 D4        JSR RANDPM                                  ;Y Delta "randomness" factor
for SET_OBJECT_DELTAXY
B2F5      85 B4           STA TEMP20
B2F7      20 03 BA        JSR SET_OBJECT_DELTAXY_$BA03                ;Set X Delta and Y Delta of
object using passed parameters
B2FA      20 A8 D3        JSR RANDOM
B2FD      29 3F           AND #$3F
B2FF      C9 20           CMP #$20
B301      10 F7           BPL $B2FA
B303      9D 82 1C        STA MOVES_B4_DIR_CHANGE_$1C82,X
B306      A9 01           LDA #$01                                    ;Make active
B308      9D 91 1F        STA SPRITE_STATE_$1F91,X
B30B      A9 00           LDA #$00
B30D      9D E8 1F        STA $1FE8,X
B310      8A              TXA
B311      29 07           AND #MASK3
B313      9D 7D 1B        STA MTTBL,X
B316      A9 01           LDA #$01
B318      9D D9 1C        STA SATBL,X
B31B      E6 C9           INC CRELEFT                                 ;Increment number of enemies
on screen count
B31D      E8              INX
B31E      88              DEY
B31F      10 90           BPL WSSLOOP                                 ;MORE SPHEROIDS TO SET UP

*********QUARKS
;Setup Quarks - Quark AI Handler routine resides at $9824.

WSSCONT:
B321      86 D9           STX QPTR                                    ;Store X as index of first
Quark
B323      AC 0C 19        LDY QNUM                                    ;Read number of Quarks
B326      D0 03           BNE WSQGO
B328      4C 7E B3        JMP WSQCONT_$B37E

WSQGO:
B32B      88              DEY

ADDSQ:
B32C      20 D8 D3        JSR RANDXYBX_$D3D8                          ;GET A VALID POSITION
B32F      A5 C3           LDA RANDOMX
B331      9D CF 1A        STA XTBL,X                                  :XPOS
B334      18              CLC
B335      69 06           ADC #SQWID                                  ;COMPUTE EXTENT
B337      9D E3 1E        STA XEXTBL,X                                ;X EXTENT
B33A      8A              TXA
B33B      29 01           AND #MASK1
B33D      F0 04           BEQ $B343
B33F      A9 12           LDA #$12
B341      D0 02           BNE $B345
B343      A9 B2           LDA #$B2
B345      9D 26 1B        STA SPRITE_Y,X
B348      18              CLC
B349      69 09           ADC #SQHEIGHT                               ;COMPUTE EXTENT
B34B      9D 3A 1F        STA YEXTBL,X                                ;Y EXTENT
B34E      A9 07           LDA #QCODE                                  ;Quark
B350      9D 8C 1E        STA CRTBL,X                                 ;QUARK OBJECT CODE
B353      20 A8 D3        JSR RANDOM
B356      29 07           AND #MASK3
B358      9D D4 1B        STA SPRITE_DELTA_X_$1BD4,X
B35B      9D 2B 1C        STA SPRITE_DELTA_Y_$1C2B,X
B35E      8A              TXA
B35F      29 07           AND #$07
```

```
B361      9D 7D 1B        STA MTTBL,X
B364      A9 04           LDA #$04
B366      9D 82 1C        STA MOVES_B4_DIR_CHANGE_$1C82,X
B369      A9 01           LDA #$01
B36B      9D D9 1C        STA SATBL,X
B36E      A9 01           LDA #$01
B370      9D 91 1F        STA SPRITE_STATE_$1F91,X
B373      A9 00           LDA #$00
B375      9D E8 1F        STA $1FE8,X
B378      E6 C9           INC CRELEFT                                ;ANOTHER LIVING CREATURE IS
CREATED
B37A      E8              INX
B37B      88              DEY
B37C      10 AE           BPL WSQLOOP
B37E      A9 00           LDA #$00
B380      85 ED           STA $ED
B382      85 EE           STA $EE

*********TANKS
; Set up tanks - the in-game AI for tanks resides at $B4C7
B384      AC 0E 19        LDY TNUM                                   ;LOOP THRU ALL TANKS
B387      D0 03           BNE WSTGO                                  ;AT LEAST 1 TANK - DISTRIBUTE
TANKS
B389      4C DA B3        JMP WSTCONT                                ;GO TO GLOBAL TANK VARIABLE
SETUP

WSTGO:
B38C      88              DEY                                        ;Y RUNS UNTIL NEGATIVE

WSTLOOP:
B38D      20 D8 D3        JSR RANDXYBX
B390      A5 C3           LDA RANDOMX
B392      9D CF 1A        STA XTBL,X                                 ;TANK XPOS
B395      18              CLC
B396      69 09           ADC #TWID                                  ;COMPUTE EXTENT
B398      9D E3 1E        STA XEXTBL,X                               ;TANK X EXTENT
B39B      A5 C4           LDA RANDOMY
B39D      9D 26 1B        STA YTBL,X                                 ;TANK YPOS
B3A0      18              CLC
B3A1      69 0F           ADC #THEIGHT                               ;COMPUTE EXTENT
B3A3      9D 3A 1F        STA YEXTBL,X                               ;TANK Y EXTENT
B3A6      98              TYA
                          ;USE BRAIN # AS SEED TO GET GOOD DISTRIBUTION OF MOVE TIMERS
B3A7      29 07           AND #MASK3
B3A9      9D 7D 1B        STA MTTBL,X                                ;NUMBER OF FRAMES UNTIL MOVE
B3AC      09 02           ORA #$02
B3AE      9D D4 1B        STA DXTBL,X
B3B1      29 03           AND #MASK2
B3B3      09 04           ORA #$04
B3B5      9D D9 1C        STA SATBL,X
B3B8      9D 2B 1C        STA DYTBL,X
B3BB      20 A8 D3        JSR RANDOM
B3BE      29 0F           AND #MASK3
B3C0      69 0F           ADC #TSTIME                                ;BASE TIME BETWEEN TANK SHOTS
B3C2      9D 30 1D        STA MISCTBL,X
B3C5      A9 09           LDA #TCODE                                 ;Tank
B3C7      9D 8C 1E        STA CRTBL,X
B3CA      A9 01           LDA #$01
B3CC      9D 91 1F        STA SPRITE_STATE_$1F91,X
B3CF      A9 00           LDA #$00
B3D1      9D E8 1F        STA $1FE8,X
B3D4      E6 C9           INC CRELEFT                                ;ANOTHER LIVING CREATURE IS
CREATED
B3D6      E8              INX                                        ;INCREMENT RUNNING POINTER
B3D7      88              DEY
B3D8      10 B3           BPL WSTLOOP                                ;MORE TANKS TO SET UP

*********BRAINS
; Set up Brains - the in-game AI for Brain resides at $9A9A
WSTCONT:
B3DA      86 DC           STX BPTR                                   ;POINTER TO START OF BRAINS
B3DC      A5 D6           LDA FPTR
B3DE      85 A0           STA TEMP0
```

```
B3E0      A9 01          LDA #$01
B3E2      85 CB          STA $CB
B3E4      A4 6E          LDY $6E
B3E6      B9 D4 EE       LDA $EED4,Y
B3E9      85 D2          STA $D2
B3EB      AC 0F 19       LDY BNUM                                  ;Get number of Brains
B3EE      D0 03          BNE WSBGO
B3F0      4C 63 B4       JMP $B463

WSBGO:
B3F3      88             DEY

WSBLOOP:
B3F4      20 D8 D3       JSR RANDXYBX                              ;Get a random X and Y
coordinate
B3F7      A5 C3          LDA RANDOMX
B3F9      9D CF 1A       STA XTBL,X                                ;BRAIN XPOS
B3FC      18             CLC
B3FD      69 09          ADC #BWID                                 ;COMPUTE EXTENT
B3FF      9D E3 1E       STA XEXTBL,X                              ;BRAIN X EXTENT - 9 pixels wide
B402      A5 C4          LDA RANDOMY
B404      9D 26 1B       STA YTBL,X                                ;BRAIN YPOS
B407      18             CLC
B408      69 0C          ADC #BHEIGHT                              ;COMPUTE EXTENT - 12 pixels
high
B40A      9D 3A 1F       STA YEXTBL,X                              ;BRAIN Y EXTENT
B40D      98             TYA
                                                                   ;USE BRAIN # AS SEED TO GET GOOD
                                                                   ; DISTRIBUTION OF MOVE TIMERS
B40E      29 0F          AND #MASK3
B410      18             CLC
B411      69 08          ADC #$08
B413      9D 7D 1B       STA MTTBL,X                               ;NUMBER OF FRAMES UNTIL MOVE
B416      20 14 D4       JSR RAND2_$D414                           ;Get a number between 0 and 2
B419      9D D9 1C       STA SATBL,X                               ;BRAIN ANIMATION STEP
B41C      20 A8 D3       JSR RANDOM_$D3A8
B41F      29 07          AND #MASK3
B421      9D D4 1B       STA DTTBL,X                               ;Pick a random direction
B424      A9 00          LDA #$00
B426      9D 82 1C       STA DXTBL,X                               ;Moves immediately
B429      A5 A0          LDA TEMP0
B42B      F0 10          BEQ WSB02
B42D      C5 D7          CMP HPTR
B42F      90 0C          BCC WSB02
B431      A5 D6          LDA FPTR
B433      85 A0          STA TEMP0
B435      C5 D7          CMP $D7
B437      D0 04          BNE $B43D
B439      A9 00          LDA #$00
B43B      85 A0          STA TEMP0

;Disassembly of $B43D-$BD6D compliments of Dan Boris & "Scotty"
WSB02:    ;NOW PUT TEMP0 INTO DYTBL - BRAIN TARGET
B43D      9D 2B 1C       STA DYTBL,X                               ;This is not for the Brain's
delta Y,
                                                                   ; because an 8-dir object like
the Brain
                                                                   ; packs its direction into
sprite delta X
                                                                   ; ...intriguing
B440      A5 A0          LDA TEMP0
B442      F0 02          BEQ $B446
B444      E6 A0          INC TEMP0

B446      20 A8 D3       JSR RANDOM_$D3A8
B449      29 3F          AND #MASK6
B44B      9D 30 1D       STA MISCTBL_$1D30,X
B44E      A9 0A          LDA #BCODE                                ;Brain type
B450      9D 8C 1E       STA CRTBL,X                               ;BRAIN OBJECT CODE
B453      A9 01          LDA #$01                                  ;Make active
B455      9D 91 1F       STA SPRITE_STATE_$1F91,X
B458      A9 00          LDA #$00
B45A      9D E8 1F       STA $1FE8,X
```

```
*          DLPHTBL,DLPLTBL AND DL2PTBL WILL BE SET UP BY THE LOAD AT THE END OF
*                  THE WAVESTRT ROUTINE
*          DONE WITH THIS BRAIN, ON TO NEXT...

B45D      E6 C9          INC CRELEFT                               ;ANOTHER LIVING CREATURE IS
CREATED
B45F      E8             INX                                       ;INCREMENT RUNNING POINTER
B460      88             DEY                                       ;Decrement Brain counter
B461      10 91          BPL WSBLOOP_$B3F4                         ;NEED MORE BRAINS

B463      A9 80          LDA #$80
B465      85 EF          STA $EF
B467      E0 53          CPX #$53                                  ;Hit max entities?
B469      B0 0A          BCS $B475                                 ;Yes
B46B      A5 EF          LDA $EF
B46D      86 EF          STX $EF
B46F      9D 30 1D       STA MISCTBL_$1D30,X
B472      E8             INX

B473      D0 F2          BNE WSBLOOP_$B3F4

; Now we're setting up the player
B475      A9 00          LDA #$00
B477      A2 03          LDX $03
B479      9D 27 1C       STA SHOT_DIR_TBL_$1C27,X
B47C      9D 2C 1D       STA $1D2C,X
B47F      9D E4 1F       STA $1FE4,X
B482      CA             DEX
B483      10 F4          BPL $B479
B485      8D 30 1D       STA MISCTBL_$1D30
B488      8D 7D 1B       STA MTTBL,X
B48B      A9 0F          LDA #$0F
B48D      8D DF 1E       STA $1EDF
B490      8D E0 1E       STA $1EE0
B493      8D E1 1E       STA $1EE1
B496      8D E2 1E       STA $1EE2

B499      A9 4B          LDA #$4B                                  ;
B49B      8D CF 1A       STA SPRITE_X                              ;Set player X-pos Left
B49E      A9 50          LDA #$50                                  ;
B4A0      8D E3 1E       STA SPRITE_X_EXTENT                       ;Set player X-pos Right
B4A3      A9 62          LDA #$62                                  ;
B4A5      8D 26 1B       STA SPRITE_Y                              ;Set player Y-pos Top
B4A8      A9 6D          LDA #$6D                                  ;
B4AA      8D 3A 1F       STA SPRITE_Y_EXTENT                       ;Set player Y-pos Bottom
B4AD      A9 00          LDA #$00                                  ;
B4AF      8D D9 1C       STA SATBL,X                               ;Set player animation frame
B4B2      A9 0D          LDA #$0D                                  ;
B4B4      8D D4 1B       STA SPRITE_DELTA_X_$1BD4                  ;Set direction player is
facing
B4B7      A9 01          LDA #$01
B4B9      85 C8          STA $C8
B4BB      A9 00          LDA #$00
B4BD      8D 91 1F       STA SPRITE_STATE_$1F91                    ;Set player as disabled - we
don't want
                                                                   ; him moving just yet
B4C0      A9 00          LDA #$00
B4C2      85 CA          STA $CA
B4C4      4C 28 91       JMP $9128
;
; TANK AI HANDLER
;
B4C7      A5 E2          LDA $E2                                   ;Is screen being drawn?
B4C9      D0 0A          BNE $B4D5                                 ;Yeah, so just process next
item
B4CB      BD 91 1F       LDA SPRITE_STATE_$1F91,X
B4CE      D0 08          BNE $B4D8
B4D0      A9 00          LDA #$00                                  ;Kill this tank permanently
(which D71E
                                                                   ; does proper, this looks a
c&p)
B4D2      9D 8C 1E       STA SPRITE_TYPE_$1E8C,X
```

```
B4D5      4C FC 91        JMP OBJCONT_$91FC                            ;PROCESS NEXT OBJECT
B4D8      C9 03           CMP #$03                                     ;Is this tank in the process of
dying
B4DA      D0 2C           BNE $B508                                    ;No
B4DC      BD D9 1C        LDA SATBL,X
B4DF      C9 08           CMP #$08
B4E1      B0 11           BCS $B4F4
B4E3      A9 08           LDA #$08
B4E5      9D D9 1C        STA SATBL,X
B4E8      A9 02           LDA #$02                                     ;Generic explosion sound
B4EA      20 95 E3        JSR DOTUNE_$E395
B4ED      A9 03           LDA #$03
B4EF      9D 7D 1B        STA MTTBL,X
B4F2      D0 37           BNE $B52B
B4F4      C9 0B           CMP #$0B
B4F6      90 06           BCC $B4FE
B4F8      20 1C D7        JSR $D71C                                    ;Permanently remove this tank
B4FB      4C FC 91        JMP $OBJCONT_$91FC                           ;PROCESS NEXT OBJECT

B4FE      A9 01           LDA #$01
B500      9D 7D 1B        STA MTTBL,X
B503      FE D9 1C        INC SATBL,X
B506      D0 23           BNE $B52B

B508      A9 03           LDA #$03
B50A      9D 7D 1B        STA MTTBL,X
B50D      BD D9 1C        LDA SATBL,X
B510      C9 04           CMP #$04
B512      B0 1D           BCS $B531
B514      FE D9 1C        INC SATBL,X
B517      BD 26 1B        LDA SPRITE_Y,X
B51A      85 BF           STA YINTEND_BF
B51C      BD CF 1A        LDA SPRITE_X,X
B51F      85 BE           STA XINTEND_BE
B521      BD 3A 1F        LDA SPRITE_Y_EXTENT,X
B524      85 C1           STA YYINTEND_C1
B526      BD E3 1E        LDA SPRITE_X_EXTENT,X
B529      85 C0           STA XXINTEND_C0

B52B      20 AF E1        JSR $E1AF                                    ;Draw tank
B52E      4C FC 91        JMP OBJCONT_$91FC                            ;PROCESS NEXT OBJECT
B531      A9 02           LDA #$02
B533      3D D4 1B        AND SPRITE_DELTA_X_$1BD4,X
B536      D0 11           BNE $B549
B538      FE D9 1C        INC SATBL,X
B53B      BD D9 1C        LDA SATBL,X
B53E      C9 08           CMP #$08
B540      90 16           BCC $B558
B542      A9 04           LDA #$04
B544      9D D9 1C        STA SATBL,X
B547      D0 0F           BNE $B558
B549      DE D9 1C        DEC SATBL,X
B54C      BD D9 1C        LDA SATBL,X
B54F      C9 04           CMP #$04
B551      B0 05           BCS $B558
B553      A9 07           LDA #$07
B555      9D D9 1C        STA SATBL,X
B558      BC D4 1B        LDY SPRITE_DELTA_X_$1BD4,X
B55B      18              CLC
B55C      BD CF 1A        LDA SPRITE_X,X
B55F      79 1D EC        ADC XDIRTBL_$EC1D,Y
B562      85 BE           STA XINTEND_BE
B564      18              CLC
B565      BD E3 1E        LDA SPRITE_X_EXTENT,X
B568      79 1D EC        ADC XDIRTBL_$EC1D,Y
B56B      85 C0           STA XXINTEND_C0
B56D      18              CLC
B56E      BD 26 1B        LDA SPRITE_Y,X
B571      79 25 EC        ADC YDIRTBL_$EC25,Y
B574      85 BF           STA YINTEND_BF
B576      18              CLC
B577      BD 3A 1F        LDA SPRITE_Y_EXTENT,X
B57A      79 25 EC        ADC YDIRTBL_$EC25,Y
```

```
B57D      85 C1          STA YYINTEND_C1
B57F      20 ED D1       JSR $D1ED
B582      A5 A4          LDA TEMP4
B584      F0 05          BEQ $B58B
B586      A9 00          LDA #$00
B588      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
B58B      DE 2B 1C       DEC SPRITE_DELTA_Y_$1C2B,X
B58E      10 14          BPL $B5A4
B590      20 A8 D3       JSR RANDOM_$D3A8
B593      29 07          AND #$07
B595      C9 02          CMP #$02
B597      90 F7          BCC $B590
B599      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
B59C      20 A8 D3       JSR RANDOM_$D3A8
B59F      29 1F          AND #$1F
B5A1      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
B5A4      A5 CB          LDA $CB
B5A6      D0 13          BNE $B5BB
B5A8      DE 30 1D       DEC MISCTBL_$1D30,X
B5AB      10 0E          BPL $B5BB
B5AD      A5 7D          LDA $7D
B5AF      C9 0B          CMP #$0B
B5B1      B0 08          BCS $B5BB
B5B3      20 A2 BB       JSR $BBA2                              ;Fire a tank shell
B5B6      A9 0F          LDA #$0F
B5B8      9D 30 1D       STA MISCTBL_$1D30,X
B5BB      4C F8 97       JMP $97F8
;
; Fire a spark for the Enforcer (subroutine)
;
B5BE      86 A6          STX TEMP6                              ;Save current object index in
temp
B5C0      20 34 92       JSR GET_MISSILE_SLOT_$9234
B5C3      30 7F          BMI $B644
B5C5      A4 A6          LDY TEMP6                              ;Put object index into Y. Now
x = index of
                                                                ; free missile slot, y = index
of Enforcer
B5C7      B9 CF 1A       LDA SPRITE_X,Y                         ;Copy Enforcer X to spark X
B5CA      9D CF 1A       STA SPRITE_X,X
B5CD      18             CLC
B5CE      69 05          ADC #$05
B5D0      9D E3 1E       STA SPRITE_X_EXTENT,X                  ;Spark X is 5 wide
B5D3      B9 26 1B       LDA SPRITE_Y,Y                         ;Copy Enforcer Y to spark Y
B5D6      9D 26 1B       STA SPRITE_Y,X
B5D9      18             CLC
B5DA      69 07          ADC #$07
B5DC      9D 3A 1F       STA SPRITE_Y_EXTENT,X                  ;Spark is 7 height
B5DF      A9 01          LDA #$01
B5E1      9D 91 1F       STA SPRITE_STATE_$1F91,X               ;Mark spark active
B5E4      A9 0C          LDA #$0C
B5E6      9D 8C 1E       STA SPRITE_TYPE_$1E8C,X                ;Set sprite type to spark
B5E9      20 36 E1       JSR $E136
B5EC      AD 3E 21       LDA $213E
B5EF      F0 07          BEQ $B5F8
B5F1      8A             TXA
B5F2      A8             TAY
B5F3      A6 A6          LDX TEMP6
B5F5      4C FB BA       JMP RECORD_OPEN_SLOT_$BAFB
B5F8      A9 00          LDA #$00
B5FA      85 B7          STA FRMCNT
B5FC      AD CF 1A       LDA SPRITE_X
B5FF      85 B8          STA TEMPX
B601      AD 26 1B       LDA SPRITE_Y
B604      85 B9          STA TEMPY
B606      20 DC BC       JSR COMPUTE_DELTAS_$BCDC
B609      A9 01          LDA #$01                               ;We want a number between -1
and 1
B60B      85 A4          STA TEMP4
B60D      20 05 D4       JSR RANDPM_$D405
B610      85 B3          STA TEMP19
B612      20 05 D4       JSR RANDPM_$D405
B615      85 B4          STA TEMP20
```

```
B617     20 03 BA       JSR SET_OBJECT_DELTAXY_$BA03                ;$b1 and $b2 set by
COMPUTE_DELTAS_BCDC above
B61A     20 A8 D3       JSR RANDOM_$D3A8                            ;Now get any random number
B61D     29 07          AND #$07                                    ;Make it 0-7
B61F     18             CLC
B620     69 01          ADC #$01                                    ;Adjust by 1 to make it 1-8
B622     85 A4          STA TEMP4                                   ;Save original value in temp
B624     0A             ASL A                                       ; * 16
B625     0A             ASL A
B626     0A             ASL A
B627     0A             ASL A
B628     05 A4          ORA TEMP4                                   ;Add in original value
B62A     9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X
B62D     A9 20          LDA #$20
B62F     9D 30 1D       STA SPARK_LIFE,X
B632     A5 AA          LDA TEMP10                                  ;Set by COMPUTE_DELTAS_BCDC
B634     9D 7D 1B       STA MTTBL,X
B637     0A             ASL A
B638     0A             ASL A
B639     0A             ASL A                                       ;Divide by 8
B63A     9D D9 1C       STA SATBL,X
B63D     E6 EE          INC $EE                                     ;Increment sparks on screen
count
B63F     A9 0C          LDA #$0C                                    ;Play Spark Fired Sound
B641     20 95 E3       JSR DOTUNE_$E395
B644     A6 A6          LDX TEMP6                                   ;Restore object x
B646     60             RTS
;
; SPARK (ENFORCER SHOT) AI
;
B647     BD 91 1F       LDA SPRITE_STATE_$1F91,X                    ;What's the state of our
spark?
B64A     C9 03          CMP #$03
B64C     D0 03          BNE $B651
B64E     4C 18 B7       JMP $B718
B651     DE 30 1D       DEC SPARK_LIFE,X                            ;Count down its life
B654     D0 03          BNE $B659                                   ;If <>0 we're still alive
B656     4C 18 B7       JMP $B718
B659     BD D9 1C       LDA SATBL,X                                 ;Do some jiggery pokery with
the current anim frame
B65C     A8             TAY
B65D     29 03          AND #$03
B65F     38             SEC
B660     E9 01          SBC #$01
B662     10 02          BPL $B666
B664     A9 03          LDA #$03
B666     85 B7          STA FRMCNT
B668     98             TYA
B669     29 F8          AND #$F8
B66B     05 B7          ORA FRMCNT
B66D     9D D9 1C       STA SATBL,X                                 ;End animation frame
shenanigans
B670     20 28 BA       JSR ALTER_DELTAS_$BA28                      ;Alter the sparks deltas, if
required,
                                                                    ; to change its angle of
movement.
B673     18             CLC
B674     BD CF 1A       LDA SPRITE_X,X
B677     65 B1          ADC TEMP17
B679     85 BE          STA XINTEND_BE
B67B     18             CLC
B67C     BD E3 1E       LDA SPRITE_X_EXTENT,X
B67F     65 B1          ADC TEMP17
B681     85 C0          STA XXINTEND_C0
B683     C9 9C          CMP #$9C
B685     B0 09          BCS $B690
B687     A5 BE          LDA XINTEND_BE
B689     18             CLC
B68A     69 10          ADC #$10
B68C     C9 12          CMP #$12
B68E     B0 0A          BCS $B69A
B690     BD CF 1A       LDA SPRITE_X,X
B693     85 BE          STA XINTEND_BE
```

```
B695     BD E3 1E       LDA SPRITE_X_EXTENT,X
B698     85 C0          STA XXINTEND_C0
B69A     18             CLC
B69B     BD 26 1B       LDA SPRITE_Y,X
B69E     65 B2          ADC TEMP18
B6A0     85 BF          STA YINTEND_BF
B6A2     18             CLC
B6A3     BD 3A 1F       LDA SPRITE_Y_EXTENT,X
B6A6     65 B2          ADC TEMP18
B6A8     85 C1          STA YYINTEND_C1
B6AA     C9 BC          CMP #$BC
B6AC     B0 06          BCS $B6B4
B6AE     A5 BF          LDA YINTEND_BF
B6B0     C9 12          CMP #$12
B6B2     B0 0A          BCS $B6BE
B6B4     BD 26 1B       LDA SPRITE_Y,X
B6B7     85 BF          STA YINTEND_BF
B6B9     BD 3A 1F       LDA SPRITE_Y_EXTENT,X
B6BC     85 C1          STA YYINTEND_C1
B6BE     BD 82 1C       LDA MOVES_B4_DIR_CHANGE_$1C82,X
B6C1     85 B7          STA FRMCNT
B6C3     29 0F          AND #$0F
B6C5     38             SEC
B6C6     E9 01          SBC #$01
B6C8     D0 3B          BNE $B705
B6CA     A5 B1          LDA TEMP17
B6CC     18             CLC
B6CD     65 B3          ADC TEMP19
B6CF     85 B1          STA TEMP17
B6D1     30 08          BMI $B6DB
B6D3     A9 08          LDA #$08
B6D5     C5 B1          CMP TEMP17
B6D7     90 08          BCC $B6E1
B6D9     B0 08          BCS $B6E3
B6DB     A9 F8          LDA #$F8
B6DD     C5 B1          CMP TEMP17
B6DF     90 02          BCC $B6E3
B6E1     85 B1          STA TEMP17
B6E3     A5 B2          LDA TEMP18
B6E5     18             CLC
B6E6     65 B4          ADC TEMP20
B6E8     85 B2          STA TEMP18
B6EA     30 08          BMI $B6F4
B6EC     A9 08          LDA #$08
B6EE     C5 B2          CMP TEMP18
B6F0     90 08          BCC $B6FA
B6F2     B0 08          BCS $B6FC
B6F4     A9 F8          LDA #$F8
B6F6     C5 B2          CMP TEMP18
B6F8     90 02          BCC $B6FC
B6FA     85 B2          STA TEMP18
B6FC     20 03 BA       JSR SET_OBJECT_DELTAXY_$BA03
B6FF     A5 B7          LDA FRMCNT
B701     4A             LSR A
B702     4A             LSR A
B703     4A             LSR A
B704     4A             LSR A
B705     85 A1          STA TEMP1
B707     A5 B7          LDA FRMCNT
B709     29 F0          AND #$F0
B70B     05 A1          ORA TEMP1
B70D     9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X
B710     20 AF E1       JSR $E1AF                                  ;Draw spark
B713     AD 3E 21       LDA $213E
B716     F0 08          BEQ $B720
B718     C6 EE          DEC $EE
B71A     20 1E D7       JSR $D71E                                  ;Kill this sprite
B71D     4C FC 91       JMP OBJCONT_$91FC                          ;PROCESS NEXT OBJECT
B720     BD D9 1C       LDA SATBL,X
B723     4A             LSR A
B724     4A             LSR A
B725     4A             LSR A
B726     9D 7D 1B       STA MTTBL,X
```

```
B729      A5 BE          LDA XINTEND_BE
B72B      9D CF 1A       STA SPRITE_X,X
B72E      A5 C0          LDA XXINTEND_C0
B730      9D E3 1E       STA SPRITE_X_EXTENT,X
B733      A5 BF          LDA YINTEND_BF
B735      9D 26 1B       STA SPRITE_Y,X
B738      A5 C1          LDA YYINTEND_C1
B73A      9D 3A 1F       STA SPRITE_Y_EXTENT,X
B73D      4C FC 91       JMP OBJCONT_$91FC                          ;PROCESS NEXT OBJECT
;
; PROG AI HANDLER
;
B740      A9 04          LDA #$04
B742      9D 7D 1B       STA MTTBL,X
B745      BD 91 1F       LDA SPRITE_STATE_$1F91,X                   ;Read sprite state of Prog
B748      29 02          AND #$02                                   ;Dying?
B74A      F0 2D          BEQ $B779                                  ;No

PROGDYING:
B74C      BD D9 1C       LDA SATBL,X
B74F      D0 0F          BNE $B760
B751      BC 30 1D       LDY MISCTBL_$1D30,X                        ;Unsure what's happening here
B754      F0 05          BEQ $B75B
B756      A9 01          LDA #$01
B758      99 7D 1B       STA MTTBL,X
B75B      A9 02          LDA #$02                                   ;Generic explosion sound
B75D      20 95 E3       JSR DOTUNE_$E395                           ;Play sound
B760      FE D9 1C       INC SATBL,X
B763      A9 01          LDA #$01
B765      9D 7D 1B       STA MTTBL,X
B768      A9 04          LDA #$04
B76A      85 A1          STA TEMP1
B76C      BD D9 1C       LDA SATBL,X
B76F      C9 06          CMP #$06
B771      90 3F          BCC $B7B2
B773      20 1E D7       JSR $D71E                                  ;Kill this sprite
B776      4C FC 91       JMP OBJCONT_$91FC                          ;PROCESS NEXT OBJECT

; if we get here, the Prog is still alive
B779      DE 82 1C       DEC MOVES_B4_DIR_CHANGE_$1C82,X            ;Decrement move count
B77C      F0 08          BEQ $B786                                        ;If zero, then we can
move
B77E      BD D4 1B       LDA SPRITE_DELTA_X_$1BD4,X
B781      85 A1          STA TEMP1
B783      4C B2 B7       JMP $B7B2
B786      20 A8 D3       JSR RANDOM_$D3A8                           ;Get a random number
B789      29 07          AND #$07                                   ;Mask off lower 3 bits
B78B      F0 F9          BEQ $B786                                  ;If its 0, get random number
again
B78D      9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X                         ;Save random
number to moves before
                                                                          ; direction change.
B790      20 A8 D3       JSR RANDOM_$D3A8

B793      29 0F          AND #$0F
B795      D0 0A          BNE $B7A1
B797      20 A8 D3       JSR RANDOM_$D3A8
B79A      29 03          AND #$03
B79C      85 A1          STA TEMP1
B79E      4C B2 B7       JMP $B7B2
B7A1      A9 01          LDA #$01
B7A3      85 B7          STA FRMCNT
B7A5      A0 00          LDY $00
B7A7      20 6E BD       JSR PICK_DIRECTION_$BD6E
B7AA      C9 0F          CMP #$0F
B7AC      D0 02          BNE $B7B0
B7AE      A9 03          LDA #$03
B7B0      85 A1          STA TEMP1
B7B2      BD D4 1B       LDA SPRITE_DELTA_X_$1BD4,X
B7B5      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
B7B8      85 A2          STA TEMP2
B7BA      A5 A1          LDA TEMP1
B7BC      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
```

```
B7BF      A5 A2          LDA TEMP2
B7C1      0A             ASL A
B7C2      0A             ASL A
B7C3      0A             ASL A
B7C4      0A             ASL A
B7C5      18             CLC
B7C6      65 A1          ADC TEMP1
B7C8      38             SEC
B7C9      A4 A2          LDY TEMP2
B7CB      F9 8F EC       SBC $EC8F,Y
B7CE      A8             TAY
B7CF      B9 5D EC       LDA $EC5D,Y
B7D2      85 B8          STA TEMPX
B7D4      B9 76 EC       LDA $EC76,Y
B7D7      85 B9          STA TEMPY
B7D9      BD D4 1B       LDA SPRITE_DELTA_X_$1BD4,X
B7DC      20 35 D1       JSR $D135                                     ;Outputs width & height of
frame in
                                                                       ; $b8 and $ab.
B7DF      BD CF 1A       LDA SPRITE_X,X
B7E2      18             CLC
B7E3      65 B8          ADC TEMPX
B7E5      85 BE          DTA XINTEND_BE
B7E7      18             CLC
B7E8      65 AB          ADC TEMP11
B7EA      85 C0          STA XXINTEND_C0
B7EC      BD 26 1B       LDA SPRITE_Y,X
B7EF      18             CLC
B7F0      65 B9          ADC TEMPY
B7F2      85 BF          STA YINTEND_BF
B7F4      18             CLC
B7F5      65 AC          ADC TEMP12
B7F7      85 C1          STA YYINTEND_C1
B7F9      20 ED D1       JSR $D1ED
B7FC      A5 A4          LDA TEMP4
B7FE      F0 1C          BEQ $B81C
B800      BD 30 1D       LDA MISCTBL_$1D30,X
B803      F0 92          BEQ $B797
B805      20 53 D1       JSR $D153
B808      BD CF 1A       LDA SPRITE_X,X
B80B      85 BE          STA XINTEND_BE
B80D      BD 26 1B       LDA SPRITE_Y,X
B810      85 BF          STA YINTEND_BF
B812      BD E3 1E       LDA SPRITE_X_EXTENT,X
B815      85 C0          STA XXINTEND_C0
B817      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
B81A      85 C1          STA YYINTEND_C1
B81C      20 AF E1       JSR $E1AF
B81F      AD 3E 21       LDA $213E
B822      F0 0D          BEQ $B831
B824      20 A8 D3       JSR RANDOM_$D3A8
B827      29 01          AND #$01
B829      18             CLC
B82A      69 02          ADC #$02
B82C      85 A1          STA TEMP1
B82E      4C BA B7       JMP $B7BA
B831      A5 BE          LDA XINTEND_BE
B833      9D CF 1A       STA SPRITE_X,X
B836      A5 C0          LDA XXINTEND_C0
B838      9D E3 1E       STA SPRITE_X_EXTENT,X
B83B      A5 BF          LDA YINTEND_BF
B83D      9D 26 1B       STA SPRITE_Y,X
B840      A5 C1          LDA YYINTEND_C1
B842      9D 3A 1F       STA SPRITE_Y_EXTENT,X
B845      A9 00          LDA #$00
B847      9D 30 1D       STA MISCTBL_$1D30,X
B84A      4C FC 91       JMP OBJCONT_$91FC                             ;PROCESS NEXT OBJECT
;
; Brain shot (cruise missile)
;
B84D      A9 03          LDA #$03
B84F      9D 7D 1B       STA MTTBL,X
B852      BD 91 1F       LDA SPRITE_STATE_$1F91,X
```

```
B855      29 02          AND #$02
B857      F0 0B          BEQ $B864
B859      A9 02          LDA #$02                                  ;Play explosion noise
B85B      20 95 E3       JSR DOTUNE_$E395
B85E      20 1E D7       JSR $D71E                                 ;Kill this sprite
B861      4C FC 91       JMP OBJCONT_$91FC                         ;PROCESS NEXT OBJECT
B864      BD D4 1B       LDA SPRITE_DELTA_X_$1BD4,X
B867      85 AA          STA TEMP10
B869      BD 2B 1C       LDA SPRITE_DELTA_Y_$1C2B,X
B86C      85 AB          STA TEMP11
B86E      DE 82 1C       DEC MOVES_B4_DIR_CHANGE_$1C82,X
B871      F0 08          BEQ $B87B                                 ;Time for a direction change
B873      BD D4 1B       LDA SPRITE_DELTA_X_$1BD4,X
B876      85 A1          STA TEMP1                                 ;Save direction in $A1
B878      4C A8 B8       JMP $B8A8
B87B      20 A8 D3       JSR RANDOM_$D3A8                          ;Get a random number
B87E      29 03          AND #$03
B880      18             CLC
B881      69 02          ADC #$02
B883      9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X                 ; OK, wait 2-5 frames
before next dir change
B886      20 A8 D3       JSR RANDOM_$D3A8
B889      29 03          AND #$03
B88B      D0 0A          BNE $B897
B88D      20 A8 D3       JSR RANDOM_$D3A8                          ;Get another random number
B890      29 07          AND #$07                                  ;Make sure its no bigger than
7
B892      85 A1          STA TEMP1                                 ;This will change the X delta
temp var
B894      4C A8 B8       JMP $B8A8


B897      A9 00          LDA #$00
B899      85 B7          STA FRMCNT
B89B      A0 00          LDY $00
B89D      20 6E BD       JSR PICK_DIRECTION_$BD6E
B8A0      C9 0F          CMP #$0F
B8A2      D0 02          BNE $B8A6
B8A4      A9 03          LDA #$03
B8A6      85 A1          STA TEMP1

B8A8      BD D4 1B       LDA SPRITE_DELTA_X_$1BD4,X                ;Swap X deltas and Y deltas
B8AB      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
B8AE      85 A2          STA TEMP2                                 ;$a2 = X delta
B8B0      A5 A1          LDA TEMP1                                 ; $
B8B2      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X                ;Restore
B8B5      A4 A2          LDY TEMP2
B8B7      D9 94 EC       CMP $EC94,Y
B8BA      D0 0D          BNE $B8C9
B8BC      A5 AA          LDA TEMP10
B8BE      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
B8C1      A5 AB          LDA TEMP11
B8C3      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
B8C6      4C 8D B8       JMP $B88D
B8C9      20 E9 B9       JSR $B9E9
B8CC      A4 A2          LDY TEMP2                                 ;Get direction cruise missile
is heading in
B8CE      18             CLC
B8CF      BD CF 1A       LDA SPRITE_X,X
B8D2      79 9C EC       ADC CRUISEXDIRTBL_$EC9C,Y                 ;Add on the X component to X
B8D5      85 BE          STA XINTEND_BE
B8D7      18             CLC
B8D8      BD E3 1E       LDA SPRITE_X_EXTENT,X
B8DB      79 9C EC       ADC CRUISEXDIRTBL_$EC9C,Y                 ;And adjust the X extent
B8DE      85 C0          STA XXINTEND_C0
B8E0      18             CLC
B8E1      BD 26 1B       LDA SPRITE_Y,X
B8E4      79 A4 EC       ADC CRUISEYDIRTBL_$ECA4,Y                 ;Add on the Y component to Y
B8E7      85 BF          STA YINTEND_BF
B8E9      18             CLC
B8EA      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
B8ED      79 A4 EC       ADC CRUISEYDIRTBL_$ECA4,Y                 ;And adjust the Y extent
B8F0      85 C1          STA YYINTEND_C1
```

```
B8F2      A4 A1          LDY TEMP1
B8F4      A5 BE          LDA XINTEND_BE
B8F6      18             CLC
B8F7      69 10          ADC #$10
B8F9      C9 15          CMP #$15
B8FB      B0 0E          BCS $B90B
B8FD      C0 03          CPY #$03
B8FF      F0 42          BEQ $B943
B901      C0 06          CPY #$06
B903      F0 3E          BEQ $B943
B905      C0 07          CPY #$07
B907      F0 3A          BEQ $B943
B909      D0 12          BNE $B91D
B90B      A5 C0          LDA XXINTEND_C0
B90D      C9 9A          CMP #$9A
B90F      90 0C          BCC $B91D
B911      C0 02          CPY #$02
B913      F0 2E          BEQ $B943
B915      C0 04          CPY #$04
B917      F0 2A          BEQ $B943
B919      C0 05          CPY #$05
B91B      F0 26          BEQ $B943
B91D      A5 BF          LDA YINTEND_BF
B91F      C9 17          CMP #$17
B921      B0 0E          BCS $B931
B923      C0 00          CPY #$00
B925      F0 1C          BEQ $B943
B927      C0 04          CPY #$04
B929      F0 18          BEQ $B943
B92B      C0 07          CPY #$07
B92D      F0 14          BEQ $B943
B92F      D0 1F          BNE $B950
B931      A5 C1          LDA YYINTEND_C1
B933      C9 B8          CMP #$B8
B935      90 19          BCC $B950
B937      C0 01          CPY #$01
B939      F0 08          BEQ $B943
B93B      C0 05          CPY #$05
B93D      F0 04          BEQ $B943
B93F      C0 06          CPY #$06
B941      D0 0D          BNE $B950
B943      A5 AA          LDA TEMP10
B945      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
B948      A5 AB          LDA TEMP11
B94A      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
B94D      4C 8D B8       JMP $B88D
B950      20 AF E1       JSR $E1AF
B953      AD 3E 21       LDA $213E
B956      F0 12          BEQ $B96A
B958      A9 01          LDA #$01
B95A      9D 7D 1B       STA MTTBL,X
B95D      A5 AA          LDA TEMP10
B95F      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
B962      A5 AB          LDA TEMP11
B964      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
B967      4C FC 91       JMP OBJCONT_$91FC                    ;PROCESS NEXT OBJECT
B96A      A5 BE          LDA XINTEND_BE
B96C      9D CF 1A       STA SPRITE_X,X
B96F      A5 C0          LDA XXINTEND_C0
B971      9D E3 1E       STA SPRITE_X_EXTENT,X
B974      A5 BF          LDA YINTEND_BF
B976      9D 26 1B       STA SPRITE_Y,X
B979      A5 C1          LDA YYINTEND_C1
B97B      9D 3A 1F       STA SPRITE_Y_EXTENT,X
B97E      4C FC 91       JMP OBJCONT_$91FC                    ;PROCESS NEXT OBJECT
;
; Fire a cruise missile from a Brain
; Called by Brain routine
; x = Brain index
;
B981      86 A6          STX TEMP6                            ;Save X as next sub call will
change it
B983      20 34 92       JSR GET_MISSILE_SLOT_$9234           ;Get an index for cruise
```

```
missile into X
B986      30 5E          BMI $B9E6
B988      A4 A6          LDY TEMP6                                ;Y = Brain index
B98A      B9 CF 1A       LDA SPRITE_X,Y
B98D      9D CF 1A       STA SPRITE_X,X                           ; obj2.x = obj1.x
B990      18             CLC
B991      69 07          ADC #$07
B993      9D E3 1E       STA SPRITE_X_EXTENT,X
B996      B9 26 1B       LDA SPRITE_Y,Y                           ; obj2.y = obj1.y
B999      9D 26 1B       STA SPRITE_Y,X
B99C      18             CLC
B99D      69 0B          ADC #$0B
B99F      9D 3A 1F       STA SPRITE_Y_EXTENT,X
B9A2      A9 01          LDA #$01
B9A4      9D 91 1F       STA SPRITE_STATE_$1F91,X                 ;Mark as active
B9A7      A9 0D          LDA #$0D
B9A9      9D 8C 1E       STA SPRITE_TYPE_$1E8C,X                  ;Cruise Missile
B9AC      20 36 E1       JSR $E136
B9AF      AD 3E 21       LDA $213E
B9B2      F0 07          BEQ $B9BB
B9B4      8A             TXA
B9B5      A8             TAY
B9B6      A6 A6          LDX TEMP6                                ;Restore x from temp variable
B9B8      4C FB BA       JMP RECORD_OPEN_SLOT_$BAFB

B9BB      A9 00          LDA #$00
B9BD      85 B7          STA FRMCNT
B9BF      A0 00          LDY $00
B9C1      20 6E BD       JSR PICK_DIRECTION_$BD6E
B9C4      85 A1          STA TEMP1
B9C6      85 A2          STA TEMP2
B9C8      9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
B9CB      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
B9CE      20 E9 B9       JSR $B9E9
B9D1      20 A8 D3       JSR RANDOM_$D3A8
B9D4      29 07          AND #$07
B9D6      18             CLC
B9D7      69 01          ADC #$01
B9D9      9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X
B9DC      A9 01          LDA #$01
B9DE      9D 7D 1B       STA MTTBL,X
B9E1      A9 12          LDA #$12                                 ;Cruise Missile Fired Sound
B9E3      20 95 E3       JSR DOTUNE_$E395
B9E6      A6 A6          LDX TEMP6
B9E8      60             RTS
;
;
;
B9E9 A5 A2               LDA TEMP2
B9EB 0A   ASL A
B9EC 0A   ASL A
B9ED 0A   ASL A
B9EE 18   CLC
B9EF 65 A1               ADC TEMP1
B9F1 A8   TAY
B9F2 B9 B4 EC            LDA $ECB4,Y
B9F5 9D D9 1C            STA SATBL,X
B9F8 B9 F4 EC            LDA $ECF4,Y
B9FB A8   TAY
B9FC B9 AC EC            LDA $ECAC,Y
B9FF 9D 30 1D            STA MISCTBL_$1D30,X
BA02 60   RTS
;
; Sets an objects X and Y delta directions.
; Called by the Enforcer, spark AI.
; Note the element of randomness that may be permitted by the $B3 and $B4 parameters,
; the Enforcer AI logic uses this sub to fire sparks "near to" the player, for example,
; without always firing at the player directly (which would make the game v. hard)
;
; Inputs
; X = index of spark
; $B1 = X Delta of spark
; $B2 = Y Delta of spark
; $B3 = number to affect spark X delta
; $B4 = number to affect spark Y delta
; Outputs
; objects SPRITE_DELTA_X and SPRITE_DELTA_Y are set
;
SET_OBJECT_DELTAXY_$BA03
BA03     A5 B3          LDA TEMP19
BA05     0A             ASL A
BA06     0A             ASL A
BA07     0A             ASL A
BA08     0A             ASL A
BA09     0A             ASL A
BA0A     85 B3          STA TEMP19                              ;$B3 = $B3 * 32
BA0C     A5 B1          LDA TEMP17
BA0E     29 1F          AND #$1F
BA10     05 B3          ORA TEMP19
BA12     9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
BA15     A5 B4          LDA TEMP20
BA17     0A             ASL A
BA18     0A             ASL A
BA19     0A             ASL A
BA1A     0A             ASL A
BA1B     0A             ASL A
BA1C     85 B4          STA TEMP20                              ;$B4 = $B4 * 32
BA1E     A5 B2          LDA TEMP18
BA20     29 1F          AND #$1F
BA22     05 B4          ORA TEMP20
BA24     9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
BA27     60             RTS
;
; Input
; x = index of object
;
; Returns
; $B1 = adjusted X Delta
; $B2 = adjusted Y Delta
; $B3 = ???
; $B4 = ???
;
ALTER_DELTAS_$BA28
BA28     BD D4 1B       LDA SPRITE_DELTA_X_$1BD4,X
BA2B     85 B3          STA TEMP19
BA2D     29 1F          AND #$1F
BA2F     85 B1          STA TEMP17
BA31     29 10          AND #$10
BA33     F0 06          BEQ $BA3B
BA35     A5 B1          LDA TEMP17
BA37     09 F0          ORA #$F0
BA39     85 B1          STA TEMP17
BA3B     BD 2B 1C       LDA SPRITE_DELTA_Y_$1C2B,X
BA3E     85 B4          STA TEMP20
BA40     29 1F          AND #$1F
BA42     85 B2          STA TEMP18
BA44     29 10          AND #$10
BA46     F0 06          BEQ $BA4E
BA48     A5 B2          LDA TEMP18
BA4A     09 F0          ORA #$F0
BA4C     85 B2          STA TEMP18
BA4E     A5 B3          LDA TEMP19
BA50     30 07          BMI $BA59
BA52     4A             LSR A
BA53     4A             LSR A
BA54     4A             LSR A
BA55     4A             LSR A
BA56     4A             LSR A
BA57     10 07          BPL $BA60
BA59     4A             LSR A
BA5A     4A             LSR A
BA5B     4A             LSR A
BA5C     4A             LSR A
BA5D     4A             LSR A
BA5E     09 F8          ORA #$F8
BA60     85 B3          STA TEMP19
```

```
BA62     A5 B4          LDA TEMP20
BA64     30 07          BMI $BA6D
BA66     4A             LSR A
BA67     4A             LSR A
BA68     4A             LSR A
BA69     4A             LSR A
BA6A     4A             LSR A
BA6B     10 07          BPL $BA74
BA6D     4A             LSR A
BA6E     4A             LSR A
BA6F     4A             LSR A
BA70     4A             LSR A
BA71     4A             LSR A
BA72     09 F8          ORA #$F8
BA74     85 B4          STA TEMP20
BA76     60             RTS

BA77     86 A2          STX TEMP2
BA79     20 34 92       JSR GET_MISSILE_SLOT_$9234
BA7C     A6 A2          LDX TEMP2
BA7E     A8             TAY
BA7F     10 03          BPL $BA84
BA81     4C 0C BB       JMP $BB0C
BA84     A9 01          LDA #$01
BA86     99 91 1F       STA SPRITE_STATE_$1F91,Y
BA89     BD CF 1A       LDA SPRITE_X,X
BA8C     99 CF 1A       STA SPRITE_X,Y
BA8F     18             CLC
BA90     69 07          ADC #$07
BA92     99 E3 1E       STA SPRITE_X_EXTENT,Y
BA95     BD 26 1B       LDA SPRITE_Y,X
BA98     99 26 1B       STA SPRITE_Y,Y
BA9B     18             CLC
BA9C     69 0A          ADC #$0A
BA9E     99 3A 1F       STA SPRITE_Y_EXTENT,Y
BAA1     A9 08          LDA #$08
BAA3     99 8C 1E       STA SPRITE_TYPE_$1E8C,Y
BAA6     84 A4          STY TEMP4
BAA8     86 A5          STX TEMP5
BAAA     98             TYA
BAAB     AA             TAX
BAAC     20 36 E1       JSR $E136
BAAF     A4 A4          LDY TEMP4
BAB1     A6 A5          LDX TEMP5
BAB3     AD 3E 21       LDA $213E
BAB6     D0 43          BNE RECORD_OPEN_SLOT_BAFB
BAB8     A9 00          LDA #$00
BABA     99 D9 1C       STA SATBL,Y
BABD     A9 00          LDA #$00
BABF     99 D4 1B       STA SPRITE_DELTA_X_$1BD4,Y
BAC2     99 2B 1C       STA SPRITE_DELTA_Y_$1C2B,Y
BAC5     A9 01          LDA #$01
BAC7     99 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,Y
BACA     A9 04          LDA #$04
BACC     99 7D 1B       STA MTTBL,X
BACF     99 30 1D       STA MISCTBL_$1D30,Y
BAD2     A0 07          LDY $07
BAD4     B9 26 19       LDA $1926,Y
BAD7     30 03          BMI $BADC
BAD9     88             DEY
BADA     D0 F8          BNE $BAD4
BADC     A9 04          LDA #$04
BADE     99 26 19       STA $1926,Y
BAE1     98             TYA
BAE2     18             CLC
BAE3     6A             ROR A
BAE4     6A             ROR A
BAE5     6A             ROR A
BAE6     6A             ROR A
BAE7     A4 A4          LDY TEMP4
BAE9     19 8C 1E       ORA SPRITE_TYPE_$1E8C,Y
BAEC     99 8C 1E       STA SPRITE_TYPE_$1E8C,Y
BAEF     E6 ED          INC $ED
```

```
BAF1      E6 C9          INC CRELEFT
BAF3      A9 0A          LDA #$0A                             ;Play Enforcer Spark Sound
BAF5      20 95 E3       JSR DOTUNE_$E395
BAF8      A9 00          LDA #$00
BAFA      60             RTS
;
; Record open slot for an object to be created in
;
; Used when Missiles are created and a "free" object needs to be found
;  or when Quarks give birth to tanks and a "free" object needs to be found for the
;  tank or when Spheroids give birth to Enforcer and a "free" object needs to be
;  found for the Enforcer.
;
; Expects
; Y = index of slot (into object table)
;
; Returns
; A = #$80  (must be so that BMI instruction can fire)
;
RECORD_OPEN_SLOT_BAFB
BAFB      A5 EF          LDA $EF
BAFD      84 EF          STY $EF
BAFF      99 30 1D       STA MISCTBL_$1D30,Y
BB02      A9 00          LDA #$00
BB04      99 91 1F       STA SPRITE_STATE_$1F91,Y
BB07      99 8C 1E       STA SPRITE_TYPE_$1E8C,Y
BB0A      A9 80          LDA #$80
BB0C      60             RTS
;
; TANK SHOT AI HANDLER
;
BB0D      BD 91 1F       LDA SPRITE_STATE_$1F91,X             ;Get sprite state
BB10      C9 03          CMP #$03
BB12      F0 05          BEQ $BB19
BB14      DE 30 1D       DEC TANK_SHOT_LIFE,X                 ;Decrement lifespan counter
BB17      D0 08          BNE $BB21                            ;If non-zero, it's still alive
BB19      20 1E D7       JSR $D71E                            ;Otherwise, lifespan is zero,
                                                              ; so kill this sprite
BB1C      C6 7D          DEC $7D                              ;Reduce count of tank shots
BB1E      4C FC 91       JMP OBJCONT_$91FC                    ;PROCESS NEXT OBJECT
;
; if we get here the tank shot's "alive"
; What we do here is check to see if the tank shot is at any "edge"
; and if so, make the shot "bounce" off the wall.
;
BB21      BD CF 1A       LDA SPRITE_X,X                       ;Get tank shot X
BB24      18             CLC
BB25      7D D4 1B       ADC SPRITE_DELTA_X_$1BD4,X           ;Add its delta
BB28      85 BE          STA XINTEND_BE
BB2A      18             CLC
BB2B      69 04          ADC #$04
BB2D      85 C0          STA XXINTEND_C0                      ;Update its X extent (the
usual, you
                                                              : see this in all routines)
BB2F      BD 26 1B       LDA SPRITE_Y,X
BB32      18             CLC
BB33      7D 2B 1C       ADC SPRITE_DELTA_Y_$1C2B,X
BB36      85 BF          STA YINTEND_BF
BB38      18             CLC
BB39      69 05          ADC #$05
BB3B      85 C1          STA YYINTEND_C1

; Now, we do the check to see where the shell is
BB3D      A5 BE          LDA XINTEND_BE
BB3F      18             CLC
BB40      69 10          ADC #$10
BB42      C9 12          CMP #$12                             ;#$12 (left edge)?
BB44      90 06          BCC $BB4C                            ; < #$12, bounce!!
BB46      A5 C0          LDA XXINTEND_C0
BB48      C9 9C          CMP #$9C                             ;#$9C (right edge)?
BB4A      90 11          BCC $BB5D                            ;Less, now check intended Y

; if we get here, time to bounce off the left or right edge!!
```

```
BB4C      38              SEC
BB4D      A9 00           LDA #$00
BB4F      FD D4 1B        SBC SPRITE_DELTA_X_$1BD4,X                    ;Make delta X = -delta X, to
reverse
                                                                        ; X direction

BB52      9D D4 1B        STA SPRITE_DELTA_X_$1BD4,X
BB55      A9 0E           LDA #$0E                                      ;Play boing noise!!!!!!!
BB57      20 95 E3        JSR DOTUNE_$E395
BB5A      4C 21 BB        JMP $BB21                                     ;We go back and repeat the
checks we
                                                                        ; just made, just for safety's
sake
BB5D      A5 BF           LDA YINTEND_BF                                ;
BB5F      C9 12           CMP #$12                                      ;#$12 (top edge)
BB61      90 06           BCC $BB69                                     ; < #$12, bounce!!
BB63      A5 C1           LDA YYINTEND_C1
BB65      C9 BC           CMP #$BC                                      ;#$BC (bottom edge)
BB67      90 11           BCC $BB7A                                     ; < #$BC, we're OK
BB69      38              SEC
BB6A      A9 00           LDA #$00
BB6C      FD 2B 1C        SBC SPRITE_DELTA_Y_$1C2B,X                    ;Make delta Y = -delta Y, to
reverse
                                                                        ; Y direction

BB6F      9D 2B 1C        STA SPRITE_DELTA_Y_$1C2B,X
BB72      A9 0E           LDA #$0E                                      ;Play "Boing" Noise!!
BB74      20 95 E3        JSR DOTUNE_$E395
BB77      4C 21 BB        JMP $BB21                                     ;Repeat checks again,
eventually the
                                                                        ; checks will all pass and we
get to
                                                                        ; next line...
BB7A      20 AF E1        JSR $E1AF                                     ;Draw tank shot
BB7D      AD 3E 21        LDA $213E
BB80      F0 03           BEQ $BB85
BB82      4C 19 BB        JMP $BB19                                     ;Kill the shot
BB85      BD 82 1C        LDA MOVES_B4_DIR_CHANGE_$1C82,X
BB88      9D 7D 1B        STA MTTBL,X
BB8B      A5 BE           LDA XINTEND_BE
BB8D      9D CF 1A        STA SPRITE_X,X
BB90      A5 C0           LDA XXINTEND_C0
BB92      9D E3 1E        STA SPRITE_X_EXTENT,X
BB95      A5 BF           LDA YINTEND_BF
BB97      9D 26 1B        STA SPRITE_Y,X
BB9A      A5 C1           LDA YYINTEND_C1
BB9C      9D 3A 1F        STA SPRITE_Y_EXTENT,X
BB9F      4C FC 91        JMP OBJCONT_$91FC                             ;PROCESS NEXT OBJECT
;
; Creates a tank shell
;
FIRE_TANK_SHELL
BBA2      86 A6           STX TEMP6                                     ;Save current object (in this
case, a tank) index in temp variable
BBA4      20 34 92        JSR GET_MISSILE_SLOT_$9234
BBA7      10 03           BPL $BBAC                                     ;Branch if we have a slot
BBA9      4C D9 BC        JMP $BCD9                                     ;Otherwise, nothing to do

; at this point, x = index of free missile slot

BBAC      A4 A6           LDY TEMP6                                     ;Restore current object index
in temp variable
BBAE      B9 CF 1A        LDA SPRITE_X,Y                                ;Get Tank's X coord
BBB1      18              CLC
BBB2      69 02           ADC #$02                                      ;Add 2 to get Tank's centre X
BBB4      9D CF 1A        STA SPRITE_X,X
BBB7      18              CLC
BBB8      69 04           ADC #$04
BBBA      9D E3 1E        STA SPRITE_X_EXTENT,X
BBBD      B9 26 1B        LDA SPRITE_Y,Y
BBC0      18              CLC
BBC1      69 06           ADC #$06
BBC3      9D 26 1B        STA SPRITE_Y,X                                ;6 to get Tank's centre Y
BBC6      18              CLC
BBC7      69 05           ADC #$05
```

```
BBC9      9D 3A 1F       STA SPRITE_Y_EXTENT,X
BBCC      A9 01          LDA #$01                                        ;Make active
BBCE      9D 91 1F       STA SPRITE_STATE_$1F91,X
BBD1      A9 0E          LDA #$0E                                        ;Tank shot
BBD3      9D 8C 1E       STA SPRITE_TYPE_$1E8C,X
BBD6      20 36 E1       JSR $E136
BBD9      AD 3E 21       LDA $213E                                       ;Is this shell allowed to be
fired?
BBDC      F0 07          BEQ $BBE5                                       ;Yes, so set up the rest of the
shell
                                                                         ; parameters.

; if we get here, the shell can't be fired.
BBDE      8A             TXA                                             ;Let's record its index for
use later as a
                                                                         ; free missile "slot"
BBDF      A8             TAY                                             ;Y now = free slot index
BBE0      A6 A6          LDX TEMP6                                       ;Restore object index from
temp var. X now
                                                                         ; = index of tank that fired
shell
BBE2      4C FB BA       JMP RECORD_OPEN_SLOT_$BAFB

; if we get here
BBE5      20 A8 D3       JSR RANDOM_$D3A8
BBE8      29 03          AND #$03
BBEA      F0 03          BEQ $BBEF
BBEC      4C 99 BC       JMP $BC99
BBEF      20 A8 D3       JSR RANDOM_$D3A8
BBF2      29 03          AND #$03
BBF4      D0 2E          BNE $BC24
BBF6      20 A8 D3       JSR RANDOM_$D3A8
BBF9      29 03          AND #$03
BBFB      A8             TAY
BBFC      C0 00          CPY #$00
BBFE      D0 07          BNE $BC07
BC00      A9 02          LDA #$02
BC02      85 B8          STA TEMPX
BC04      4C 62 BC       JMP $BC62
BC07      C0 01          CPY #$01
BC09      D0 07          BNE $BC12
BC0B      A9 9C          LDA #$9C
BC0D      85 B8          STA TEMPX
BC0F      4C 62 BC       JMP $BC62
BC12      C0 02          CPY #$02
BC14      D0 07          BNE $BC1D
BC16      A9 12          LDA #$12
BC18      85 B9          STA TEMPY
BC1A      4C 7D BC       JMP $BC7D
BC1D      A9 BC          LDA #$BC
BC1F      85 B9          STA TEMPY
BC21      4C 7D BC       JMP $BC7D
BC24      AD CF 1A       LDA SPRITE_X
BC27      C9 4D          CMP #$4D
BC29      B0 0B          BCS $BC36
BC2B      38             SEC
BC2C      E9 02          SBC #$02
BC2E      85 A1          STA TEMP1
BC30      A9 02          LDA #$02
BC32      85 B8          STA TEMPX
BC34      D0 0A          BNE $BC40
BC36      38             SEC
BC37      A9 9C          LDA #$9C
BC39      85 B8          STA TEMPX
BC3B      ED CF 1A       SBC SPRITE_X
BC3E      85 A1          STA TEMP1
BC40      AD 26 1B       LDA SPRITE_Y
BC43      C9 67          CMP #$67
BC45      B0 0B          BCS $BC52
BC47      38             SEC
BC48      E9 12          SBC #$12
BC4A      85 A2          STA TEMP2
BC4C      A9 12          LDA #$12
```

```
BC4E     85 B9          STA TEMPY
BC50     D0 0A          BNE $BC5C
BC52     38             SEC
BC53     A9 BC          LDA #$BC
BC55     85 B9          STA TEMPY
BC57     ED 26 1B       SBC SPRITE_Y
BC5A     85 A2          STA TEMP2
BC5C     A5 A1          LDA TEMP1
BC5E     C5 A2          CMP TEMP2
BC60     B0 1B          BCS $BC7D
BC62     AD 26 1B       LDA SPRITE_Y
BC65     85 B7          STA FRMCNT
BC67     4A             LSR A
BC68     85 A1          STA TEMP1
BC6A     86 A3          STX TEMP3
BC6C     A6 A6          LDX TEMP6
BC6E     BD 26 1B       LDA SPRITE_Y,X
BC71     4A             LSR A
BC72     38             SEC
BC73     E5 A1          SBC TEMP1
BC75     18             CLC
BC76     65 B7          ADC FRMCNT
BC78     85 B9          STA TEMPY
BC7A     4C 95 BC       JMP $BC95
BC7D     AD CF 1A       LDA SPRITE_X
BC80     85 B7          STA FRMCNT
BC82     4A             LSR A
BC83     85 A1          STA TEMP1
BC85     86 A3          STX TEMP3
BC87     A6 A6          LDX TEMP6
BC89     BD CF 1A       LDA SPRITE_X,X
BC8C     4A             LSR A
BC8D     38             SEC
BC8E     E5 A1          SBC TEMP1
BC90     18             CLC
BC91     65 B7          ADC FRMCNT
BC93     85 B8          STA TEMPX
BC95     A6 A3          LDX TEMP3
BC97     D0 10          BNE $BCA9
BC99     AD CF 1A       LDA SPRITE_X
BC9C     18             CLC
BC9D     69 02          ADC #$02
BC9F     85 B8          STA TEMPX
BCA1     AD 26 1B       LDA SPRITE_Y
BCA4     18             CLC
BCA5     69 04          ADC #$04
BCA7     85 B9          STA TEMPY
BCA9     A9 80          LDA #$80
BCAB     85 B7          STA FRMCNT
BCAD     20 DC BC       JSR COMPUTE_DELTAS_$BCDC
BCB0     A5 AA          LDA TEMP10
BCB2     C9 02          CMP #$02
BCB4     B0 02          BCS $BCB8
BCB6     A9 02          LDA #$02
BCB8     9D 82 1C       STA MOVES_B4_DIR_CHANGE_$1C82,X
BCBB     9D 7D 1B       STA MTTBL,X
BCBE     A5 B1          LDA TEMP17
BCC0     9D D4 1B       STA SPRITE_DELTA_X_$1BD4,X
BCC3     A5 B2          LDA TEMP18
BCC5     9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X
BCC8     A9 00          LDA #$00
BCCA     9D D9 1C       STA SATBL,X
BCCD     A9 24          LDA #$24
BCCF     9D 30 1D       STA TANK_SHOT_LIFE_$1D30,X
BCD2     A9 13          LDA #$13                                ;Tank Shot Life Sound
BCD4     20 95 E3       JSR DOTUNE_$E395
BCD7     E6 7D          INC $7D
BCD9     A6 A6          LDX TEMP6
BCDB     60             RTS
;
; This is used by the Enforcer and the tanks.  I believe it computes the deltas
;   required to get from X1, Y1 to X2, Y2.
;
```

```
; Inputs
; x = index of object
; $B8 = target X coordinate
; $B9 = target Y coordinate
;
; Outputs
; $A2 = 1 if the result of the X coord subtraction caused a carry
; $A4 = 1 if the result of the Y coord subtraction caused a carry
; $B1 = result of X coord subtraction (the X delta result)
; $B2 = result of Y coord subtraction (the Y delta result)
; $AA = (I believe) a delay factor when moving
;
COMPUTE_DELTAS_BCDC
BCDC     38          SEC
BCDD     A5 B8       LDA TEMPX                              ;Get X delta parameter
BCDF     FD CF 1A    SBC SPRITE_X,X                         ;Subtract from X
BCE2     85 B1       STA TEMP17                             ;X result goes in $B1
BCE4     A9 00       LDA #$00
BCE6     2A          ROL A
BCE7     85 A2       STA TEMP2                              ;Will be 1 if the subtraction
before
                                                            ; caused a carry
BCE9     38          SEC
BCEA     A5 B9       LDA TEMPY                              ;Get Y delta parameter
BCEC     FD 26 1B    SBC SPRITE_Y,X                         ;Subtract from Y
BCEF     85 B2       STA TEMP18                             ;Result goes in $B2
BCF1     A9 00       LDA #$00
BCF3     2A          ROL A
BCF4     85 A4       STA TEMP4                              ;Will be 1 if the subtraction
before
                                                            ; caused a carry
BCF6     D0 07       BNE $BCFF
BCF8     38          SEC
BCF9     A9 00       LDA #$00
BCFB     E5 B2       SBC TEMP18                             ;
BCFD     85 B2       STA TEMP18                             ;Y result = -Y result, store in
$B2
BCFF     A5 A2       LDA TEMP2
BD01     D0 07       BNE $BD0A
BD03     A9 00       LDA #$00
BD05     38          SEC
BD06     E5 B1       SBC TEMP17
BD08     85 B1       STA TEMP17                             ;X result = -X result, store in
B1
BD0A     18          CLC
BD0B     A5 B1       LDA TEMP17                             ;Get X result
BD0D     65 B2       ADC TEMP18                             ;Add Y result
BD0F     6A          ROR A                                  ;Divide by 2, but add in the
carry also
                                                            ; (which, as it would go in the
high bit,
                                                            ; would make the result
negative if carry
                                                            ; was set)
BD10     85 A5       STA TEMP5                              ;We'll call this product,
store it in $A5
BD12     A9 00       LDA #$00
BD14     85 AA       STA TEMP10
BD16     A5 A5       LDA TEMP5
BD18     E6 AA       INC TEMP10                             ;$AA = 1
BD1A     4A          LSR A                                  ;Product = product div 2 (no,
I don't know
                                                            ; why either).
BD1B     D0 FB       BNE $BD18
BD1D     A9 0A       LDA #$0A
BD1F     85 A1       STA TEMP1
BD21     A5 B7       LDA FRMCNT
BD23     10 04       BPL $BD29
BD25     A9 09       LDA #$09
BD27     85 A1       STA TEMP1
BD29     38          SEC
BD2A     A5 A1       LDA TEMP1
BD2C     E5 AA       SBC TEMP10
```

```
BD2E      85 AA          STA TEMP10
BD30      A9 09          LDA #$09
BD32      85 A1          STA TEMP1
BD34      A5 B7          LDA FRMCNT
BD36      F0 06          BEQ $BD3E
BD38      30 04          BMI $BD3E
BD3A      A9 05          LDA #$05
BD3C      85 A1          STA TEMP1
BD3E      A5 B1          LDA TEMP17
BD40      C5 A1          CMP TEMP1
BD42      B0 06          BCS $BD4A
BD44      A5 B2          LDA TEMP18
BD46      C5 A1          CMP TEMP1
BD48      90 0D          BCC $BD57
BD4A      A5 B1          LDA TEMP17
BD4C      4A             LSR A
BD4D      85 B1          STA TEMP17
BD4F      A5 B2          LDA TEMP18
BD51      4A             LSR A
BD52      85 B2          STA TEMP18
BD54      4C 3E BD       JMP $BD3E
BD57      A5 A2          LDA TEMP2
BD59      D0 07          BNE $BD62
BD5B      A9 00          LDA #$00
BD5D      38             SEC
BD5E      E5 B1          SBC TEMP17
BD60      85 B1          STA TEMP17
BD62      A5 A4          LDA TEMP4
BD64      D0 07          BNE $BD6D
BD66      A9 00          LDA #$00
BD68      38             SEC
BD69      E5 B2          SBC TEMP18
BD6B      85 B2          STA TEMP18
BD6D      60             RTS

;Comments in green compliments of Dan Boris & "Scotty"
; Get direction entity needs to travel in order for obj1 to get to obj2's position.
;   Used, for example, by Brains to determine direction to move to get to family
;   members or player
;
; Expects
; x to be index of an object (obj1) - usually the current object being processed by the game (e.g.
the Brain)
; y to be index of an object (obj2) - usually the target of the object (e.g. a family member)
; $B7 - set to 0 if you want the best direction to be picked, to get obj1 to obj2 in the shortest time.
; 1 set to 1 if you want to randomize things a bit (don't ask me what that means,
;   I just know there's randomness if $B7 == 1)
;
; Returns
; $B8 = -1 if obj2 is to the left of obj1.
;       1 if obj2 is to the right of obj1.
; $B9 = -1 if obj2 is above obj1.
;       1 if obj2 is below obj1.
; A = new direction
;
; PICK_DIRECTION_BD6E
;
BCHASE:
BD6E      A9 00          LDA #$00                                    ;PUT A ZERO INCREMENT IN X AND
Y
BD70      85 B8          STA TEMPX
BD72      85 B9          STA TEMPY
BD74      B9 CF 1A       LDA XTBL,Y                                  ;GET TARGET'S X POSITION
BD77      DD E3 1E       CMP XEXTBL,X                                ;COMPARE WITH BRAIN'S RIGHT
EDGE
BD7A      90 02          BCC BCHASE1                                 ;NOT YET THERE
BD7C      E6 B8          INC TEMPX

BCHASE1:
BD7E      B9 E3 1E       LDA XEXTBL,Y                                ;GET TARGET'S RIGHT EDGE
BD81      DD CF 1A       CMP XTBL,X                                  ;COMPARE WITH BRAIN'S LEFT
BD84      B0 02          BCS BCHASE2                                 ;NOT ON TOP
BD86      C6 B8          DEC TEMPX
```

```
BCHASE2:
BD88      B9 26 1B       LDA YTBL,Y                                  ;GET TARGET'S Y POSITION
BD8B      DD 3A 1F       CMP YEXTBL,X                                ;COMPARE WITH BRAIN'S FAR EDGE
BD8E      90 02          BCC BCHASE3                                 ;NOT YET THERE
BD90      E6 B9          INC TEMPY

BCHASE3:
BD92      B9 3A 1F       LDA YEXTBL,Y                                ;GET TARGET'S FAR EDGE
BD95      DD 26 1B       CMP YTBL,X                                  ;COMPARE WITH BRAIN'S NEAR
BD98      B0 02          BCS BCHASE4                                 ;NOT ON TOP
BD9A      C6 B9          DEC TEMPY

*          NOW THAT WE HAVE THE DIFFERENCE IN TEMP X AND Y
*          CONVERT IT INTO STICK FORM WITH F SIGNIFYING A HIT

BCHASE4
BD9C      A9 0F          LDA #$0F
BD9E      85 A0          STA TEMP0
BDA0      A5 B8          LDA TEMPX                                   ;GET THE X DIFFERENCE
BDA2      10 06          BPL BNOTLEFT
BDA4      A9 0B          LDA #$0B                                    ;CLEAR WEST BIT
BDA6      85 A0          STA TEMP0
BDA8      D0 06          BNE BTRYY

BNOTLEFT:
BDAA      F0 04          BEQ BTRYY                                   ;A CHECK FOR EAST
BDAC      A9 07          LDA #$07                                    ;CLEAR EAST BIT
BDAE      85 A0          STA TEMP0

BTRYY:
BDB0      A5 B9          LDA TEMPY
BDB2      10 08          BPL BNOTUP                                  ;IT SHOULD CLEAR THE NORTH BIT
BDB4      A5 A0          LDA TEMP0
BDB6      29 0E          AND #$0E
BDB8      85 A0          STA TEMP0
BDBA      D0 08          BNE BSTICK                                  ;THIS ALWAYS BRANCHES

BNOTUP:
BDBC      F0 06          BEQ BSTICK
BDBE      A5 A0          LDA TEMP0
BDC0      29 0D          AND #$0D                                    ;CLEAR THE SOUTH BIT
BDC2      85 A0          STA TEMP0

BSTICK:
BDC4      84 A1          STY TEMP1
BDC6      A5 B7          LDA TEMP0                                   ;GET THE STICK FORM
BDC8      F0 1B          BEQ $BDE5
BDCA      A4 A0          LDY TEMP0
BDCC      C0 0B          CMP #$0B                                    ;SEE IF COMPLETE OVERLAP
BDCE      B0 15          BEQ BPROG
BDD0      C0 07          CPY #$07
BDD2      F0 11          BEQ $BDDE
BDD4      20 A8 D3       JSR RANDOM
BDD7      29 01          AND #$01
BDD9      D0 05          BNE $BDE0
BDDB      98             TAY
BDDC      09 03          ORA #$03

BDDE      D0 03          BNE $BDE3
BDE0      98             TAY
BDE1      09 0C          ORA #$0C
BDE3      85 A0          STA TEMP0
BDE5      A5 A0          LDA TEMP0
BDE7      C9 0F          CMP #$09
BDE9      F0 06          BEQ $BDF1
BDEB      A8             TAY
BDEC      B9 05 EC       LDY $EC05,Y
BDEF      A4 A1          LDY TEMP1
BDF1      60             RTS

BDF2                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
```

```
BE00                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BE10                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BE20                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BE30                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BE40                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BE50                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BE60                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BE70                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BE80                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BE90                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BEA0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BEB0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BEC0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BED0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BEE0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BEF0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

BF00                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BF10                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BF20                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BF30                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BF40                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BF50                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BF60                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BF70                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BF80                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BF90                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BFA0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BFB0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BFC0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BFD0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BFE0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
BFF0                                                                        .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
*                                                       *
*                                                       *
*    ROBOTRON    20-JULY-83                             *
*                12-AUGUST-83          C:00             *
*                19-AUGUST-83          5:30             *
*                                                       *
*         RSTAMPS.S                  STAMP DATA         *
*                                                       *
*********************************************************
```

```
*********************************************************
****  LINE  F
          ORG     STAMPS+$000

C000                    $00,$00                                     ;MC D0 S0
C002                    $00,$00
C004                    $00,$00
C006                    $00,$00                                     ;MC D1
C008                    $00,$00
C00A                    $00,$00
C00C                    $00,$00                                     ;MC D2
C00E                    $00,$00
C010                    $00,$00
C012                    $00,$00                                     ;MC D3
C014                    $00,$00
C016                    $00,$00
C018                    $00,$00                                     ;G D0
C01A                    $00,$00
C01C                    $00,$00
C01E                    $00,$00                                     ;MO D0
C020                    $00,$00
C022                    $00,$00
C024                    $00,$00                                     ;MO D1
C026                    $00,$00
C028                    $00,$00
C02A                    $00,$00                                     ;MO D2
C02C                    $00,$00
C02E                    $00,$00
C030                    $00,$00                                     ;MO D3
C032                    $00,$00
C034                    $00,$00
C036                    $00,$00                                     ;D D0
C038                    $00,$00
C03A                    $00,$00
C03C                    $00,$00                                     ;D D1
C03E                    $00,$00
C040                    $00,$00
C042                    $00,$00                                     ;D D2
C044                    $00,$00
C046                    $00,$00
C048                    $00,$00                                     ;D D3
C04A                    $00,$00
C04C                    $00,$00
C04E                    $00,$00                                     ;MID0
C050                    $00,$00
C052                    $00,$00
C054                    $00,$00                                     ;MID1
C056                    $00,$00
C058                    $00,$00
C05A                    $00,$00                                     ;MID2
C05C                    $00,$00
C05E                    $00,$00
C060                    $00,$00                                     ;MID3
C062                    $00,$00
C064                    $00,$00
C066                    $00,$00,$00                                 ;SK
C069                    $00,$00,$00                                 ;1K
C06C                    $00,$00,$00
C06F                    $00,$00,$00
C072                    $00,$00,$00
C075                    $00,$00,$00                                 ;5K
C078                    $00,$00                                     ;H D0
C07A                    $00,$00
C07C                    $00,$00
C07E                    $00,$00
C080                    $00,$00
C082                    $00,$00
C084                    $00,$00
C086                    $00,$00
C088                    $00,$00                                     ;H D1
C08A                    $00,$00
C08C                    $00,$00
C08E                    $00,$00                                     ;H D2
```

```
C090                    $00,$00
C092                    $00,$00
C094                    $00,$00                                     ;H D3
C096                    $00,$00
C098                    $00,$00
C09A                    $00,$00                                     ;S D0 S0
C09C                    $00,$00
C09E                    $00,$00
C0A0                    $00,$00
C0A2                    $00,$00
C0A4                    $00,$00
C0A6                    $00,$00
C0A8                    $00,$00
C0AA                    $00,$00                                     ;Q D0 S0
C0AC                    $00,$00
C0AE                    $00,$00
C0B0                    $00,$00
C0B2                    $00,$00
C0B4                    $00,$00
C0B6                    $00,$00
C0B8                    $00,$00
C0BA                    $00,$00                                     ;E D0 S0
C0BC                    $00,$00
C0BE                    $00,$00                                     ;T D0 S0
C0C0                    $00,$00
C0C2                    $00,$00
C0C4                    $00,$00
C0C6                    $00,$00                                     ;B D0
C0C8                    $00,$00
C0CA                    $00,$00
C0CC                    $00,$00                                     ;B D1
C0CE                    $00,$00
C0D0                    $00,$00
C0D2                    $00,$00                                     ;B D2
C0D4                    $00,$00
C0D6                    $00,$00
C0D8                    $00,$00                                     ;B D3
C0DA                    $00,$00
C0DC                    $00,$00
C0DE                    $00                                         ;MCSD0
C0DF                    $00                                         ;D5
C0E0                    $00                                         ;D6
C0E1                    $00                                         ;D7
C0E2                    $00                                         ;DD
C0E3                    $00,$00
C0E5                    $00,$00
C0E7                    $00,$00
C0E9                    $00,$00
C0EB                    $00,$00
C0ED                    $00,$00
C0EF                    $00,$00
C0E1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  E
          ORG     STAMPS+$100

C100                    .BYTE $00,$00                               ;MC D0 S0
C102                    .BYTE $00,$00
C104                    .BYTE $00,$00
C106                    .BYTE $00,$00                               ;MC D1
C108                    .BYTE $00,$00
C10A                    .BYTE $00,$00
C10C                    .BYTE $00,$00                               ;MC D2
C10E                    .BYTE $00,$00
C110                    .BYTE $00,$00
C112                    .BYTE $00,$00                               ;MC D3
C114                    .BYTE $00,$00
C116                    .BYTE $00,$00
C118                    .BYTE $00,$00                               ;G D0
C11A                    .BYTE $00,$00
C11C                    .BYTE $00,$00
C11E                    .BYTE $00,$00                               ;MO D0
```

```
C120                    .BYTE $00,$00
C122                    .BYTE $00,$00
C124                    .BYTE $00,$00                           ;MO D1
C126                    .BYTE $00,$00
C128                    .BYTE $00,$00
C12A                    .BYTE $00,$00                           ;MO D2
C12C                    .BYTE $00,$00
C12E                    .BYTE $00,$00
C130                    .BYTE $00,$00                           ;MO D3
C132                    .BYTE $00,$00
C134                    .BYTE $00,$00
C136                    .BYTE $00,$00                           ;D D0
C138                    .BYTE $00,$00
C13A                    .BYTE $00,$00
C13C                    .BYTE $00,$00                           ;D D1
C13E                    .BYTE $00,$00
C140                    .BYTE $00,$00
C142                    .BYTE $00,$00                           ;D D2
C144                    .BYTE $00,$00
C146                    .BYTE $00,$00
C148                    .BYTE $00,$00                           ;D D3
C14A                    .BYTE $00,$00
C14C                    .BYTE $00,$00
C14E                    .BYTE $00,$00                           ;MID0
C150                    .BYTE $00,$00
C152                    .BYTE $00,$00
C154                    .BYTE $00,$00                           ;MID1
C156                    .BYTE $00,$00
C158                    .BYTE $00,$00
C15A                    .BYTE $00,$00                           ;MID2
C15C                    .BYTE $00,$00
C15E                    .BYTE $00,$00
C160                    .BYTE $00,$00                           ;MID3
C162                    .BYTE $00,$00
C164                    .BYTE $00,$00
C166                    .BYTE $00,$00,$00                       ;SK
C169                    .BYTE $00,$00,$00                       ;1K
C16C                    .BYTE $00,$00,$00
C16F                    .BYTE $00,$00,$00
C172                    .BYTE $00,$00,$00
C175                    .BYTE $00,$00,$00                       ;5K
C178                    .BYTE $00,$00                           ;H D0
C17A                    .BYTE $00,$00
C17C                    .BYTE $00,$00
C17E                    .BYTE $00,$00
C180                    .BYTE $00,$00
C182                    .BYTE $00,$00
C184                    .BYTE $00,$50
C186                    .BYTE $14,$00
C188                    .BYTE $00,$00                           ;H D1
C18A                    .BYTE $00,$50
C18C                    .BYTE $14,$00
C18E                    .BYTE $00,$00                           ;H D2
C190                    .BYTE $00,$00
C192                    .BYTE $00,$00
C194                    .BYTE $00,$00                           ;H D3
C196                    .BYTE $00,$00
C198                    .BYTE $00,$00
C19A                    .BYTE $00,$00                           ;S D0 S0
C19C                    .BYTE $00,$00
C19E                    .BYTE $00,$00
C1A0                    .BYTE $00,$00
C1A2                    .BYTE $00,$00
C1A4                    .BYTE $00,$00
C1A6                    .BYTE $00,$00
C1A8                    .BYTE $00,$00
C1AA                    .BYTE $00,$00                           ;Q D0 S0
C1AC                    .BYTE $00,$00
C1AE                    .BYTE $00,$00
C1B0                    .BYTE $00,$00
C1B2                    .BYTE $00,$00
C1B4                    .BYTE $00,$00
C1B6                    .BYTE $00,$00
```

```
C1B8                    .BYTE $00,$00
C1BA                    .BYTE $00,$00                              ;E D0 S0
C1BC                    .BYTE $00,$00
C1BE                    .BYTE $00,$00                              ;T D0 S0
C1C0                    .BYTE $00,$00
C1C2                    .BYTE $00,$00
C1C4                    .BYTE $00,$00
C1C6                    .BYTE $00,$00                              ;B D0
C1C8                    .BYTE $00,$00
C1CA                    .BYTE $00,$00
C1CC                    .BYTE $00,$00                              ;B D1
C1CE                    .BYTE $00,$00
C1D0                    .BYTE $00,$00
C1D2                    .BYTE $00,$00                              ;B D2
C1D4                    .BYTE $00,$00
C1D6                    .BYTE $00,$00
C1D8                    .BYTE $00,$00                              ;B D3
C1DA                    .BYTE $00,$00
C1DC                    .BYTE $00,$00
C1DE                    .BYTE $00                                  ;MCSD0
C1DF                    .BYTE $00                                  ;D5
C1E0                    .BYTE $00                                  ;D6
C1E1                    .BYTE $00                                  ;D7
C1E2                    .BYTE $00                                  ;DD
C1E3                    .BYTE $00,$00                              ;G,BEX0
C1E5                    .BYTE $00,$00                              ;G,BEX1
C1E7                    .BYTE $00,$00                              ;G,BEX2
C1E9                    .BYTE $00,$00                              ;G,BEX3
C1EB                    .BYTE $00,$00                              ;G,BEX4
C1ED                    .BYTE $00,$00                              ;G,BEX5
C1EF                    .BYTE $00,$FF                              ;G,BEX6
C1F1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  D
          ORG     STAMPS+$200

C200                    .BYTE $00,$00                              ;MC D0 S0
C202                    .BYTE $00,$00
C204                    .BYTE $00,$00
C206                    .BYTE $00,$00                              ;MC D1
C208                    .BYTE $00,$00
C20A                    .BYTE $00,$00
C20C                    .BYTE $00,$00                              ;MC D2
C20E                    .BYTE $00,$00
C210                    .BYTE $00,$00
C212                    .BYTE $00,$00                              ;MC D3
C214                    .BYTE $00,$00
C216                    .BYTE $00,$00
C218                    .BYTE $00,$00                              ;G D0
C21A                    .BYTE $00,$00
C21C                    .BYTE $00,$00
C21E                    .BYTE $00,$00                              ;MO D0
C220                    .BYTE $00,$00
C222                    .BYTE $00,$00
C224                    .BYTE $00,$00                              ;MO D1
C226                    .BYTE $00,$00
C228                    .BYTE $00,$00
C22A                    .BYTE $00,$00                              ;MO D2
C22C                    .BYTE $00,$00
C22E                    .BYTE $00,$00
C230                    .BYTE $00,$00                              ;MO D3
C232                    .BYTE $00,$00
C234                    .BYTE $00,$00
C236                    .BYTE $00,$00                              ;D D0
C238                    .BYTE $00,$00
C23A                    .BYTE $00,$00
C23C                    .BYTE $00,$00                              ;D D1
C23E                    .BYTE $00,$00
C240                    .BYTE $00,$00
C242                    .BYTE $00,$00                              ;D D2
C244                    .BYTE $00,$00
C246                    .BYTE $00,$00
```

```
C248                    .BYTE $00,$00                           ;D D3
C24A                    .BYTE $00,$00
C24C                    .BYTE $00,$00
C24E                    .BYTE $00,$00                           ;MID0
C250                    .BYTE $00,$00
C252                    .BYTE $00,$00
C254                    .BYTE $00,$00                           ;MID1
C256                    .BYTE $00,$00
C258                    .BYTE $00,$00
C25A                    .BYTE $00,$00                           ;MID2
C25C                    .BYTE $00,$00
C25E                    .BYTE $00,$00
C260                    .BYTE $00,$00                           ;MID3
C262                    .BYTE $00,$00
C264                    .BYTE $00,$00
C266                    .BYTE $00,$00,$00                       ;SK
C269                    .BYTE $00,$00,$00                       ;1K
C26C                    .BYTE $00,$00,$00
C26F                    .BYTE $00,$00,$00
C272                    .BYTE $00,$00,$00
C275                    .BYTE $00,$00,$00                       ;5K
C278                    .BYTE $00,$00
C27A                    .BYTE $00,$00
C27C                    .BYTE $00,$00
C27E                    .BYTE $00,$00
C280                    .BYTE $00,$00
C282                    .BYTE $14,$50                           ;H D0
C284                    .BYTE $00,$50
C286                    .BYTE $14,$00
C288                    .BYTE $14,$50                           ;H D1
C28A                    .BYTE $00,$50
C28C                    .BYTE $14,$00
C28E                    .BYTE $05,$00                           ;H D2
C290                    .BYTE $50,$00
C292                    .BYTE $50,$00
C294                    .BYTE $01,$40                           ;H D3
C296                    .BYTE $00,$14
C298                    .BYTE $00,$14
C29A                    .BYTE $00,$00                           ;S D0 S0
C29C                    .BYTE $00,$00
C29E                    .BYTE $00,$00
C2A0                    .BYTE $00,$00
C2A2                    .BYTE $00,$00
C2A4                    .BYTE $00,$00
C2A6                    .BYTE $00,$00
C2A8                    .BYTE $00,$00
C2AA                    .BYTE $00,$00                           ;Q D0 S0
C2AC                    .BYTE $00,$00
C2AE                    .BYTE $00,$00
C2B0                    .BYTE $00,$00
C2B2                    .BYTE $00,$00
C2B4                    .BYTE $00,$00
C2B6                    .BYTE $00,$00
C2B8                    .BYTE $00,$00
C2BA                    .BYTE $00,$00                           ;E D0 S0
C2BC                    .BYTE $00,$00
C2BE                    .BYTE $00,$00                           ;T D0 S0
C2C0                    .BYTE $00,$00
C2C2                    .BYTE $00,$00
C2C4                    .BYTE $00,$00
C2C6                    .BYTE $00,$00                           ;B D0
C2C8                    .BYTE $00,$00
C2CA                    .BYTE $00,$00
C2CC                    .BYTE $00,$00                           ;B D1
C2CE                    .BYTE $00,$00
C2D0                    .BYTE $00,$00
C2D2                    .BYTE $00,$00                           ;B D2
C2D4                    .BYTE $00,$00
C2D6                    .BYTE $00,$00
C2D8                    .BYTE $00,$00                           ;B D3
C2DA                    .BYTE $00,$00
C2DC                    .BYTE $00,$00
C2DE                    .BYTE $00                               ;MCSD0
```

```
C2DF                    .BYTE $00                                   ;D5
C2E0                    .BYTE $00                                   ;D6
C2E1                    .BYTE $00                                   ;D7
C2E2                    .BYTE $00                                   ;DD
C2E3                    .BYTE $00,$00                               ;G,BEX0
C2E5                    .BYTE $00,$00                               ;G,BEX1
C2E7                    .BYTE $00,$00                               ;G,BEX2
C2E9                    .BYTE $00,$00                               ;G,BEX3
C2EB                    .BYTE $00,$00                               ;G,BEX4
C2ED                    .BYTE $00,$00                               ;G,BEX5
C2EF                    .BYTE $00,$FF                               ;G,BEX6
C2F1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  C
          ORG     STAMPS+$300

C300                    .BYTE $00,$00                               ;MC D0 S0
C302                    .BYTE $50,$00
C304                    .BYTE $01,$40
C306                    .BYTE $00,$00                               ;MC D1
C308                    .BYTE $50,$00
C30A                    .BYTE $01,$40
C30C                    .BYTE $00,$00                               ;MC D2
C30E                    .BYTE $00,$00
C310                    .BYTE $00,$00
C312                    .BYTE $00,$00                               ;MC D3
C314                    .BYTE $00,$00
C316                    .BYTE $00,$00
C318                    .BYTE $00,$00                               ;G D0
C31A                    .BYTE $00,$00
C31C                    .BYTE $00,$00
C31E                    .BYTE $00,$00                               ;MO D0
C320                    .BYTE $00,$00
C322                    .BYTE $00,$00
C324                    .BYTE $00,$00                               ;MO D1
C326                    .BYTE $00,$F0
C328                    .BYTE $00,$00
C32A                    .BYTE $04,$00                               ;MO D2
C32C                    .BYTE $10,$00
C32E                    .BYTE $00,$00
C330                    .BYTE $04,$00                               ;MO D3
C332                    .BYTE $01,$00
C334                    .BYTE $00,$00
C336                    .BYTE $00,$00                               ;D D0
C338                    .BYTE $00,$00
C33A                    .BYTE $00,$00
C33C                    .BYTE $00,$00                               ;D D1
C33E                    .BYTE $00,$00
C340                    .BYTE $00,$00
C342                    .BYTE $14,$00                               ;D D2
C344                    .BYTE $00,$C0
C346                    .BYTE $00,$00
C348                    .BYTE $10,$00                               ;D D3
C34A                    .BYTE $01,$00
C34C                    .BYTE $00,$00
C34E                    .BYTE $00,$00                               ;MID0
C350                    .BYTE $00,$00
C352                    .BYTE $00,$00
C354                    .BYTE $00,$00                               ;MID1
C356                    .BYTE $00,$00
C358                    .BYTE $00,$00
C35A                    .BYTE $00,$00                               ;MID2
C35C                    .BYTE $00,$00
C35E                    .BYTE $00,$00
C360                    .BYTE $00,$00                               ;MID3
C362                    .BYTE $00,$00
C364                    .BYTE $00,$00
C366                    .BYTE $00,$00,$00                           ;SK
C369                    .BYTE $00,$00,$00                           ;1K
C36C                    .BYTE $00,$00,$00
C36F                    .BYTE $00,$00,$00
C372                    .BYTE $00,$00,$00
```

```
C375                    .BYTE $00,$00,$00                           ;5K
C378                    .BYTE $00,$00
C37A                    .BYTE $00,$00
C37C                    .BYTE $00,$00
C37E                    .BYTE $00,$00
C380                    .BYTE $00,$00
C382                    .BYTE $14,$50                               ;H D0
C384                    .BYTE $00,$40
C386                    .BYTE $04,$00
C388                    .BYTE $14,$50                               ;H D1
C38A                    .BYTE $00,$40
C38C                    .BYTE $04,$00
C38E                    .BYTE $05,$00                               ;H D2
C390                    .BYTE $40,$50
C392                    .BYTE $40,$50
C394                    .BYTE $01,$40                               ;H D3
C396                    .BYTE $14,$04
C398                    .BYTE $14,$04
C39A                    .BYTE $00,$00                               ;S D0 S0
C39C                    .BYTE $00,$00
C39E                    .BYTE $00,$00
C3A0                    .BYTE $00,$00
C3A2                    .BYTE $00,$00
C3A4                    .BYTE $00,$00
C3A6                    .BYTE $00,$80
C3A8                    .BYTE $00,$00
C3AA                    .BYTE $00,$00                               ;Q D0 S0
C3AC                    .BYTE $00,$00
C3AE                    .BYTE $0C,$00
C3B0                    .BYTE $00,$C0
C3B2                    .BYTE $00,$00
C3B4                    .BYTE $0C,$00
C3B6                    .BYTE $00,$0C
C3B8                    .BYTE $00,$00
C3BA                    .BYTE $00,$00                               ;E D0 S0
C3BC                    .BYTE $00,$00
C3BE                    .BYTE $00,$00                               ;T D0 S0
C3C0                    .BYTE $00,$00
C3C2                    .BYTE $00,$00
C3C4                    .BYTE $00,$00
C3C6                    .BYTE $00,$00                               ;B D0
C3C8                    .BYTE $00,$00
C3CA                    .BYTE $00,$00
C3CC                    .BYTE $00,$00                               ;B D1
C3CE                    .BYTE $00,$00
C3D0                    .BYTE $00,$00
C3D2                    .BYTE $00,$00                               ;B D2
C3D4                    .BYTE $00,$00
C3D6                    .BYTE $00,$00
C3D8                    .BYTE $00,$00                               ;B D3
C3DA                    .BYTE $00,$00
C3DC                    .BYTE $00,$00
C3DE                    .BYTE $00                                   ;MCSD0
C3DF                    .BYTE $00                                   ;D5
C3E0                    .BYTE $00                                   ;D6
C3E1                    .BYTE $00                                   ;D7
C3E2                    .BYTE $00                                   ;DD
C3E3                    .BYTE $00,$00                               ;G,BEX0
C3E5                    .BYTE $00,$00                               ;G,BEX1
C3E7                    .BYTE $00,$00                               ;G,BEX2
C3E9                    .BYTE $00,$00                               ;G,BEX3
C3EB                    .BYTE $00,$00                               ;G,BEX4
C3ED                    .BYTE $FF,$FF                               ;G,BEX5
C3EF                    .BYTE $FF,$FF                               ;G,BEX6
C3F1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  B
          ORG     STAMPS+$400

C400                    .BYTE $51,$40                               ;MC D0 S0
C402                    .BYTE $10,$00
C404                    .BYTE $01,$00
```

```
C406                    .BYTE $51,$40                           ;MC D1
C408                    .BYTE $10,$00
C40A                    .BYTE $01,$00
C40C                    .BYTE $15,$40                           ;MC D2
C40E                    .BYTE $14,$50
C410                    .BYTE $51,$50
C412                    .BYTE $15,$00                           ;MC D3
C414                    .BYTE $14,$50                           ;TEMP TRY
C416                    .BYTE $51,$40
C418                    .BYTE $00,$00                           ;G D0
C41A                    .BYTE $51,$40
C41C                    .BYTE $51,$40
C41E                    .BYTE $00,$00                           ;MO D0
C420                    .BYTE $51,$40
C422                    .BYTE $3C,$F0
C424                    .BYTE $3C,$00                           ;MO D1
C426                    .BYTE $00,$80
C428                    .BYTE $55,$00
C42A                    .BYTE $04,$00                           ;MO D2
C42C                    .BYTE $10,$00
C42E                    .BYTE $15,$40
C430                    .BYTE $04,$00                           ;MO D3
C432                    .BYTE $01,$00
C434                    .BYTE $04,$00
C436                    .BYTE $11,$00                           ;D D0
C438                    .BYTE $11,$00
C43A                    .BYTE $10,$00
C43C                    .BYTE $44,$00                           ;D D1
C43E                    .BYTE $44,$00
C440                    .BYTE $14,$50
C442                    .BYTE $04,$00                           ;D D2
C444                    .BYTE $00,$40
C446                    .BYTE $51,$40
C448                    .BYTE $10,$00                           ;D D3
C44A                    .BYTE $01,$00
C44C                    .BYTE $28,$00
C44E                    .BYTE $28,$A0                           ;MID0
C450                    .BYTE $28,$A0
C452                    .BYTE $28,$00
C454                    .BYTE $A2,$80                           ;MID1
C456                    .BYTE $A2,$80
C458                    .BYTE $00,$00
C45A                    .BYTE $00,$00                           ;MID2
C45C                    .BYTE $00,$00
C45E                    .BYTE $00,$00
C460                    .BYTE $00,$00                           ;MID3
C462                    .BYTE $00,$00
C464                    .BYTE $00,$00
C466                    .BYTE $00,$00,$00                       ;SK
C469                    .BYTE $00,$00,$00                       ;1K
C46C                    .BYTE $00,$00,$00
C46F                    .BYTE $00,$00,$00
C472                    .BYTE $00,$00,$00
C475                    .BYTE $00,$00,$00                       ;5K
C478                    .BYTE $00,$00
C47A                    .BYTE $00,$00
C47C                    .BYTE $00,$00
C47E                    .BYTE $00,$00
C480                    .BYTE $00,$00
C482                    .BYTE $04,$40                           ;H D0
C484                    .BYTE $14,$40
C486                    .BYTE $04,$50
C488                    .BYTE $04,$40                           ;H D1
C48A                    .BYTE $14,$40
C48C                    .BYTE $04,$50
C48E                    .BYTE $04,$00                           ;H D2
C490                    .BYTE $40,$44
C492                    .BYTE $40,$44
C494                    .BYTE $00,$40                           ;H D3
C496                    .BYTE $44,$04
C498                    .BYTE $44,$04
C49A                    .BYTE $00,$00                           ;S D0 S0
C49C                    .BYTE $00,$00
```

```
C49E                    .BYTE $00,$00
C4A0                    .BYTE $00,$00
C4A2                    .BYTE $00,$00
C4A4                    .BYTE $80,$00
C4A6                    .BYTE $02,$A0
C4A8                    .BYTE $00,$00
C4AA                    .BYTE $80,$00                             ;Q D0 S0
C4AC                    .BYTE $0C,$00
C4AE                    .BYTE $0C,$00
C4B0                    .BYTE $00,$C0
C4B2                    .BYTE $0C,$00
C4B4                    .BYTE $0C,$00
C4B6                    .BYTE $00,$C0
C4B8                    .BYTE $03,$C0
C4BA                    .BYTE $08,$F0                             ;E D0 S0
C4BC                    .BYTE $0C,$E0
C4BE                    .BYTE $0F,$00                             ;T D0 S0
C4C0                    .BYTE $2C,$C0
C4C2                    .BYTE $3C,$80
C4C4                    .BYTE $00,$00
C4C6                    .BYTE $00,$00                             ;B D0
C4C8                    .BYTE $00,$3C
C4CA                    .BYTE $F0,$2C
C4CC                    .BYTE $E0,$20                             ;B D1
C4CE                    .BYTE $20,$20
C4D0                    .BYTE $20,$00
C4D2                    .BYTE $00,$00                             ;B D2
C4D4                    .BYTE $00,$0C
C4D6                    .BYTE $C0,$00
C4D8                    .BYTE $00,$00                             ;B D3
C4DA                    .BYTE $00,$00
C4DC                    .BYTE $00,$00
C4DE                    .BYTE $80                                 ;MCSD0
C4DF                    .BYTE $00                                 ;D5
C4E0                    .BYTE $00                                 ;D6
C4E1                    .BYTE $5F                                 ;D7
C4E2                    .BYTE $55                                 ;DD
C4E3                    .BYTE $55,$5F                             ;G,BEX0
C4E5                    .BYTE $00,$00                             ;G,BEX1
C4E7                    .BYTE $00,$00                             ;G,BEX2
C4E9                    .BYTE $00,$00                             ;G,BEX3
C4EB                    .BYTE $00,$00                             ;G,BEX4
C4ED                    .BYTE $FF,$FF                             ;G,BEX5
C4EF                    .BYTE $FF,$FF                             ;G,BEX6
C4F1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  A
          ORG     STAMPS+$500

C500                    .BYTE $11,$00                             ;MC D0 S0
C502                    .BYTE $11,$40
C504                    .BYTE $51,$00
C506                    .BYTE $11,$00                             ;MC D1
C508                    .BYTE $11,$40
C50A                    .BYTE $51,$00
C50C                    .BYTE $15,$00                             ;MC D2
C50E                    .BYTE $14,$50
C510                    .BYTE $51,$50
C512                    .BYTE $15,$00                             ;MC D3
C514                    .BYTE $14,$50                             ;TEMP TRY
C516                    .BYTE $51,$40
C518                    .BYTE $00,$00                             ;G D0
C51A                    .BYTE $00,$00
C51C                    .BYTE $00,$00
C51E                    .BYTE $00,$00                             ;MO D0
C520                    .BYTE $00,$00
C522                    .BYTE $08,$80
C524                    .BYTE $08,$F0                             ;MO D1
C526                    .BYTE $3C,$80
C528                    .BYTE $14,$00
C52A                    .BYTE $54,$00                             ;MO D2
C52C                    .BYTE $15,$00
```

```
C52E                     .BYTE $05,$00
C530                     .BYTE $05,$40                                ;MO D3
C532                     .BYTE $15,$00
C534                     .BYTE $04,$00
C536                     .BYTE $11,$00                                ;D D0
C538                     .BYTE $11,$00
C53A                     .BYTE $10,$00
C53C                     .BYTE $44,$00                                ;D D1
C53E                     .BYTE $44,$00
C540                     .BYTE $F4,$40
C542                     .BYTE $04,$50                                ;D D2
C544                     .BYTE $FC,$40
C546                     .BYTE $13,$F0
C548                     .BYTE $13,$F0                                ;D D3
C54A                     .BYTE $51,$00
C54C                     .BYTE $DF,$00
C54E                     .BYTE $1F,$70                                ;MID0
C550                     .BYTE $DF,$40
C552                     .BYTE $F7,$00
C554                     .BYTE $DF,$40                                ;MID1
C556                     .BYTE $1F,$70
C558                     .BYTE $51,$40
C55A                     .BYTE $01,$40                                ;MID2
C55C                     .BYTE $50,$00
C55E                     .BYTE $51,$40
C560                     .BYTE $50,$00                                ;MID3
C562                     .BYTE $01,$40
C564                     .BYTE $14,$00
C566                     .BYTE $10,$40,$11                            ;SK
C569                     .BYTE $40,$50,$00                            ;1K
C56C                     .BYTE $41,$00,$51
C56F                     .BYTE $00,$00,$00
C572                     .BYTE $00,$00,$00
C575                     .BYTE $00,$00,$00                            ;5K
C578                     .BYTE $00,$00
C57A                     .BYTE $00,$00
C57C                     .BYTE $00,$00
C57E                     .BYTE $00,$00
C580                     .BYTE $00,$00
C582                     .BYTE $04,$40                                ;H D0
C584                     .BYTE $14,$40
C586                     .BYTE $04,$50
C588                     .BYTE $04,$40                                ;H D1
C58A                     .BYTE $14,$40
C58C                     .BYTE $04,$50
C58E                     .BYTE $04,$00                                ;H D2
C590                     .BYTE $11,$00
C592                     .BYTE $11,$00
C594                     .BYTE $00,$40                                ;H D3
C596                     .BYTE $01,$10
C598                     .BYTE $01,$10
C59A                     .BYTE $00,$00                                ;S D0 S0
C59C                     .BYTE $00,$00
C59E                     .BYTE $00,$00
C5A0                     .BYTE $00,$00
C5A2                     .BYTE $00,$02
C5A4                     .BYTE $A0,$00
C5A6                     .BYTE $00,$00
C5A8                     .BYTE $00,$00
C5AA                     .BYTE $00,$00                                ;Q D0 S0
C5AC                     .BYTE $03,$00
C5AE                     .BYTE $03,$C0
C5B0                     .BYTE $0F,$00
C5B2                     .BYTE $03,$00
C5B4                     .BYTE $03,$C0
C5B6                     .BYTE $0F,$00
C5B8                     .BYTE $03,$00
C5BA                     .BYTE $0B,$00                                ;E D0 S0
C5BC                     .BYTE $0F,$00
C5BE                     .BYTE $03,$00                                ;T D0 S0
C5C0                     .BYTE $03,$C0
C5C2                     .BYTE $03,$80
C5C4                     .BYTE $00,$00
```

```
C5C6                    .BYTE $00,$00                               ;B D0
C5C8                    .BYTE $00,$08
C5CA                    .BYTE $80,$08
C5CC                    .BYTE $80,$08                               ;B D1
C5CE                    .BYTE $00,$00
C5D0                    .BYTE $00,$00
C5D2                    .BYTE $80,$00                               ;B D2
C5D4                    .BYTE $00,$30
C5D6                    .BYTE $30,$03
C5D8                    .BYTE $00,$00                               ;B D3
C5DA                    .BYTE $00,$00
C5DC                    .BYTE $00,$20
C5DE                    .BYTE $00                                   ;MCSD0
C5DF                    .BYTE $80                                   ;D5
C5E0                    .BYTE $00                                   ;D6
C5E1                    .BYTE $57                                   ;D7
C5E2                    .BYTE $55                                   ;DD
C5E3                    .BYTE $55,$D5                               ;G,BEX0
C5E5                    .BYTE $00,$00                               ;G,BEX1
C5E7                    .BYTE $00,$00                               ;G,BEX2
C5E9                    .BYTE $00,$00                               ;G,BEX3
C5EB                    .BYTE $00,$00                               ;G,BEX4
C5ED                    .BYTE $FF,$FF                               ;G,BEX5
C5EF                    .BYTE $FF,$FF                               ;G,BEX6
C5F1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  9
          ORG     STAMPS+$600

C600                    .BYTE $11,$00                               ;MC D0 S0
C602                    .BYTE $11,$00
C604                    .BYTE $11,$00
C606                    .BYTE $11,$00                               ;MC D1
C608                    .BYTE $11,$00
C60A                    .BYTE $11,$00
C60C                    .BYTE $28,$00                               ;MC D2
C60E                    .BYTE $28,$80
C610                    .BYTE $A2,$00
C612                    .BYTE $0A,$00                               ;MC D3
C614                    .BYTE $08,$A0
C616                    .BYTE $22,$80
C618                    .BYTE $00,$01
C61A                    .BYTE $10,$00
C61C                    .BYTE $01,$10
C61E                    .BYTE $00,$00
C620                    .BYTE $11,$00
C622                    .BYTE $0A,$80                               ;G D0
C624                    .BYTE $0A,$80
C626                    .BYTE $0A,$80
C628                    .BYTE $14,$00                               ;MO D0
C62A                    .BYTE $14,$00
C62C                    .BYTE $14,$00
C62E                    .BYTE $05,$00                               ;MO D1
C630                    .BYTE $05,$00
C632                    .BYTE $05,$00
C634                    .BYTE $28,$00                               ;MO D2
C636                    .BYTE $2A,$00
C638                    .BYTE $2A,$00
C63A                    .BYTE $28,$00                               ;MO D3
C63C                    .BYTE $A8,$00
C63E                    .BYTE $A8,$00
C640                    .BYTE $F4,$40                               ;D D0
C642                    .BYTE $F4,$40
C644                    .BYTE $FC,$40
C646                    .BYTE $13,$F0                               ;D D1
C648                    .BYTE $13,$F0
C64A                    .BYTE $13,$F0
C64C                    .BYTE $DF,$00                               ;D D2
C64E                    .BYTE $1F,$70
C650                    .BYTE $DF,$40
C652                    .BYTE $F7,$00                               ;D D3
C654                    .BYTE $DF,$40
```

```
C656                    .BYTE $1F,$70
C658                    .BYTE $51,$40                           ;MID0
C65A                    .BYTE $51,$40
C65C                    .BYTE $51,$40
C65E                    .BYTE $51,$40                           ;MID1
C660                    .BYTE $51,$40
C662                    .BYTE $51,$40
C664                    .BYTE $14,$00                           ;MID2
C666                    .BYTE $10,$40
C668                    .BYTE $11,$40
C66A                    .BYTE $50,$00                           ;MID3
C66C                    .BYTE $41,$00
C66E                    .BYTE $51,$00
C670                    .BYTE $05,$14,$00                       ;SK
C673                    .BYTE $00,$00,$00                       ;1K
C676                    .BYTE $00,$00,$00
C679                    .BYTE $00,$00,$00
C67C                    .BYTE $00,$00,$00
C67F                    .BYTE $00,$00,$00                       ;5K
C682                    .BYTE $C4,$4C                           ;H D0
C684                    .BYTE $C4,$4C
C686                    .BYTE $C4,$4C
C688                    .BYTE $C4,$4C                           ;H D1
C68A                    .BYTE $C4,$4C
C68C                    .BYTE $C4,$4C
C68E                    .BYTE $04,$00                           ;H D2
C690                    .BYTE $15,$00
C692                    .BYTE $05,$00
C694                    .BYTE $00,$40                           ;H D3
C696                    .BYTE $01,$40
C698                    .BYTE $01,$50
C69A                    .BYTE $00,$00                           ;S D0 S0
C69C                    .BYTE $00,$00
C69E                    .BYTE $00,$00
C6A0                    .BYTE $00,$80
C6A2                    .BYTE $00,$0A
C6A4                    .BYTE $A8,$00
C6A6                    .BYTE $00,$00
C6A8                    .BYTE $00,$00
C6AA                    .BYTE $00,$00                           ;Q D0 S0
C6AC                    .BYTE $33,$30
C6AE                    .BYTE $33,$30
C6B0                    .BYTE $33,$30
C6B2                    .BYTE $33,$30
C6B4                    .BYTE $33,$30
C6B6                    .BYTE $33,$30
C6B8                    .BYTE $03,$F0
C6BA                    .BYTE $03,$F0                           ;E D0 S0
C6BC                    .BYTE $03,$F0
C6BE                    .BYTE $3F,$00                           ;T D0 S0
C6C0                    .BYTE $3F,$00
C6C2                    .BYTE $3F,$00
C6C4                    .BYTE $00,$00
C6C6                    .BYTE $00,$00                           ;B D0
C6C8                    .BYTE $00,$08
C6CA                    .BYTE $80,$08
C6CC                    .BYTE $80,$00                           ;B D1
C6CE                    .BYTE $80,$00
C6D0                    .BYTE $80,$20
C6D2                    .BYTE $00,$80                           ;B D2
C6D4                    .BYTE $00,$03
C6D6                    .BYTE $C0,$30
C6D8                    .BYTE $00,$00                           ;B D3
C6DA                    .BYTE $00,$02
C6DC                    .BYTE $00,$00
C6DE                    .BYTE $00                               ;MCSD0
C6DF                    .BYTE $00                               ;D5
C6E0                    .BYTE $00                               ;D6
C6E1                    .BYTE $7F                               ;D7
C6E2                    .BYTE $55                               ;DD
C6E3                    .BYTE $55,$FD                           ;G,BEX0
C6E5                    .BYTE $00,$00                           ;G,BEX1
C6E7                    .BYTE $00,$00                           ;G,BEX2
```

```
C6E9                    .BYTE $00,$00                              ;G,BEX3
C6EB                    .BYTE $00,$00                              ;G,BEX4
C6ED                    .BYTE $FF,$FF                              ;G,BEX5
C6EF                    .BYTE $FF,$FF                              ;G,BEX6
C6F1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

**********************************************************
****  LINE  8
          ORG     STAMPS+$700

C700                    .BYTE $95,$80                              ;MC D0 S0
C702                    .BYTE $15,$80
C704                    .BYTE $95,$C0
C706                    .BYTE $95,$80                              ;MC D1
C708                    .BYTE $15,$80
C70A                    .BYTE $95,$00
C70C                    .BYTE $15,$00                              ;MC D2
C70E                    .BYTE $15,$80
C710                    .BYTE $56,$00
C712                    .BYTE $15,$00                              ;MC D3
C714                    .BYTE $09,$50
C716                    .BYTE $25,$40
C718                    .BYTE $00,$00
C71A                    .BYTE $00,$00
C71C                    .BYTE $00,$00
C71E                    .BYTE $00,$00
C720                    .BYTE $00,$00
C722                    .BYTE $CA,$8C                              ;G D0
C724                    .BYTE $0A,$8C
C726                    .BYTE $CA,$80
C728                    .BYTE $AA,$00                              ;MO D0
C72A                    .BYTE $AA,$00
C72C                    .BYTE $AA,$C0
C72E                    .BYTE $2A,$80                              ;MO D1
C730                    .BYTE $2A,$80
C732                    .BYTE $FA,$80
C734                    .BYTE $38,$00                              ;MO D2
C736                    .BYTE $2F,$00
C738                    .BYTE $F8,$00
C73A                    .BYTE $2C,$00                              ;MO D3
C73C                    .BYTE $2F,$00
C73E                    .BYTE $F8,$00
C740                    .BYTE $C5,$40                              ;D D0
C742                    .BYTE $F5,$40
C744                    .BYTE $CD,$40
C746                    .BYTE $17,$30                              ;D D1
C748                    .BYTE $17,$30
C74A                    .BYTE $17,$70
C74C                    .BYTE $DF,$00                              ;D D2
C74E                    .BYTE $15,$70
C750                    .BYTE $D5,$40
C752                    .BYTE $F7,$00                              ;D D3
C754                    .BYTE $D5,$40
C756                    .BYTE $15,$70
C758                    .BYTE $22,$00                              ;MID0
C75A                    .BYTE $52,$00
C75C                    .BYTE $21,$40
C75E                    .BYTE $22,$00                              ;MID1
C760                    .BYTE $21,$40
C762                    .BYTE $52,$00
C764                    .BYTE $20,$00                              ;MID2
C766                    .BYTE $22,$00
C768                    .BYTE $22,$00
C76A                    .BYTE $20,$00                              ;MID3
C76C                    .BYTE $22,$00
C76E                    .BYTE $22,$00
C770                    .BYTE $01,$10,$00                          ;SK
C773                    .BYTE $00,$00,$00                          ;1K
C776                    .BYTE $00,$00,$00
C779                    .BYTE $00,$00,$00
C77C                    .BYTE $00,$00,$00
C77F                    .BYTE $00,$00,$00                          ;5K
C782                    .BYTE $EA,$AC                              ;H D0
```

```
C784                    .BYTE $EA,$AC
C786                    .BYTE $EA,$AC
C788                    .BYTE $FA,$BC                              ;H D1
C78A                    .BYTE $FA,$BC
C78C                    .BYTE $FA,$BC
C78E                    .BYTE $AA,$80                              ;H D2
C790                    .BYTE $AA,$80
C792                    .BYTE $AA,$80
C794                    .BYTE $0A,$A8                              ;H D3
C796                    .BYTE $0A,$A8
C798                    .BYTE $0A,$A8
C79A                    .BYTE $00,$00                              ;S D0 S0
C79C                    .BYTE $00,$00
C79E                    .BYTE $80,$00
C7A0                    .BYTE $02,$A0
C7A2                    .BYTE $00,$08
C7A4                    .BYTE $08,$00
C7A6                    .BYTE $20,$02
C7A8                    .BYTE $00,$00
C7AA                    .BYTE $00,$00                              ;Q D0 S0
C7AC                    .BYTE $0F,$C0
C7AE                    .BYTE $0F,$C0
C7B0                    .BYTE $0F,$C0
C7B2                    .BYTE $0F,$C0
C7B4                    .BYTE $0F,$C0
C7B6                    .BYTE $0F,$C0
C7B8                    .BYTE $03,$00
C7BA                    .BYTE $03,$00                              ;E D0 S0
C7BC                    .BYTE $03,$00
C7BE                    .BYTE $03,$00                              ;T D0 S0
C7C0                    .BYTE $03,$00
C7C2                    .BYTE $03,$00
C7C4                    .BYTE $00,$00
C7C6                    .BYTE $00,$00                              ;B D0
C7C8                    .BYTE $00,$00
C7CA                    .BYTE $00,$CA
C7CC                    .BYTE $8C,$C8                              ;B D1
C7CE                    .BYTE $0C,$C8
C7D0                    .BYTE $08,$00
C7D2                    .BYTE $00,$00                              ;B D2
C7D4                    .BYTE $00,$0A
C7D6                    .BYTE $80,$03
C7D8                    .BYTE $00,$02                              ;B D3
C7DA                    .BYTE $00,$02
C7DC                    .BYTE $00,$00
C7DE                    .BYTE $00                                  ;MCSD0
C7DF                    .BYTE $00                                  ;D5
C7E0                    .BYTE $08                                  ;D6
C7E1                    .BYTE $57                                  ;D7
C7E2                    .BYTE $55                                  ;DD
C7E3                    .BYTE $55,$D5                              ;G,BEX0
C7E5                    .BYTE $00,$00                              ;G,BEX1
C7E7                    .BYTE $00,$00                              ;G,BEX2
C7E9                    .BYTE $00,$00                              ;G,BEX3
C7EB                    .BYTE $00,$00                              ;G,BEX4
C7ED                    .BYTE $FF,$FF                              ;G,BEX5
C7EF                    .BYTE $FF,$FF                              ;G,BEX6
C7F1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  7
          ORG     STAMPS+$800

C800                    .BYTE $95,$80                              ;MC D0 S0
C802                    .BYTE $15,$80
C804                    .BYTE $95,$00
C806                    .BYTE $99,$80                              ;MC D1
C808                    .BYTE $99,$80
C80A                    .BYTE $99,$80
C80C                    .BYTE $19,$00                              ;MC D2
C80E                    .BYTE $95,$00
C810                    .BYTE $54,$00
C812                    .BYTE $19,$00                              ;MC D3
```

```
C814                     .BYTE $01,$50
C816                     .BYTE $06,$60
C818                     .BYTE $00,$37
C81A                     .BYTE $70,$00
C81C                     .BYTE $03,$77
C81E                     .BYTE $00,$00
C820                     .BYTE $DD,$C0
C822                     .BYTE $E9,$AC                     ;G D0
C824                     .BYTE $C9,$AC
C826                     .BYTE $E9,$8C
C828                     .BYTE $AB,$C0                     ;MO D0
C82A                     .BYTE $AA,$00
C82C                     .BYTE $AA,$C0
C82E                     .BYTE $FA,$80                     ;MO D1
C830                     .BYTE $2A,$40
C832                     .BYTE $DA,$80
C834                     .BYTE $38,$00                     ;MO D2
C836                     .BYTE $2F,$00
C838                     .BYTE $F8,$00
C83A                     .BYTE $2C,$00                     ;MO D3
C83C                     .BYTE $2F,$00
C83E                     .BYTE $F8,$00
C840                     .BYTE $15,$70                     ;D D0
C842                     .BYTE $D5,$70
C844                     .BYTE $15,$40
C846                     .BYTE $9D,$40                     ;D D1
C848                     .BYTE $1D,$40
C84A                     .BYTE $9F,$40
C84C                     .BYTE $67,$00                     ;D D2
C84E                     .BYTE $15,$F0
C850                     .BYTE $D5,$00
C852                     .BYTE $D9,$00                     ;D D3
C854                     .BYTE $F5,$40
C856                     .BYTE $05,$70
C858                     .BYTE $2A,$00                     ;MID0
C85A                     .BYTE $AA,$00
C85C                     .BYTE $2A,$80
C85E                     .BYTE $2A,$00                     ;MID1
C860                     .BYTE $2A,$80
C862                     .BYTE $AA,$00
C864                     .BYTE $98,$00                     ;MID2
C866                     .BYTE $2A,$40
C868                     .BYTE $69,$40
C86A                     .BYTE $98,$00                     ;MID3
C86C                     .BYTE $6A,$00
C86E                     .BYTE $5A,$40
C870                     .BYTE $04,$44,$00                 ;SK
C873                     .BYTE $00,$00,$00                 ;1K
C876                     .BYTE $00,$00,$00
C879                     .BYTE $00,$00,$00
C87C                     .BYTE $00,$00,$00
C87F                     .BYTE $00,$00,$00                 ;5K
C882                     .BYTE $EA,$AC                     ;H D0
C884                     .BYTE $EA,$AC
C886                     .BYTE $EA,$AC
C888                     .BYTE $EA,$AC                     ;H D1
C88A                     .BYTE $EA,$AC
C88C                     .BYTE $EA,$AC
C88E                     .BYTE $AF,$80                     ;H D2
C890                     .BYTE $AA,$80
C892                     .BYTE $AA,$80
C894                     .BYTE $0B,$E8                     ;H D3
C896                     .BYTE $0A,$A8
C898                     .BYTE $0A,$A8
C89A                     .BYTE $00,$80                     ;S D0 S0
C89C                     .BYTE $00,$02
C89E                     .BYTE $A0,$00
C8A0                     .BYTE $02,$20
C8A2                     .BYTE $00,$28
C8A4                     .BYTE $0A,$00
C8A6                     .BYTE $20,$02
C8A8                     .BYTE $00,$20
C8AA                     .BYTE $02,$00                     ;Q D0 S0
```

```
C8AC                    .BYTE $0A,$80
C8AE                    .BYTE $0A,$80
C8B0                    .BYTE $0A,$80
C8B2                    .BYTE $0A,$80
C8B4                    .BYTE $0A,$80
C8B6                    .BYTE $0A,$80
C8B8                    .BYTE $0A,$A0
C8BA                    .BYTE $0A,$A0                               ;E D0 S0
C8BC                    .BYTE $0A,$A0
C8BE                    .BYTE $2A,$80                               ;T D0 S0
C8C0                    .BYTE $2A,$80
C8C2                    .BYTE $2A,$80
C8C4                    .BYTE $00,$01
C8C6                    .BYTE $40,$00                               ;B D0
C8C8                    .BYTE $40,$C0
C8CA                    .BYTE $8C,$C8
C8CC                    .BYTE $8C,$C8                               ;B D1
C8CE                    .BYTE $8C,$00
C8D0                    .BYTE $00,$00
C8D2                    .BYTE $00,$00                               ;B D2
C8D4                    .BYTE $08,$A0
C8D6                    .BYTE $A0,$0A
C8D8                    .BYTE $80,$01                               ;B D3
C8DA                    .BYTE $80,$0B
C8DC                    .BYTE $80,$03
C8DE                    .BYTE $08                                   ;MCSD0
C8DF                    .BYTE $00                                   ;D5
C8E0                    .BYTE $00                                   ;D6
C8E1                    .BYTE $6A                                   ;D7
C8E2                    .BYTE $95                                   ;DD
C8E3                    .BYTE $56,$A9                               ;G,BEX0
C8E5                    .BYTE $00,$00                               ;G,BEX1
C8E7                    .BYTE $00,$00                               ;G,BEX2
C8E9                    .BYTE $00,$00                               ;G,BEX3
C8EB                    .BYTE $00,$00                               ;G,BEX4
C8ED                    .BYTE $FF,$FF                               ;G,BEX5
C8EF                    .BYTE $FF,$FF                               ;G,BEX6
C8F1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  6
          ORG     STAMPS+$900

C900                    .BYTE $95,$80                               ;MC D0 S0
C902                    .BYTE $95,$80
C904                    .BYTE $95,$80
C906                    .BYTE $99,$80                               ;MC D1
C908                    .BYTE $99,$80
C90A                    .BYTE $99,$80
C90C                    .BYTE $19,$00                               ;MC D2
C90E                    .BYTE $A5,$00
C910                    .BYTE $6A,$00
C912                    .BYTE $19,$00                               ;MC D3
C914                    .BYTE $0A,$90
C916                    .BYTE $05,$A0
C918                    .BYTE $00,$00
C91A                    .BYTE $00,$00
C91C                    .BYTE $00,$00
C91E                    .BYTE $00,$00
C920                    .BYTE $00,$00
C922                    .BYTE $F5,$7C                               ;G D0
C924                    .BYTE $F5,$7C
C926                    .BYTE $F5,$7C
C928                    .BYTE $69,$C0                               ;MO D0
C92A                    .BYTE $6B,$C0
C92C                    .BYTE $29,$00
C92E                    .BYTE $DA,$40                               ;MO D1
C930                    .BYTE $FA,$40
C932                    .BYTE $1A,$00
C934                    .BYTE $2A,$00                               ;MO D2
C936                    .BYTE $2A,$00
C938                    .BYTE $2A,$00
C93A                    .BYTE $A8,$00                               ;MO D3
```

```
C93C                     .BYTE $A8,$00
C93E                     .BYTE $A8,$00
C940                     .BYTE $15,$50                              ;D D0
C942                     .BYTE $15,$50
C944                     .BYTE $15,$70
C946                     .BYTE $5D,$40                              ;D D1
C948                     .BYTE $9D,$40
C94A                     .BYTE $5D,$40
C94C                     .BYTE $64,$00                              ;D D2
C94E                     .BYTE $95,$80
C950                     .BYTE $16,$80
C952                     .BYTE $19,$00                              ;D D3
C954                     .BYTE $25,$60
C956                     .BYTE $29,$40
C958                     .BYTE $6A,$40                              ;MID0
C95A                     .BYTE $6A,$40
C95C                     .BYTE $6A,$40
C95E                     .BYTE $6A,$40                              ;MID1
C960                     .BYTE $6A,$40
C962                     .BYTE $6A,$40
C964                     .BYTE $98,$00                              ;MID2
C966                     .BYTE $6A,$40
C968                     .BYTE $69,$40
C96A                     .BYTE $98,$00                              ;MID3
C96C                     .BYTE $6A,$40
C96E                     .BYTE $5A,$40
C970                     .BYTE $51,$51,$40                          ;SK
C973                     .BYTE $00,$00,$00                          ;1K
C976                     .BYTE $00,$00,$00
C979                     .BYTE $00,$00,$00
C97C                     .BYTE $00,$00,$00
C97F                     .BYTE $00,$00,$00                          ;5K
C982                     .BYTE $EA,$AC                              ;H D0
C984                     .BYTE $EA,$AC
C986                     .BYTE $EA,$AC
C988                     .BYTE $EA,$AC                              ;H D1
C98A                     .BYTE $EA,$AC
C98C                     .BYTE $EA,$AC
C98E                     .BYTE $AE,$80                              ;H D2
C990                     .BYTE $AA,$80
C992                     .BYTE $BA,$80
C994                     .BYTE $0A,$E8                              ;H D3
C996                     .BYTE $0A,$B8
C998                     .BYTE $0A,$A8
C99A                     .BYTE $00,$80                              ;S D0 S0
C99C                     .BYTE $00,$02
C99E                     .BYTE $20,$00
C9A0                     .BYTE $08,$08
C9A2                     .BYTE $00,$28
C9A4                     .BYTE $0A,$00
C9A6                     .BYTE $A0,$02
C9A8                     .BYTE $80,$20
C9AA                     .BYTE $02,$00                              ;Q D0 S0
C9AC                     .BYTE $2A,$A0
C9AE                     .BYTE $2A,$A0
C9B0                     .BYTE $2A,$A0
C9B2                     .BYTE $2A,$A0
C9B4                     .BYTE $2A,$A0
C9B6                     .BYTE $2A,$A0
C9B8                     .BYTE $1A,$A0
C9BA                     .BYTE $1A,$A0                              ;E D0 S0
C9BC                     .BYTE $1A,$A0
C9BE                     .BYTE $2A,$90                              ;T D0 S0
C9C0                     .BYTE $2A,$90
C9C2                     .BYTE $2A,$90
C9C4                     .BYTE $00,$01
C9C6                     .BYTE $40,$00                              ;B D0
C9C8                     .BYTE $40,$E4
C9CA                     .BYTE $6C,$E0
C9CC                     .BYTE $2C,$C0                              ;B D1
C9CE                     .BYTE $2C,$80
C9D0                     .BYTE $20,$03
C9D2                     .BYTE $08,$00                              ;B D2
```

```
C9D4                    .BYTE $00,$0A
C9D6                    .BYTE $A0,$0A
C9D8                    .BYTE $80,$0A                              ;B D3
C9DA                    .BYTE $00,$0A
C9DC                    .BYTE $40,$00
C9DE                    .BYTE $00                                  ;MCSD0
C9DF                    .BYTE $00                                  ;D5
C9E0                    .BYTE $00                                  ;D6
C9E1                    .BYTE $6A                                  ;D7
C9E2                    .BYTE $A5                                  ;DD
C9E3                    .BYTE $5A,$A9                              ;G,BEX0
C9E5                    .BYTE $08,$00                              ;G,BEX1
C9E7                    .BYTE $00,$00                              ;G,BEX2
C9E9                    .BYTE $0C,$00                              ;G,BEX3
C9EB                    .BYTE $00,$00                              ;G,BEX4
C9ED                    .BYTE $FF,$FF                              ;G,BEX5
C9EF                    .BYTE $FF,$FF                              ;G,BEX6
C9F1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  5
          ORG     STAMPS+$A00

CA00                    .BYTE $95,$80                              ;MC D0 S0
CA02                    .BYTE $95,$80
CA04                    .BYTE $95,$80
CA06                    .BYTE $99,$80                              ;MC D1
CA08                    .BYTE $99,$80
CA0A                    .BYTE $99,$80
CA0C                    .BYTE $19,$00                              ;MC D2
CA0E                    .BYTE $15,$00
CA10                    .BYTE $64,$00
CA12                    .BYTE $19,$00                              ;MC D3
CA14                    .BYTE $01,$90
CA16                    .BYTE $05,$40
CA18                    .BYTE $03,$77
CA1A                    .BYTE $00,$00
CA1C                    .BYTE $00,$37
CA1E                    .BYTE $70,$00
CA20                    .BYTE $DD,$C0
CA22                    .BYTE $D6,$5C                              ;G D0
CA24                    .BYTE $D6,$5C
CA26                    .BYTE $D6,$5C
CA28                    .BYTE $69,$00                              ;MO D0
CA2A                    .BYTE $69,$C0
CA2C                    .BYTE $69,$00
CA2E                    .BYTE $1A,$40                              ;MO D1
CA30                    .BYTE $DA,$40
CA32                    .BYTE $1A,$40
CA34                    .BYTE $2A,$00                              ;MO D2
CA36                    .BYTE $2A,$00
CA38                    .BYTE $2A,$00
CA3A                    .BYTE $A8,$00                              ;MO D3
CA3C                    .BYTE $A8,$00
CA3E                    .BYTE $A8,$00
CA40                    .BYTE $15,$50                              ;D D0
CA42                    .BYTE $15,$50
CA44                    .BYTE $15,$50
CA46                    .BYTE $5D,$40                              ;D D1
CA48                    .BYTE $5D,$40
CA4A                    .BYTE $5D,$40
CA4C                    .BYTE $64,$00                              ;D D2
CA4E                    .BYTE $95,$00
CA50                    .BYTE $16,$00
CA52                    .BYTE $19,$00                              ;D D3
CA54                    .BYTE $05,$60
CA56                    .BYTE $09,$40
CA58                    .BYTE $6A,$40                              ;MID0
CA5A                    .BYTE $6A,$40
CA5C                    .BYTE $6A,$40
CA5E                    .BYTE $6A,$40                              ;MID1
CA60                    .BYTE $6A,$40
CA62                    .BYTE $6A,$40
```

```
CA64                    .BYTE $98,$00                       ;MID2
CA66                    .BYTE $5A,$00
CA68                    .BYTE $69,$00
CA6A                    .BYTE $98,$00                       ;MID3
CA6C                    .BYTE $29,$40
CA6E                    .BYTE $1A,$40
CA70                    .BYTE $11,$11,$00                   ;SK
CA73                    .BYTE $00,$00,$00                   ;1K
CA76                    .BYTE $00,$00,$00
CA79                    .BYTE $00,$00,$00
CA7C                    .BYTE $00,$00,$00
CA7F                    .BYTE $00,$00,$00                   ;5K
CA82                    .BYTE $EA,$AC                       ;H D0
CA84                    .BYTE $EA,$AC
CA86                    .BYTE $EA,$AC
CA88                    .BYTE $EA,$AC                       ;H D1
CA8A                    .BYTE $EA,$AC
CA8C                    .BYTE $EA,$AC
CA8E                    .BYTE $AE,$80                       ;H D2
CA90                    .BYTE $AA,$C0
CA92                    .BYTE $EA,$80
CA94                    .BYTE $0A,$E8                       ;H D3
CA96                    .BYTE $0A,$AC
CA98                    .BYTE $0E,$A8
CA9A                    .BYTE $00,$00                       ;S D0 S0
CA9C                    .BYTE $00,$02
CA9E                    .BYTE $A0,$00
CAA0                    .BYTE $02,$20
CAA2                    .BYTE $00,$28
CAA4                    .BYTE $0A,$00
CAA6                    .BYTE $20,$02
CAA8                    .BYTE $00,$20
CAAA                    .BYTE $02,$00                       ;Q D0 S0
CAAC                    .BYTE $A6,$98
CAAE                    .BYTE $A6,$98
CAB0                    .BYTE $A6,$98
CAB2                    .BYTE $96,$58
CAB4                    .BYTE $96,$58
CAB6                    .BYTE $96,$58
CAB8                    .BYTE $9A,$58
CABA                    .BYTE $9A,$58                       ;E D0 S0
CABC                    .BYTE $9A,$58
CABE                    .BYTE $16,$98                       ;T D0 S0
CAC0                    .BYTE $16,$98
CAC2                    .BYTE $16,$98
CAC4                    .BYTE $00,$04
CAC6                    .BYTE $10,$00                       ;B D0
CAC8                    .BYTE $40,$D4
CACA                    .BYTE $1C,$D0
CACC                    .BYTE $0C,$E0                       ;B D1
CACE                    .BYTE $0C,$00
CAD0                    .BYTE $00,$00
CAD2                    .BYTE $00,$00                       ;B D2
CAD4                    .BYTE $00,$12
CAD6                    .BYTE $50,$02
CAD8                    .BYTE $40,$0B                       ;B D3
CADA                    .BYTE $00,$0A
CADC                    .BYTE $40,$02
CADE                    .BYTE $00                           ;MCSD0
CADF                    .BYTE $00                           ;D5
CAE0                    .BYTE $00                           ;D6
CAE1                    .BYTE $66                           ;D7
CAE2                    .BYTE $99                           ;DD
CAE3                    .BYTE $66,$99                       ;G,BEX0
CAE5                    .BYTE $08,$00                       ;G,BEX1
CAE7                    .BYTE $80,$C0                       ;G,BEX2
CAE9                    .BYTE $0C,$00                       ;G,BEX3
CAEB                    .BYTE $0C,$80                       ;G,BEX4
CAED                    .BYTE $FF,$FF                       ;G,BEX5
CAEF                    .BYTE $FF,$FF                       ;G,BEX6
CAF1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
```

```
****  LINE  4
          ORG     STAMPS+$B00

CB00                     .BYTE $15,$00                              ;MC D0 S0
CB02                     .BYTE $15,$00
CB04                     .BYTE $15,$00
CB06                     .BYTE $15,$00                              ;MC D1
CB08                     .BYTE $15,$00
CB0A                     .BYTE $15,$00
CB0C                     .BYTE $2A,$00                              ;MC D2
CB0E                     .BYTE $0A,$80
CB10                     .BYTE $2A,$00
CB12                     .BYTE $2A,$00                              ;MC D3
CB14                     .BYTE $0A,$80
CB16                     .BYTE $2A,$00
CB18                     .BYTE $00,$00
CB1A                     .BYTE $00,$00
CB1C                     .BYTE $00,$00
CB1E                     .BYTE $00,$00
CB20                     .BYTE $00,$00
CB22                     .BYTE $DA,$9C                              ;G D0
CB24                     .BYTE $DA,$9C
CB26                     .BYTE $DA,$9C
CB28                     .BYTE $AA,$00                              ;MO D0
CB2A                     .BYTE $A9,$00
CB2C                     .BYTE $6A,$00
CB2E                     .BYTE $2A,$80                              ;MO D1
CB30                     .BYTE $2A,$80
CB32                     .BYTE $2A,$80
CB34                     .BYTE $28,$00                              ;MO D2
CB36                     .BYTE $28,$00
CB38                     .BYTE $28,$00
CB3A                     .BYTE $28,$00                              ;MO D3
CB3C                     .BYTE $28,$00
CB3E                     .BYTE $28,$00
CB40                     .BYTE $15,$50                              ;D D0
CB42                     .BYTE $15,$50
CB44                     .BYTE $15,$50
CB46                     .BYTE $5D,$40                              ;D D1
CB48                     .BYTE $5D,$40
CB4A                     .BYTE $5D,$40
CB4C                     .BYTE $64,$00                              ;D D2
CB4E                     .BYTE $25,$00
CB50                     .BYTE $01,$00
CB52                     .BYTE $19,$00                              ;D D3
CB54                     .BYTE $05,$80
CB56                     .BYTE $05,$40
CB58                     .BYTE $6A,$40                              ;MID0
CB5A                     .BYTE $2A,$40
CB5C                     .BYTE $6A,$00
CB5E                     .BYTE $6A,$40                              ;MID1
CB60                     .BYTE $6A,$00
CB62                     .BYTE $2A,$40
CB64                     .BYTE $20,$00                              ;MID2
CB66                     .BYTE $5A,$00
CB68                     .BYTE $6A,$00
CB6A                     .BYTE $20,$00                              ;MID3
CB6C                     .BYTE $29,$40
CB6E                     .BYTE $2A,$40
CB70                     .BYTE $0F,$7C,$00                          ;SK
CB73                     .BYTE $FE,$AA,$A0                          ;1K
CB76                     .BYTE $FE,$AA,$A0
CB79                     .BYTE $FE,$AA,$A0
CB7C                     .BYTE $0E,$AA,$A0
CB7F                     .BYTE $FE,$AA,$A0                          ;5K
CB82                     .BYTE $EA,$AC                              ;H D0
CB84                     .BYTE $EA,$AC
CB86                     .BYTE $EA,$AC
CB88                     .BYTE $EA,$AC                              ;H D1
CB8A                     .BYTE $EA,$AC
CB8C                     .BYTE $EA,$AC
CB8E                     .BYTE $AE,$80                              ;H D2
CB90                     .BYTE $AB,$B0
```

```
CB92                    .BYTE $BA,$B0
CB94                    .BYTE $0A,$E8                               ;H D3
CB96                    .BYTE $3A,$B8
CB98                    .BYTE $3B,$A8
CB9A                    .BYTE $00,$00                               ;S D0 S0
CB9C                    .BYTE $00,$00
CB9E                    .BYTE $80,$00
CBA0                    .BYTE $02,$A0
CBA2                    .BYTE $00,$08
CBA4                    .BYTE $08,$00
CBA6                    .BYTE $20,$02
CBA8                    .BYTE $00,$00
CBAA                    .BYTE $00,$00                               ;Q D0 S0
CBAC                    .BYTE $9A,$68
CBAE                    .BYTE $9A,$68
CBB0                    .BYTE $9A,$68
CBB2                    .BYTE $AA,$A8
CBB4                    .BYTE $AA,$A8
CBB6                    .BYTE $AA,$A8
CBB8                    .BYTE $66,$A8
CBBA                    .BYTE $66,$A8                               ;E D0 S0
CBBC                    .BYTE $66,$A8
CBBE                    .BYTE $AA,$64                               ;T D0 S0
CBC0                    .BYTE $AA,$64
CBC2                    .BYTE $AA,$64
CBC4                    .BYTE $00,$04
CBC6                    .BYTE $10,$00                               ;B D0
CBC8                    .BYTE $40,$DA
CBCA                    .BYTE $9C,$D2
CBCC                    .BYTE $18,$C2                               ;B D1
CBCE                    .BYTE $18,$08
CBD0                    .BYTE $08,$02
CBD2                    .BYTE $00,$00                               ;B D2
CBD4                    .BYTE $00,$AA
CBD6                    .BYTE $A8,$2A
CBD8                    .BYTE $A0,$08                               ;B D3
CBDA                    .BYTE $80,$00
CBDC                    .BYTE $00,$00
CBDE                    .BYTE $00                                   ;MCSD0
CBDF                    .BYTE $00                                   ;D5
CBE0                    .BYTE $00                                   ;D6
CBE1                    .BYTE $AA                                   ;D7
CBE2                    .BYTE $65                                   ;DD
CBE3                    .BYTE $59,$AA                               ;G,BEX0
CBE5                    .BYTE $08,$00                               ;G,BEX1
CBE7                    .BYTE $23,$00                               ;G,BEX2
CBE9                    .BYTE $0C,$00                               ;G,BEX3
CBEB                    .BYTE $32,$00                               ;G,BEX4
CBED                    .BYTE $FF,$FF                               ;G,BEX5
CBEF                    .BYTE $FF,$FF                               ;G,BEX6
CBF1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  3
          ORG     STAMPS+$C00

CC00                    .BYTE $2A,$00                               ;MC D0 S0
CC02                    .BYTE $2A,$00
CC04                    .BYTE $2A,$00
CC06                    .BYTE $3F,$00                               ;MC D1
CC08                    .BYTE $3F,$00
CC0A                    .BYTE $3F,$00
CC0C                    .BYTE $EA,$C0                               ;MC D2
CC0E                    .BYTE $3A,$B0
CC10                    .BYTE $EA,$C0
CC12                    .BYTE $EA,$C0                               ;MC D3
CC14                    .BYTE $3A,$B0
CC16                    .BYTE $EA,$C0
CC18                    .BYTE $0F,$C0                               ;G D0
CC1A                    .BYTE $00,$00
CC1C                    .BYTE $00,$00
CC1E                    .BYTE $FC,$00
CC20                    .BYTE $3F,$00
```

```
CC22                    .BYTE $00,$00
CC24                    .BYTE $00,$00
CC26                    .BYTE $00,$00
CC28                    .BYTE $55,$00                              ;MO D0
CC2A                    .BYTE $55,$00
CC2C                    .BYTE $55,$00
CC2E                    .BYTE $15,$40                              ;MO D1
CC30                    .BYTE $15,$40
CC32                    .BYTE $15,$40
CC34                    .BYTE $60,$00                              ;MO D2
CC36                    .BYTE $60,$00
CC38                    .BYTE $60,$00
CC3A                    .BYTE $09,$00                              ;MO D3
CC3C                    .BYTE $09,$00
CC3E                    .BYTE $09,$00
CC40                    .BYTE $0F,$C0                              ;D D0
CC42                    .BYTE $0F,$C0
CC44                    .BYTE $0F,$C0
CC46                    .BYTE $3F,$00                              ;D D1
CC48                    .BYTE $3F,$00
CC4A                    .BYTE $3F,$00
CC4C                    .BYTE $FC,$00                              ;D D2
CC4E                    .BYTE $3F,$00
CC50                    .BYTE $3F,$00
CC52                    .BYTE $3F,$00                              ;D D3
CC54                    .BYTE $0F,$C0
CC56                    .BYTE $0F,$C0
CC58                    .BYTE $2A,$00                              ;MID0
CC5A                    .BYTE $2A,$00
CC5C                    .BYTE $2A,$00
CC5E                    .BYTE $2A,$00                              ;MID1
CC60                    .BYTE $2A,$00
CC62                    .BYTE $2A,$00
CC64                    .BYTE $A4,$00                              ;MID2
CC66                    .BYTE $29,$00
CC68                    .BYTE $29,$00
CC6A                    .BYTE $68,$00                              ;MID3
CC6C                    .BYTE $1A,$00
CC6E                    .BYTE $1A,$00
CC70                    .BYTE $05,$54,$00                          ;SK
CC73                    .BYTE $32,$22,$20                          ;1K
CC76                    .BYTE $C2,$22,$20
CC79                    .BYTE $0E,$22,$20
CC7C                    .BYTE $0E,$22,$20
CC7F                    .BYTE $0E,$22,$20                          ;5K
CC82                    .BYTE $EA,$AC                              ;H D0
CC84                    .BYTE $EA,$AC
CC86                    .BYTE $EA,$AC
CC88                    .BYTE $EA,$AC                              ;H D1
CC8A                    .BYTE $EA,$AC
CC8C                    .BYTE $EA,$AC
CC8E                    .BYTE $AE,$80                              ;H D2
CC90                    .BYTE $AE,$80
CC92                    .BYTE $AE,$80
CC94                    .BYTE $0A,$E8                              ;H D3
CC96                    .BYTE $0A,$E8
CC98                    .BYTE $0A,$E8
CC9A                    .BYTE $00,$00                              ;S D0 S0
CC9C                    .BYTE $00,$00
CC9E                    .BYTE $00,$00
CCA0                    .BYTE $00,$80
CCA2                    .BYTE $00,$0A
CCA4                    .BYTE $A8,$00
CCA6                    .BYTE $00,$00
CCA8                    .BYTE $00,$00
CCAA                    .BYTE $00,$00                              ;Q D0 S0
CCAC                    .BYTE $5A,$64
CCAE                    .BYTE $5A,$64
CCB0                    .BYTE $5A,$64
CCB2                    .BYTE $66,$94
CCB4                    .BYTE $66,$94
CCB6                    .BYTE $66,$94
CCB8                    .BYTE $96,$58
```

```
CCBA                    .BYTE $96,$58                             ;E D0 S0
CCBC                    .BYTE $96,$58
CCBE                    .BYTE $96,$58                             ;T D0 S0
CCC0                    .BYTE $96,$58
CCC2                    .BYTE $96,$58
CCC4                    .BYTE $00,$10
CCC6                    .BYTE $04,$00                             ;B D0
CCC8                    .BYTE $40,$00
CCCA                    .BYTE $00,$00
CCCC                    .BYTE $00,$00                             ;B D1
CCCE                    .BYTE $00,$00
CCD0                    .BYTE $00,$00
CCD2                    .BYTE $00,$00                             ;B D2
CCD4                    .BYTE $00,$26
CCD6                    .BYTE $94,$06
CCD8                    .BYTE $90,$00                             ;B D3
CCDA                    .BYTE $00,$00
CCDC                    .BYTE $00,$20
CCDE                    .BYTE $00                                 ;MCSD0
CCDF                    .BYTE $02                                 ;D5
CCE0                    .BYTE $00                                 ;D6
CCE1                    .BYTE $96                                 ;D7
CCE2                    .BYTE $59                                 ;DD
CCE3                    .BYTE $65,$96                             ;G,BEX0
CCE5                    .BYTE $FF,$C0                             ;G,BEX1
CCE7                    .BYTE $0C,$00                             ;G,BEX2
CCE9                    .BYTE $AE,$80                             ;G,BEX3
CCEB                    .BYTE $0C,$00                             ;G,BEX4
CCED                    .BYTE $FF,$FF                             ;G,BEX5
CCEF                    .BYTE $FF,$FF                             ;G,BEX6
CCF1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  2
          ORG     STAMPS+$D00

CD00                    .BYTE $EA,$C0                             ;MC D0 S0
CD02                    .BYTE $EA,$C0
CD04                    .BYTE $EA,$C0
CD06                    .BYTE $CC,$C0                             ;MC D1
CD08                    .BYTE $CC,$C0
CD0A                    .BYTE $CC,$C0
CD0C                    .BYTE $FF,$C0                             ;MC D2
CD0E                    .BYTE $3F,$F0
CD10                    .BYTE $FF,$C0
CD12                    .BYTE $FF,$C0                             ;MC D3
CD14                    .BYTE $3F,$F0
CD16                    .BYTE $FF,$C0
CD18                    .BYTE $00,$00
CD1A                    .BYTE $00,$00
CD1C                    .BYTE $00,$00
CD1E                    .BYTE $00,$00
CD20                    .BYTE $00,$00
CD22                    .BYTE $3F,$F0                             ;G D0
CD24                    .BYTE $3F,$F0
CD26                    .BYTE $3F,$F0
CD28                    .BYTE $55,$00                             ;MO D0
CD2A                    .BYTE $55,$00
CD2C                    .BYTE $55,$00
CD2E                    .BYTE $19,$40                             ;MO D1
CD30                    .BYTE $19,$40
CD32                    .BYTE $19,$40
CD34                    .BYTE $54,$00                             ;MO D2
CD36                    .BYTE $54,$00
CD38                    .BYTE $54,$00
CD3A                    .BYTE $15,$00                             ;MO D3
CD3C                    .BYTE $15,$00
CD3E                    .BYTE $15,$00
CD40                    .BYTE $0A,$80                             ;D D0
CD42                    .BYTE $0A,$80
CD44                    .BYTE $0A,$80
CD46                    .BYTE $3F,$00                             ;D D1
CD48                    .BYTE $3F,$00
```

```
CD4A                    .BYTE $3F,$00
CD4C                    .BYTE $BC,$00                              ;D D2
CD4E                    .BYTE $2F,$00
CD50                    .BYTE $2F,$00
CD52                    .BYTE $3E,$00                              ;D D3
CD54                    .BYTE $0F,$80
CD56                    .BYTE $0F,$80
CD58                    .BYTE $2A,$00                              ;MID0
CD5A                    .BYTE $2A,$00
CD5C                    .BYTE $2A,$00
CD5E                    .BYTE $19,$00                              ;MID1
CD60                    .BYTE $19,$00
CD62                    .BYTE $19,$00
CD64                    .BYTE $A4,$00                              ;MID2
CD66                    .BYTE $29,$00
CD68                    .BYTE $29,$00
CD6A                    .BYTE $68,$00                              ;MID3
CD6C                    .BYTE $1A,$00
CD6E                    .BYTE $1A,$00
CD70                    .BYTE $05,$54,$00                          ;SK
CD73                    .BYTE $32,$22,$20                          ;1K
CD76                    .BYTE $FE,$22,$20
CD79                    .BYTE $3E,$22,$20
CD7C                    .BYTE $FE,$22,$20
CD7F                    .BYTE $FE,$22,$20                          ;5K
CD82                    .BYTE $EA,$AC                              ;H D0
CD84                    .BYTE $EA,$AC
CD86                    .BYTE $EA,$AC
CD88                    .BYTE $EA,$AC                              ;H D1
CD8A                    .BYTE $EA,$AC
CD8C                    .BYTE $EA,$AC
CD8E                    .BYTE $AA,$80                              ;H D2
CD90                    .BYTE $AA,$80
CD92                    .BYTE $AA,$80
CD94                    .BYTE $0A,$A8                              ;H D3
CD96                    .BYTE $0A,$A8
CD98                    .BYTE $0A,$A8
CD9A                    .BYTE $00,$00                              ;S D0 S0
CD9C                    .BYTE $00,$00
CD9E                    .BYTE $00,$00
CDA0                    .BYTE $00,$00
CDA2                    .BYTE $00,$02
CDA4                    .BYTE $A0,$00
CDA6                    .BYTE $00,$00
CDA8                    .BYTE $00,$00
CDAA                    .BYTE $00,$00                              ;Q D0 S0
CDAC                    .BYTE $95,$94
CDAE                    .BYTE $95,$94
CDB0                    .BYTE $95,$94
CDB2                    .BYTE $59,$58
CDB4                    .BYTE $59,$58
CDB6                    .BYTE $59,$58
CDB8                    .BYTE $65,$64
CDBA                    .BYTE $65,$64                              ;E D0 S0
CDBC                    .BYTE $65,$64
CDBE                    .BYTE $65,$64                              ;T D0 S0
CDC0                    .BYTE $65,$64
CDC2                    .BYTE $65,$64
CDC4                    .BYTE $00,$10
CDC6                    .BYTE $04,$00                              ;B D0
CDC8                    .BYTE $40,$3C
CDCA                    .BYTE $F0,$30
CDCC                    .BYTE $30,$30                              ;B D1
CDCE                    .BYTE $00,$00
CDD0                    .BYTE $00,$20
CDD2                    .BYTE $00,$02                              ;B D2
CDD4                    .BYTE $00,$19
CDD6                    .BYTE $50,$09
CDD8                    .BYTE $40,$00                              ;B D3
CDDA                    .BYTE $00,$00
CDDC                    .BYTE $00,$00
CDDE                    .BYTE $80                                  ;MCSD0
CDDF                    .BYTE $00                                  ;D5
```

```
CDE0                    .BYTE $00                                   ;D6
CDE1                    .BYTE $66                                   ;D7
CDE2                    .BYTE $69                                   ;DD
CDE3                    .BYTE $69,$99                               ;G,BEX0
CDE5                    .BYTE $08,$00                               ;G,BEX1
CDE7                    .BYTE $32,$00                               ;G,BEX2
CDE9                    .BYTE $0C,$00                               ;G,BEX3
CDEB                    .BYTE $23,$00                               ;G,BEX4
CDED                    .BYTE $FF,$FF                               ;G,BEX5
CDEF                    .BYTE $FF,$FF                               ;G,BEX6
CDF1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

**********************************************************
****  LINE  1
          ORG     STAMPS+$E00

CE00                    .BYTE $EA,$C0                               ;MC D0 S0
CE02                    .BYTE $EA,$C0
CE04                    .BYTE $EA,$C0
CE06                    .BYTE $FF,$C0                               ;MC D1
CE08                    .BYTE $FF,$C0
CE0A                    .BYTE $FF,$C0
CE0C                    .BYTE $AA,$C0                               ;MC D2
CE0E                    .BYTE $2A,$B0
CE10                    .BYTE $AA,$C0
CE12                    .BYTE $EA,$80                               ;MC D3
CE14                    .BYTE $3A,$A0
CE16                    .BYTE $EA,$80
CE18                    .BYTE $FF,$C0
CE1A                    .BYTE $00,$00
CE1C                    .BYTE $00,$00
CE1E                    .BYTE $FF,$C0
CE20                    .BYTE $FF,$C0
CE22                    .BYTE $3F,$F0                               ;G D0
CE24                    .BYTE $3F,$F0
CE26                    .BYTE $3F,$F0
CE28                    .BYTE $14,$00                               ;MO D0
CE2A                    .BYTE $14,$00
CE2C                    .BYTE $14,$00
CE2E                    .BYTE $1A,$40                               ;MO D1
CE30                    .BYTE $1A,$40
CE32                    .BYTE $1A,$40
CE34                    .BYTE $18,$00                               ;MO D2
CE36                    .BYTE $18,$00
CE38                    .BYTE $18,$00
CE3A                    .BYTE $24,$00                               ;MO D3
CE3C                    .BYTE $24,$00
CE3E                    .BYTE $24,$00
CE40                    .BYTE $0A,$80                               ;D D0
CE42                    .BYTE $0A,$80
CE44                    .BYTE $0A,$80
CE46                    .BYTE $0C,$00                               ;D D1
CE48                    .BYTE $0C,$00
CE4A                    .BYTE $0C,$00
CE4C                    .BYTE $B0,$00                               ;D D2
CE4E                    .BYTE $2C,$00
CE50                    .BYTE $2C,$00
CE52                    .BYTE $0E,$00                               ;D D3
CE54                    .BYTE $03,$80
CE56                    .BYTE $03,$80
CE58                    .BYTE $2A,$00                               ;MID0
CE5A                    .BYTE $2A,$00
CE5C                    .BYTE $2A,$00
CE5E                    .BYTE $2A,$00                               ;MID1
CE60                    .BYTE $2A,$00
CE62                    .BYTE $2A,$00
CE64                    .BYTE $A8,$00                               ;MID2
CE66                    .BYTE $2A,$00
CE68                    .BYTE $2A,$00
CE6A                    .BYTE $A8,$00                               ;MID3
CE6C                    .BYTE $2A,$00
CE6E                    .BYTE $2A,$00
CE70                    .BYTE $05,$54,$00                           ;SK
```

```
CE73                    .BYTE $32,$22,$20                           ;1K
CE76                    .BYTE $0E,$22,$20
CE79                    .BYTE $0E,$22,$20
CE7C                    .BYTE $CE,$22,$20
CE7F                    .BYTE $C2,$22,$20                           ;5K
CE82                    .BYTE $01,$00                               ;H D0
CE84                    .BYTE $01,$00
CE86                    .BYTE $01,$00
CE88                    .BYTE $01,$00                               ;H D1
CE8A                    .BYTE $01,$00
CE8C                    .BYTE $01,$00
CE8E                    .BYTE $04,$00                               ;H D2
CE90                    .BYTE $04,$00
CE92                    .BYTE $04,$00
CE94                    .BYTE $00,$40                               ;H D3
CE96                    .BYTE $00,$40
CE98                    .BYTE $00,$40
CE9A                    .BYTE $00,$00                               ;S D0 S0
CE9C                    .BYTE $00,$00
CE9E                    .BYTE $00,$00
CEA0                    .BYTE $00,$00
CEA2                    .BYTE $00,$00
CEA4                    .BYTE $80,$00
CEA6                    .BYTE $02,$A0
CEA8                    .BYTE $00,$00
CEAA                    .BYTE $80,$00                               ;Q D0 S0
CEAC                    .BYTE $66,$68
CEAE                    .BYTE $66,$68
CEB0                    .BYTE $66,$68
CEB2                    .BYTE $A6,$64
CEB4                    .BYTE $A6,$64
CEB6                    .BYTE $A6,$64
CEB8                    .BYTE $A9,$A0
CEBA                    .BYTE $A9,$A0                               ;E D0 S0
CEBC                    .BYTE $A9,$A0
CEBE                    .BYTE $29,$A8                               ;T D0 S0
CEC0                    .BYTE $29,$A8
CEC2                    .BYTE $29,$A8
CEC4                    .BYTE $00,$40
CEC6                    .BYTE $01,$00                               ;B D0
CEC8                    .BYTE $40,$3F
CECA                    .BYTE $F0,$3C
CECC                    .BYTE $F0,$20                               ;B D1
CECE                    .BYTE $30,$20
CED0                    .BYTE $20,$00
CED2                    .BYTE $80,$00                               ;B D2
CED4                    .BYTE $00,$22
CED6                    .BYTE $60,$02
CED8                    .BYTE $00,$00                               ;B D3
CEDA                    .BYTE $00,$00
CEDC                    .BYTE $00,$00
CEDE                    .BYTE $00                                   ;MCSD0
CEDF                    .BYTE $00                                   ;D5
CEE0                    .BYTE $00                                   ;D6
CEE1                    .BYTE $69                                   ;D7
CEE2                    .BYTE $A5                                   ;DD
CEE3                    .BYTE $5A,$69                               ;G,BEX0
CEE5                    .BYTE $08,$00                               ;G,BEX1
CEE7                    .BYTE $C0,$80                               ;G,BEX2
CEE9                    .BYTE $C0,$00                               ;G,BEX3
CEEB                    .BYTE $80,$C0                               ;G,BEX4
CEED                    .BYTE $FF,$FF                               ;G,BEX5
CEEF                    .BYTE $FF,$FF                               ;G,BEX6
CEF1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

*********************************************************
****  LINE  0
          ORG     STAMPBAS+$F00

MCD0S0:
CF00                    .BYTE $FB,$C0                               ;MC D0 S0
MCD0S1:
CF02                    .BYTE $FB,$C0
```

```
MCD0S2:
CF04                    .BYTE $FB,$C0
MCD1S0:
CF06                    .BYTE $2A,$00                              ;MC D1
MCD1S1:
CF08                    .BYTE $2A,$00
MCD1S2:
CF0A                    .BYTE $2A,$00
MCD2S0:
CF0C                    .BYTE $2A,$C0                              ;MC D2
MCD2S1:
CF0E                    .BYTE $0A,$B0
MCD2S2:
CF10                    .BYTE $2A,$C0
MCD3S0:
CF12                    .BYTE $EA,$00                              ;MC D3
MCD3S1:
CF14                    .BYTE $3A,$80
MCD3S2:
CF16                    .BYTE $EA,$00

CF18                    .BYTE $A8,$00
CF1A                    .BYTE $00,$00
CF1C                    .BYTE $00,$00
CF1E                    .BYTE $0A,$80
CF20                    .BYTE $2A,$00

GD0S0:
CF22                    .BYTE $0A,$80                              ;G D0
GD0S1:
CF24                    .BYTE $0A,$80
GD0S2:
CF26                    .BYTE $0A,$80
MOD0S0:
CF28                    .BYTE $14,$00                              ;MO D0
MOD0S1:
CF2A                    .BYTE $14,$00
MOD0S2:
CF2C                    .BYTE $14,$00
MOD1S0:
CF2E                    .BYTE $05,$00                              ;MO D1
MOD1S1:
CF30                    .BYTE $05,$00
MOD1S2:
CF32                    .BYTE $05,$00
MOD2S0:
CF34                    .BYTE $14,$00                              ;MO D2
MOD2S1:
CF36                    .BYTE $14,$00
MOD2S2:
CF38                    .BYTE $14,$00
MOD3S0:
CF3A                    .BYTE $14,$00                              ;MO D3
MOD3S1:
CF3C                    .BYTE $14,$00
MOD3S2:
CF3E                    .BYTE $14,$00
DD0S0:
CF40                    .BYTE $0A,$80                              ;D D0
DD0S1:
CF42                    .BYTE $0A,$80
DD0S2:
CF44                    .BYTE $0A,$80
DD1S0:
CF46                    .BYTE $2A,$00                              ;D D1
DD1S1:
CF48                    .BYTE $2A,$00
DD1S2:
CF4A                    .BYTE $2A,$00
DD2S0:
CF4C                    .BYTE $A8,$00                              ;D D2
DD2S1:
CF4E                    .BYTE $2A,$00
```

```
DD2S2:
CF50                    .BYTE $2A,$00
DD3S0:
CF52                    .BYTE $2A,$00                              ;D D3
DD3S1:
CF54                    .BYTE $0A,$80
DD3S2:
CF56                    .BYTE $0A,$80
MID0S0:
CF58                    .BYTE $2A,$00                              ;MI D0
MID0S1:
CF5A                    .BYTE $2A,$00
MID0S2:
CF5C                    .BYTE $2A,$00
MID1S0:
CF5E                    .BYTE $2A,$00                              ;MID1
MID1S1:
CF60                    .BYTE $2A,$00
MID1S2:
CF62                    .BYTE $2A,$00
MID2S0:
CF64                    .BYTE $A8,$00                              ;MID2
MID2S1:
CF66                    .BYTE $2A,$00
MID2S2:
CF68                    .BYTE $2A,$00
MID3S0:
CF6A                    .BYTE $A8,$00                              ;MID3
MID3S1:
CF6C                    .BYTE $2A,$00
MID3S2:
CF6E                    .BYTE $2A,$00
SKULL:
CF70                    .BYTE $01,$50,$00
SCORE1K:
CF73                    .BYTE $F2,$AA,$A0
SCORE2K:
CF76                    .BYTE $FE,$AA,$A0
SCORE3K:
CF79                    .BYTE $FE,$AA,$A0
SCORE4K:
CF7C                    .BYTE $C2,$AA,$A0
SCORE5K:
CF7F                    .BYTE $FE,$AA,$A0
HD0S0:
CF82                    .BYTE $05,$40                              ;H D0
HD0S1:
CF84                    .BYTE $05,$40
HD0S2:
CF86                    .BYTE $05,$40
HD1S0:
CF88                    .BYTE $05,$40                              ;H D1
HD1S1:
CF8A                    .BYTE $05,$40
HD1S2:
CF8C                    .BYTE $05,$40
HD2S0:
CF8E                    .BYTE $15,$00                              ;H D2
HD2S1:
CF90                    .BYTE $15,$00
HD2S2:
CF92                    .BYTE $15,$00
HD3S0:
CF94                    .BYTE $01,$50                              ;H D3
HD3S1:
CF96                    .BYTE $01,$50
HD3S2:
CF98                    .BYTE $01,$50
SD0S0:
CF9A                    .BYTE $00,$00                              ;S D0 S0
SD0S1:
CF9C                    .BYTE $00,$00
SD0S2:
```

```
CF9E                    .BYTE $00,$00
SD0S3:
CFA0                    .BYTE $00,$00
SD0S4:
CFA2                    .BYTE $00,$00
SD0S5:
CFA4                    .BYTE $00,$00
SD0S6:
CFA6                    .BYTE $00,$80
SD0S7:
CFA8                    .BYTE $00,$00
QD0S0:
CFAA                    .BYTE $00,$00                               ;Q D0 S0
QD0S1:
CFAC                    .BYTE $26,$50
QD0S2:
CFAE                    .BYTE $26,$50
QD0S3:
CFB0                    .BYTE $26,$50
QD0S4:
CFB2                    .BYTE $16,$60
QD0S5:
CFB4                    .BYTE $16,$60
QD0S6:
CFB6                    .BYTE $16,$60
QD0S7:
CFB8                    .BYTE $19,$40
ED0S0:
CFBA                    .BYTE $19,$40                               ;E D0 S0
ED0S1:
CFBC                    .BYTE $19,$40
TD0S0:
CFBE                    .BYTE $05,$90                               ;T D0 S0
TD0S1:
CFC0                    .BYTE $05,$90
TD0S2:
CFC2                    .BYTE $05,$90
TD0S3:
CFC4                    .BYTE $00,$40
BD0S0:
CFC6                    .BYTE $01,$55                               ;B D0
BD0S1:
CFC8                    .BYTE $40,$0A
BD0S2:
CFCA                    .BYTE $80,$0A
BD1S0:
CFCC                    .BYTE $80,$02                               ;B D1
BD1S1:
CFCE                    .BYTE $80,$00
BD1S2:
CFD0                    .BYTE $00,$00
BD2S0:
CFD2                    .BYTE $00,$00                               ;B D2
BD2S1:
CFD4                    .BYTE $00,$04
BD2S2:
CFD6                    .BYTE $00,$00
BD3S0:
CFD8                    .BYTE $00,$00                               ;B D3
BD3S1:
CFDA                    .BYTE $00,$00
BD3S2:
CFDC                    .BYTE $00,$00
MCSD0S0:
CFDE                    .BYTE $00                                   ;MCSD0
MCSD5S0:
CFDF                    .BYTE $00                                   ;D5
MCSD6S0:
CFE0                    .BYTE $00                                   ;D6
MCSD7S0:
CFE1                    .BYTE $56                                   ;D7
MCSDDS0:
CFE2                    .BYTE $95                                   ;DD
```

```
CFE3                    .BYTE $56,$95                               ;G,BEX0
GD0S3:
CFE5                    .BYTE $08,$00                               ;G,BEX1
GD0S4:
CFE7                    .BYTE $00,$00                               ;G,BEX2
GD0S5:
CFE9                    .BYTE $0C,$00                               ;G,BEX3
GD0S6:
CFEB                    .BYTE $00,$00                               ;G,BEX4
GD0S7:
CFED                    .BYTE $FF,$FF                               ;G,BEX5
GD0S8:
CFEF                    .BYTE $FF,$FF                               ;G,BEX6
CFF1                    .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

****************************************************************
*                                                              *
*        WAVEEND - DO SOMETHING INTERESTING BETWEEN WAVES      *
*                                                              *
****************************************************************
*
*         START WAVE END SOUND
WAVEEND:
D000     20 56 E3       JSR $E356
D003     A9 06          LDA #SRACKA                                 ;Play Rack End Sound
D005     20 95 E3       JSR DOTUNE_$E395
D008     20 F9 F6       JSR DISPINIT
D00B     20 86 DC       JSR $DC86
D00E     4C 00 B0       JMP $B000

****************************************************************
*******                                                  *******
*******  MCDEATH --  JUMP HERE WHEN MC GETS KNOCKED OFF  *******
*******                                                  *******
****************************************************************
*
*         START MC DEATH SOUND

MCDEATH:                ;MC HAS KICKED THE BUCKET
D011     20 56 E3       JSR $E356                                   ;TURN OFF SOUND
D014     A9 08          LDA #$08                                    ;PLAY MC DIE SOUND
D016     20 95 E3       JSR DOTUNE_$E395
D019     20 AA DE       JSR $DEAA
D01C     A2 00          LDX $00

MCD1:                                                               ;DO SOME FANCY COLOR CYCLING
D01E     86 21          STX P0C1
D020     86 22          STX P0C2
D022     86 23          STX P0C3
D024     A5 E5          LDA $E5
D026     C5 E7          CMP $E7
D028     F0 FA          BEQ $D024
D02A     85 E7          STA $E7
D02C     20 34 DB       JSR $DB34
D02F     20 5D D3       JSR $D35D
D032     E8             INX
D033     E8             INX
D034     D0 E8          BNE MCD1
D036     A2 0F          LDX $0F

MCD2:
D038     86 21          STX P0C1
D03A     86 22          STX P0C2
D03C     86 23          STX P0C3
D03E     A0 04          LDY $04

MCD3:
;Disassembly of $D040-$D0FF compliments of Dan Boris & "Scotty"
D040     A5 E5          LDA $E5
D042     C5 E7          CMP $E7
D044     F0 FA          BEQ $D040
D046     85 E7          STA $E7
D048     20 34 DB       JSR $DB34
```

```
D04B      20 5D D3       JSR $D35D
D04E      88             DEY
D04F      10 EF          BPL $D040
D051      CA             DEX
D052      10 E4          BPL $D038
D054      A6 61          LDX $61
D056      B5 E0          LDA $E0,X
D058      D0 33          BNE $D08D
D05A      A5 62          LDA CURPLAYERS
D05C      F0 2F          BEQ $D08D
D05E      8A             TXA
D05F      49 01          EOR #$01
D061      AA             TAX
D062      B5 E0          LDA $E0,X
D064      F0 27          BEQ $D08D
D066      30 25          BMI $D08D
D068      A9 01          LDA #$01
D06A      85 67          STA $67
D06C      20 91 D6       JSR $D691
D06F      20 5A DD       JSR $DD5A
D072      20 70 F5       JSR $F570
D075      20 8A F5       JSR $F58A
D078      A9 40          LDA #$40
D07A      85 CA          STA $CA
D07C      20 38 F5       JSR $F538
D07F      20 5D D3       JSR $D35D
D082      20 34 DB       JSR $DB34
D085      A5 CA          LDA $CA
D087      D0 F3          BNE $D07C
D089      A9 00          LDA #$00
D08B      85 67          STA $67
D08D      AD 14 19       LDA $1914
D090      85 A0          STA TEMP0
D092      AD 0D 19       LDA $190D
D095      85 B7          STA FRMCNT
D097      A9 00          LDA #$00                                    ;Clear count for each object on
this wave,
D099      A2 0F          LDX $0F                                     ; it will be recalc'd at $D0A1

D09B      9D 06 19       STA OBJS_PER_LEVEL_$1906,X
D09E      CA             DEX
D09F      10 FA          BPL $D09B                                   ;

;Count number of each object type left on screen

D0A1      A0 52          LDY $52                                     ;$52 object to count
D0A3      B9 91 1F       LDA SPRITE_STATE_$1F91,Y                    ;Get sprite enabled flag
D0A6      29 03          AND #$03                                    ;Mask off
D0A8      F0 18          BEQ $D0C2                                   ;Branch if it's not enabled
D0AA      C9 03          CMP #$03                                    ;
D0AC      F0 14          BEQ $D0C2                                   ;Branch if it's in state 3
D0AE      B9 8C 1E       LDA SPRITE_TYPE_$1E8C,Y                     ;Get sprite type
D0B1      29 1F          AND #$1F                                    ;Mask it off
D0B3      F0 0D          BEQ $D0C2                                   ;Skip if it's the player
D0B5      C9 10          CMP #$10                                    ;Electrode?
D0B7      F0 04          BEQ $D0BD                                   ;Count it if it's an electrode
D0B9      C9 0B          CMP #$0B                                    ;
D0BB      B0 05          BCS $D0C2                                   ;Don't count if a
Prog,Enf,Brain,Tank shot
D0BD      AA             TAX                                               ;make x an index
D0BE      CA             DEX                                               ;offset by one because
of player
D0BF      FE 06 19       INC OBJS_PER_LEVEL_$1906,X                  ;Increment the items count
D0C2      88             DEY                                               ;Next sprite
D0C3      D0 DE          BNE $D0A3                                   ;Done?
D0C5      AD 0D 19       LDA $190D                                   ;Get number of Enforcers
D0C8      F0 11          BEQ $D0DB                                   ;Branch if there aren't any
D0CA      18             CLC
D0CB      69 03          ADC #$03
D0CD      4A             LSR A
D0CE      4A             LSR A
D0CF      D0 02          BNE $D0D3
D0D1      A9 01          LDA #$01
```

```
D0D3     CD 0B 19       CMP $190B                                   ;Compare to number of
Spheroids
D0D6     90 03          BCC $D0DB
D0D8     8D 0B 19       STA $190B
D0DB     A5 B7          LDA FRMCNT
D0DD     8D 0D 19       STA $190D
D0E0     A5 A0          LDA TEMP0
D0E2     8D 14 19       STA $1914
D0E5     68             PLA
D0E6     68             PLA
D0E7     4C A6 90       JMP $90A6
;
; Roughly GETSTAMP in original code
;
; expects
;
; x = index of object to get stamp info for
;
; returns
;
; $AF = sprite width (not same as stamp width)
; $AB = stamp width
; $AC = stamp height
; $B3 = low pointer to stamp

GETSTAMP_$D0EA
D0EA     BD 8C 1E       LDA CRTBL,X                                 ;Get sprite type
D0ED     29 1F          AND #$1F                                    ;Mask it off
D0EF     A8             TAY                                               ;move it to Y
D0F0     B9 80 FC       LDA $FC80,Y                                 ;Get page data is on
D0F3     85 B4          STA TEMP20                                        ;Store in pointer
D0F5     BD D9 1C       LDA SATBL,X                                 ;Get animation frame
D0F8     C0 0C          CPY #$0C                                    ;Enforcer shot?
D0FA     D0 02          BNE $D0FE                                   ;Branch if not
D0FC     29 03          AND #$03                                    ;Mask it
D0FE     85 B3          STA TEMP19                                        ;Store in low byte of
pointer

*
********************************************************************************
* THIS ROUTINE MUST:
*         FREEZE EVERYTHING
*         DO SOME FUNNY COLOR CYCLING WITH MC
*         MAKE HIM DISAPPEAR
*         DECREMENT LIVES LEFT AND CHECK FOR END OF GAME
*         REARRANGE OBJECTS
*         GET RID OF MISSILES AND PROGS
*         REPACK ENFORCERS INTO SPHEROIDS
*         OPTIMIZE OBJTBL (ELIMINATE NULL OBJECTS)
*         FINALLY, EITHER GO TO TITLE PAGE, OR JMP WAVESTRT
*

******************************************************************************
*                                                                            *
*         GETSTAMP                                                           *
*                       EXTRACT POINTER TO STAMP PLUS OTHER STATISTICS       *
*                         FROM THE TABLES GIVEN Y, CREDIR, AND STEP          *
*                                                                            *
******************************************************************************
*
*
*         POINTER TO THE ACTUAL STAMP IS RETURNED IN TEMP19 AND 20
*         ANIMATION STEP IS HANDLED DIRECTLY WITH THE OBJTBL
*         TEMP15 RETURNS THE WIDTH AND PALETTE, ENCODED LIKE THE HEADER
*         TEMP12 IS THE HEIGHT OF THE PARTICULAR ANIMATION
*         TEMP11 IS WIDTH IN PIXELS
*         THE CREATURE DIRECTION SHOULD BE IN THE ACCUMULATOR ON CALLING
*
*         TEMPORARILY CHANGED TO GET THE DIRECTION FROM DXTBL
*         FOR ALL CREATURES LESS THAN SPHEROIDS OR EQUAL TO BRAIN
*
GETSTAMP:
D100     A9 00          LDA #$00                                    ;@@@ TEMPORARILY SET
```

```
DIRECTION TO ZERO
D102      C0 0A          CPY #$0A                                      ;@@@ SEE IF BRAIN
D104      F0 0C          BEQ GETSTAM1_$D112                            ;@@@
D106      C0 0B          CPY #$0B                                      ;@@@ SEE IF ITS A SHOT
D108      F0 08          BEQ GETSTAM1_$D112                            ;@@@ GET DIR FROM DXTBL IF IT
IS
D10A      C0 0F          CPY #$0F
D10C      B0 04          BCS GETSTAM1_$D112
D10E      C0 06          CPY #$06                                      ;@@@ SEE IF MO,D,MI OR H
D110      B0 03          BCS GETSTAM2                                  ;@@@ GO OUT IF NOT

GETSTAM1:
D112      BD D4 1B       LDA DXTBL,X                                   ;@@@ GET DIRECTION

GETSTAM2:
D115      18             CLC
D116      79 91 FC       ADC CRETODST,Y                                ;ADD START OF DIRS FOR THAT
CREATURE
D119      A8             TAY                                           ;TO THE DIR WHICH IS IN THE ACC
D11A      B9 53 FE       LDA PALNWID,Y                                 ;GET PALETTE AND WIDTH
D11D      85 AF          STA TEMP15
D11F      B9 F9 FD       LDA STAMPHGH,Y                                ;GET STAMP HEIGHT
D122      85 AC          STA TEMP12
D124      B9 AD FE       LDA STAMPPWD,Y                                ;GET STAMP'S WIDTH IN PIXELS
D127      85 AB          STA TEMP11
D129      B9 A2 FC       LDA DIRTOSTE,Y                                ;GET START OF STEPS FOR THAT
CRE + DIR
D12C      65 B3          ADC SATBL,X                                   ;ADD STEP OF ANIMATION TO GET
STAMP #
D12E      A8             TAY
D12F      B9 FC FC       LDA #H(STAMPS)                                ;GET HIGH POINTER TO STAMP
D132      85 B3          STA TEMP20
D134      60             RTS                                           ;56 CYCLES

*
*****************************************************************************
*                                                                           *
*         GETEXTEN                                                          *
*                        EXTRACT STATISTICS                                 *
*                          FROM THE TABLES GIVEN X, CREDIR, AND STEP        *
*                                                                           *
*****************************************************************************
*
*         THIS IS JUST A SUBSET OF GETSTAMP THAT ONLY WORRIES ABOUT EXTENTS.
*         ANIMATION STEP IS HANDLED DIRECTLY WITH THE OBJTBL.
*         TEMP12 IS THE HEIGHT OF THE PARTICULAR ANIMATION
*         TEMP11 IS WIDTH IN PIXELS
*         THE CREATURE DIRECTION SHOULD BE IN THE ACCUMULATOR ON CALLING
*         OBJECT INDEX IN X
*
GETEXTEN:
D135      84 A4          STY TEMP4                                     ;SAVE Y
D137      85 B7          STA CRTBL
D139      18             CLC
D13A      BD 8C 1E       LDA CRTBL,X
D13D      29 1F          LDA #$1F
D13F      A8             TAY
D140      A5 B7          LDA CRTBL                                     ;GET THE CREATURE TYPE
D142      79 91 FC       ADC CRETODST,Y                                ;ADD START OF DIRS FOR THAT
CREATURE
D145      A8             TAY                                           ;TO THE DIR WHICH IS IN THE ACC
D146      B9 F9 FD       LDA STAMPHGH,Y                                ;GET STAMP HEIGHT
D149      85 AC          STA TEMP12
D14B      B9 AD FE       LDA STAMPPWD,Y                                ;GET STAMP'S WIDTH IN PIXELS
D14E      85 AB          STA TEMP11
D150      A4 A4          LDY TEMP4                                     ;RESTORE Y
D152      60             RTS                                           ;THIS TAKES 31 CYCLES + JSR +
RTS

*****************************************
*****************************************
*                                       *
*         MISCELLANEOUS ROUTINES        *
*                                         *
*******************************************

*****************************************************************************
*                                                                           *
*     CHKOBJBD -- CHECK OBJECT BOUNDARIES                                   *
*             GIVEN A CREATURE INDEX IN X, THIS WILL LOOK AT                *
*             X AND Y POSITION AND EXTENTS AND MUNG THE ENTRIES IN          *
*             THE OBJECT DATA TABLES IF THE OBJECT IS OUT OF BOUNDS         *
*             TEMP4 IS A FLAG,  > 0  IF OBJ HAD TO BE PUT BACK ONSCREEN     *
*                                                                           *
*****************************************************************************
*
CHKOBJBD:
D153     A9 00          LDA #$00
D155     85 A4          STA TEMP4                                  ;FLAG STARTS OUT AS 0
D157     A9 02          LDA #MINX                                  ;GET MIN X
D159     38             SEC
D15A     FD CF 1A       SBC XTBL,X                                 ;SUBTRACT X
D15D     90 11          BCC CHKOBD1                                ;JUMP PAST IF OK
D15F     18             CLC
D160     7D E3 1E       ADC XEXTBL,X                               ;ADD AMOUNT THAT IT'S OVER TO
THE
D163     9D E3 1E       STA XEXTBL,X                               ;CURRENT RIGHT EDGE
D166     A9 02          LDA #MINX                                  ;SET LEFT EDGE OF CREATURE TO
D168     9D CF 1A       STA XTBL,X                                 ;THE MIN X
D16B     E6 A4          INC TEMP4                                  ;SET FLAG
D16D     4C 86 D1       JMP CHKOBDY                                ;GO ON TO DO Y

CHKOBD1:
D170     A9 9C          LDA #MAXX                                  ;GET MAX X
D172     38             SEC
D173     FD E3 1E       SBC XEXTBL,X                               ;SUBTRACT THE FAR EDGE
D176     B0 0E          BCS CHKOBDY                                ;JUMP PAST IF OK
D178     18             CLC
D179     7D CF 1A       ADC XTBL,X                                 ;ADD AMOUNT THAT IT'S OVER TO
THE
D17C     9D CF 1A       STA XTBL,X                                 ;CURRENT LEFT EDGE
D17F     A9 9C          LDA #MAXX                                  ;SET RIGHT EDGE OF CREATURE TO
D181     9D E3 1E       STA XEXTBL,X                               ;THE MAX X
D184     E6 A4          INC TEMP4                                  ;SET FLAG

CHKOBDY:
D186     A9 12          LDA #MINY                                  ;GET MIN Y
D188     38             SEC
D189     FD 26 1B       SBC YTBL,X                                 ;SUBTRACT Y
D18C     90 0F          BCC CHKOBD2                                ;JUMP PAST IF OK
D18E     18             CLC
D18F     7D 3A 1F       ADC YEXTBL,X                               ;ADD AMOUNT THAT IT'S OVER TO
THE
D192     9D 3A 1F       STA YEXTBL,X                               ;CURRENT TOP EDGE
D195     A9 12          LDA #MINY                                  ;SET LEFT EDGE OF CREATURE TO
D197     9D 26 1B       STA YTBL,X                                 ;THE MIN Y
D19A     E6 A4          INC TEMP4                                  ;SET FLAG
D19C     60             RTS                                        ;WE ARE OUT OF HERE

CHKOBD2:
D19D     A9 BC          LDA #MAXY                                  ;GET MAX Y
D19F     38             SEC
D1A0     FD 3A 1F       SBC YEXTBL,X                               ;SUBTRACT THE FAR EDGE
D1A3     B0 F7          BCS CHKOBD2-1                              ;JUMP OUT IF OK - GO TO AN RTS
D1A5     18             CLC
D1A6     7D 26 1B       ADC YTBL,X                                 ;ADD AMOUNT THAT IT'S OVER TO
THE
D1A9     9D 26 1B       STA YTBL,X                                 ;CURRENT FAR EDGE
D1AC     A9 BC          LDA #MAXY                                  ;SET BOTTOM EDGE OF CREATURE
TO
D1AE     9D 3A 1F       STA YEXTBL,X                               ;THE MAX Y
D1B1     E6 A4          INC TEMP4                                  ;SET FLAG
D1B3     60             RTS

********************************************************************************
*                                                                              *
*         CHKINTBD -- CHECK INTENDED POSITION AND EXTENTS VERSUS BORDER         *
*                 LOOKS AT X,YINTEND AND X,YXINTEND AND MUNGES IF NECESSARY     *
*                 TEMP4 IS A FLAG,  > 0  IF OBJ HAD TO BE PUT BACK ONSCREEN     *
*                 NEEDS TO HAVE A VALID X INDEX                                 *
*                                                                               *
*********************************************************************************
*
;This green commentary compliments of Dan Boris & "Scotty"
; Check within boundary playfields, and adjust as necessary
; Inputs
; x = index of object
; $BE = X of object
; $BF = Y of object
; $C0 = X + width of object (known as X extent)
; $C1 = Y + height of object (known as Y extent)
; Outputs
; $BE, $BF, $C0, $C1 adjusted as necessary to be valid coordinates in playfield
; $A4 is nonzero if any adjustment to coordinates was required

CHKINTBD:
D1B4     A9 00          LDA #$00                                    ;RESET THE RESET FLAG
D1B6     85 A4          STA TEMP4
D1B8     A5 BE          LDA XINTEND
D1BA     18             CLC
D1BB     69 10          ADC #0A
D1BD     C9 12          CMP #MINX+$A                                ;CHECK FOR RIGHT EDGE
D1BF     90 06          BCC CHKINT1                                 ;RESET POSITION
D1C1     A5 C0          LDA XXINTEND
D1C3     C9 9C          CMP #MAXX                                   ;CHECK FOR FAR EDGE
D1C5     90 0C          BCC CHKINTY                                 ;GO ON TO Y

CHKINT1:
D1C7     BD CF 1A       LDA XTBL,X                                  ;GET ORIG X
D1CA     85 BE          STA XINTEND                                 ;RESTORE TO ITS FORMER
POSITION
D1CC     BD E3 1E       LDA XEXTBL,X                                ;WHICH WAS HOPEFULLY ON SCREEN
D1CF     85 C0          STA XXINTEND
D1D1     E6 A4          INC TEMP4                                   ;SET FLAG

CHKINTY:
D1D3     A5 BF          LDA YINTEND
D1D5     C9 12          CMP #MINY                                   ;CHECK FOR NEAR EDGE
D1D7     90 07          BCC CHKINT2                                 ;RESET POSITION
D1D9     A5 C1          LDA YXINTEND
D1DB     C9 BC          CMP #MAXY                                   ;CHECK FOR FAR EDGE
D1DD     B0 01          BCS CHKINT2                                 ;GO ON TO Y
D1DF     60             RTS

CHKINT2:
D1E0     BD 26 1B       LDA YTBL,X                                  ;GET ORIG Y
D1E3     85 BF          STA YINTEND                                 ;RESTORE TO ITS FORMER
POSITION
D1E5     BD 3A 1F       LDA YEXTBL,X                                ;WHICH WAS HOPEFULLY ON SCREEN
D1E8     85 C1          STA YXINTEND
D1EA     E6 A4          INC TEMP4                                   ;SET FLAG TO A NON ZERO VALUE
D1EC     60             RTS

;This green commentary compliments of Dan Boris & "Scotty"
;
; This routine works kinda like CHKINTBD but has some special stuff specifically for the Tank and the
Progs routines
; that call it
;
; Inputs
; x = index of object
; $BE = X
; $BF = Y
; $C0 = X Extent
; $C1 = Y Extent
;
; Outputs
; A = the adjusted part of
; $BE, $BF, $C0, $C1 adjusted as necessary to keep the coordinates in playfield
; $A4 - nonzero if X or Y or X Extent or YExtent are out of bounds.

;Disassembly of $D1ED-$D232 compliments of Dan Boris & "Scotty"
D1ED      A9 00          LDA #$00
D1EF      85 A4          STA TEMP4
D1F1      A5 BE          LDA XINTEND_BE                              ;Get X parameter
D1F3      18             CLC
D1F4      69 10          ADC #$10                                    ;Add 16 to X
D1F6      C9 12          CMP #$12                                    ; 18?
D1F8      90 15          BCC $D20F                                   ;Less than 18, we're out of
bounds, reset the X coord
D1FA      A5 C0          LDA $C0                                     ;Get X extent parameter
D1FC      C9 9C          CMP #$9C                                    ; #$9C?
D1FE      90 1B          BCC $D21B                                   ;Less

; if we get here, our X extent is #$9C or more. But if the object's going west, then that's OK, it
is heading back into valid bounds.
D200      BD D4 1B       LDA SPRITE_DELTA_X_$1BD4,X
D203      C9 03          CMP #$03                                    ;Heading West?
D205      F0 14          BEQ $D21B                                   ;Yes, so we don't need to reset
X coord in $BE
D207      C9 06          CMP #$06                                    ;SouthWest?
D209      F0 10          BEQ $D21B                                   ;Yes, so we don't need to reset
X coord in $BE
D20B      C9 07          CMP #$07                                    ;NorthWest?
D20D      F0 0C          BEQ $D21B                                   ;Yes, so we don't need to reset
X coord in $BE

; if we get here, we need to reset $BE and $C0 to the current sprite X and sprite X extent
D20F      BD CF 1A       LDA SPRITE_X,X
D212      85 BE          STA XINTEND_BE
D214      BD E3 1E       LDA SPRITE_X_EXTENT,X
D217      85 C0          STA XXINTEND_C0
D219      E6 A4          INC TEMP4                                   ;And set a flag to indicate
that we've to reset the X component

; do the Y component of the object
D21B      A5 BF          LDA YINTEND_BF                              ;Get Y parameter
D21D      C9 12          CMP #$12                                    ; 18?
D21F      90 07          BCC $D228                                   ;Less, we're out of bounds,
reset the Y coord
D221      A5 C1          LDA YYINTEND_C1                             ;Get Y extent parameter
D223      C9 BC          CMP #$BC                                    ; #$BC?
D225      B0 01          BCS $D228                                   ;More or equal so we're out of
bounds,
                                                                     ;  reset the Y coord
D227      60             RTS                                         ;Otherwise, we're done here

; reset Y and Y extent parameters to the current sprite Y and Y extent
D228      BD 26 1B       LDA SPRITE_Y,X
D22B      85 BF          STA YINTEND_BF
D22D      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
D230      85 C1          STA YYINTEND_C1
D232      E6 A4          INC TEMP4                                   ;And set a flag to indicate
that we've
                                                                     : to reset the Y component.
D234      60             RTS
*
********************************************************************************
*                                                                              *
*      CHKXBD,CHKYBD  --  TAKE AN X OR Y POSITION AND IF OFF SCREEN EDGE,      *
*                                    WILL PLACE AT SCREEN EDGE                 *
*                             CALLED WITH X OR Y POS IN A, RETURNS A           *
*                                                                              *
********************************************************************************
*
;This green commentary compliments of Dan Boris & "Scotty"
; CHKXBD in original source, I can't see anything that calls this,
:   so the code is unused I reckon...
; Inputs
; A is X coordinate
; Outputs
; A is X coordinate, adjusted if necessary to be in bounds
```

```
;
CHKXBD:
D235      C9 9C          CMP #MAXX                                  ;TAKES XPOS IN ACC AND CHECKS
FAR SIDE
D237      90 05          BCC XBD1
D239      A9 9C          LDA #MAXX                                  ;PUT BACK ON SCREEN IF TOO FAR
D23B      4C 44 D2       JMP XBD2

XBD1:
D23E      C9 02          CMP #MINX                                  ;CHECK NEAR SIDE
D240      B0 02          BCS XBD2
D242      A9 02          LDA #MINX                                  ;PUT IT BACK ON SCREEN

XBD2:
D244      60             RTS

CHKYBD:
D245      C9 BC          CMP #MAXY
D247      90 05          BCC YBD1                                   ;CHECK TOP
D249      A9 BC          LDA #MAXY                                  ;PUT IT BACK ON SCREEN
D24B      4C 54 D2       JMP YBD2

YBD1:
D24E      C9 12          CMP #MINY                                  ;CHECK BOTTOM
D250      B0 02          BCS YBD2
D252      A9 12          LDA #MINY                                  ;PUT IT ON SCREEN

YBD2:
D254      60             RTS

;Disassembly of $D255-$D363 compliments of Dan Boris & "Scotty"
;
;
; Update score
;
;
D255      A5 E3          LDA $E3                                    ;Read game mode (0 = Attract)
D257      F0 05          BEQ $D25E
D259      A5 74          LDA $74                                    ;In play mode ? (1 = Play)
D25B      F0 01          BEQ $D25E
D25D      60             RTS

;*****************************************************8
;Update score on display

D25E      86 B7          STX FRMCNT
D260      A6 61          LDX $61                                    ;Get player number
D262      D0 14          BNE $D278                                  ;Branch if player 2
D264      A9 00          LDA #$00                                   ;Set pointer to $2100
D266      85 BA          STA TADDRL                                 ;
D268      A9 21          LDA #$21                                   ;
D26A      85 BB          STA TADDRH                                 ;
D26C      A9 01          LDA #$01
D26E      85 A1          STA TEMP1
D270      A9 FF          LDA #$FF
D272      85 A5          STA TEMP5
D274      A2 03          LDX $03
D276      D0 12          BNE $D28A

D278      A9 07          LDA #$07                                   ;Set pointer to $2107
D27A      85 BA          STA TADDRL                                 ;
D27C      A9 21          LDA #$21                                   ;
D27E      85 BB          STA TADDRH                                 ;
D280      A9 0B          LDA #$0B
D282      85 A1          STA TEMP1
D284      A9 03          LDA #$03
D286      85 A5          STA TEMP5
D288      A2 07          LDX $07

D28A      A9 00          LDA #$00
D28C      A8             TAY
D28D      85 A2          STA TEMP2
D28F      B5 40          LDA $40,X                                  ;Get score
```

```
D291      85 A3           STA TEMP3
D293      CA              DEX
D294      4C C0 D2        JMP $D2C0

D297      B5 40           LDA $40,X
D299      85 A3           STA TEMP3
D29B      4A              LSR A
D29C      4A              LSR A
D29D      4A              LSR A
D29E      4A              LSR A
D29F      85 A4           STA TEMP4
D2A1      D0 08           BNE $D2AB
D2A3      A5 A2           LDA TEMP2
D2A5      D0 04           BNE $D2AB
D2A7      91 BA           STA (TADDRL),Y
D2A9      F0 0B           BEQ $D2B6
D2AB      A5 A4           LDA TEMP4
D2AD      18              CLC
D2AE      65 A1           ADC TEMP1
D2B0      91 BA           STA (TADDRL),Y
D2B2      A9 01           LDA #$01
D2B4      85 A2           STA TEMP2
D2B6      C8              INY
D2B7      CA              DEX
D2B8      E4 A5           CPX TEMP5
D2BA      D0 04           BNE $D2C0
D2BC      A9 01           LDA #$01
D2BE      85 A2           STA TEMP2

D2C0      A5 A3           LDA TEMP3
D2C2      29 0F           AND #$0F
D2C4      85 A4           STA TEMP4
D2C6      D0 09           BNE $D2D1
D2C8      A5 A2           LDA TEMP2
D2CA      D0 05           BNE $D2D1
D2CC      91 BA           STA (TADDRL),Y
D2CE      4C DC D2        JMP $D2DC

D2D1      A5 A4           LDA TEMP4
D2D3      18              CLC
D2D4      65 A1           ADC TEMP1
D2D6      91 BA           STA (TADDRL),Y
D2D8      A9 01           LDA #$01
D2DA      85 A2           STA TEMP2
D2DC      C8              INY
D2DD      E4 A5           CPX TEMP5
D2DF      D0 B6           BNE $D297
D2E1      A6 B7           LDX FRMCNT
D2E3      60              RTS
;
; **************************************************************
; Write level data to display
;
D2E4      86 B7           STX FRMCNT                                    ;Save X
D2E6      A0 00           LDY $00                                       ;index of level digit
D2E8      A6 61           LDX $61                                       ;Get player number
D2EA      D0 0E           BNE $D2FA                                     ;Branch if player 2
D2EC      A9 1C           LDA #$1C                                      ;Pointer to $211C
D2EE      85 BA           STA TADDRL                                    ;
D2F0      A9 21           LDA #$21                                      ;
D2F2      85 BB           STA TADDRH                                    ;
D2F4      A9 01           LDA #$01
D2F6      85 A1           STA TEMP1
D2F8      D0 0C           BNE $D306                                     ;Branch Always

D2FA      A9 23           LDA #$23                                      ;Pointer to $2123
D2FC      85 BA           STA TADDRL                                    ;
D2FE      A9 21           LDA #$21                                      ;
D300      85 BB           STA TADDRH                                    ;
D302      A9 0B           LDA #$0B
D304      85 A1           STA TEMP1
D306      B5 E8           LDA $E8,X                                     ;Get level
D308      85 A3           STA TEMP3                                     ;Save it
```

```
D30A      4A              LSR A                                       ;Shift down top digit
D30B      4A              LSR A                                       ;
D30C      4A              LSR A                                       ;
D30D      4A              LSR A                                       ;
D30E      D0 04           BNE $D314
D310      91 BA           STA (TADDRL),Y                              ;Write character to memory
D312      F0 05           BEQ $D319                                   ;Branch always
D314      18              CLC
D315      65 A1           ADC TEMP1
D317      91 BA           STA (TADDRL),Y                              ;Write character to memory
D319      C8              INY                                         ;next character
D31A      A5 A3           LDA TEMP3                                   ;Get level from temp
D31C      29 0F           AND #$0F                                    ;mask off bottom 4 bits
D31E      18              CLC
D31F      65 A1           ADC TEMP1
D321      91 BA           STA (TADDRL),Y                              ;Write character to memory
D323      A6 B7           LDX FRMCNT                                  ;Restore X
D325      60              RTS

;*******************************************************************
;Update lives display
;
D326      A5 E3           LDA $E3                                     ;In attract mode?
D328      F0 05           BEQ $D32F                                   ;Branch if not in attract mode

D32A      A5 74           LDA $74                                     ;In play mode?
D32C      F0 01           BEQ $D32F
D32E      60              RTS

D32F      A6 61           LDX $61                                     ;Get player number
D331      D0 0A           BNE $D33D                                   ;Player 1?
D333      A9 0E           LDA #$0E                                    ;Pointer to $210E
D335      85 BA           STA TADDRL                                  ;
D337      A9 21           LDA #$21                                    ;
D339      85 BB           STA TADDRH                                  ;
D33B      D0 08           BNE $D345                                   ;Player 2?
D33D      A9 15           LDA #$15                                    ;Pointer to $2115
D33F      85 BA           STA TADDRL                                  ;
D341      A9 21           LDA #$21                                    ;
D343      85 BB           STA TADDRH                                  ;
D345      B5 E0           LDA $E0,X                                   ;Get number of lives for player
D347      85 B6           STA TEMP22                                  ;Save it
D349      A0 00           LDY $00                                     ;
D34B      C6 B6           DEC TEMP22                                  ;Count down players
D34D      10 04           BPL $D353                                   ;Below 0?
D34F      A9 00           LDA #$00                                    ;If so set character to 0
D351      F0 02           BEQ $D355                                   ;Branch always
D353      A9 15           LDA #$15                                    ;Set character to 15
D355      91 BA           STA (TADDRL),Y                              ;Put it on the screen
D357      C8              INY                                         ;Next
D358      C0 07           CPY #$07                                    ;Draw a maximum of 7 live
indicators
D35A      90 EF           BCC $D34B                                   ;
D35C      60              TRS                                         ;

D35D      A5 CA           LDA $CA
D35F      29 03           AND #$03
D361      D0 01           BNE $D364
D363      60              RTS

*
****************************************************************************
*                                                                          *
*      MAIN    -- MASTER LOOP   -   LOOPS ONCE EACH FRAME DURING PLAY      *
*                                                                          *
****************************************************************************
*
MAIN:
D364      A5 C7           LDA TEMPCOL                                 ;USE TO HOLD MC COLOR 3
D366      18              CLC                                         ; SINCE PALETTES ARE
UNREADABLE
D367      69 01           ADC #$01
D369      85 C7           STA TEMPCOL
```

```
D36B      29 OF          AND #$0F
D36D      C9 0E          CMP #$0E
D36F      90 06          BCC $D387
D371      A9 04          LDA #$04
D373      85 B7          STA FRMCNT
D375      D0 08          BCC $D37F
D377      C9 03          CMP #$03
D379      B0 0C          BCC $D387
D37B      A9 0D          LDA #$0D
D37D      85 B7          STA FRMCNT
D37F      A5 C7          LDA TEMPCOL
D381      29 F0          AND #$F0
D383      05 B7          ORA FRMCNT
D385      85 C7          STA TEMPCOL

D387      85 23          STA P0C3
D389      A5 C7          LDA TEMPCOL
D38B      29 0F          AND #MASKL
D38D      09 30          ORA #$30                                  ;MAKE THIS INTO A RED
D38F      85 37          STA P5C3                                  ;ELECTRODES/SKULL EYES/SCORE
NUMBERS
D391      49 0F          EOR #$0F
D393      85 39          STA P6C1                                  ;BRAIN EYES AND PART OF HEAD
D395      A5 C7          LDA TEMPCOL
D397      49 FF          EOR #$FF
D399      29 0F          AND #MASKL
D39B      09 90          ORA #$90                                  ;MAKE THIS INTO A BLUE
D39D      85 36          STA P5C2                                  ;PALETTE 5 COLOR CYCLING
D39F      A5 C7          LDA TEMPCOL
D3A1      49 FF          EOR #$FF
D3A3      29 0F          AND #MASKL
D3A5      85 33          STA P4C3                                  ;CYCLE HULK ARMS
D3A7      60             RTS

*
***********************************************************************
*                                                                     *
*     RANDOM  - RETURN WITH A RANDOM NUMBER 0 - 255 IN ACC & $C5      *
*                                                                     *
***********************************************************************
*
RANDOM:
D3A8      98             TYA
D3A9      48             PHA
D3AA      A5 E5          LDA $E5
D3AC      65 C5          ADC RNDM
D3AE      65 C6          ADC RNDM+1
D3B0      E5 D0          SBC #$D0
D3B2      A4 C5          LDY RNDM
D3B4      84 C6          STY RNDM+1
D3B6      85 C5          STA RNDM
D3B8      68             PLA
D3B9      A8             TAY
D3BA      A5 C5          LDA $C5
D3BC      60             RTS

*
********************************************************************************
*                                                                              *
*      RANDXY -- RETURNS RANDOM X AND Y POSITIONS IN VALID SCREEN AREA         *
*                RETURNS VALUES IN RANDOMX AND RANDOMY                         *
*                                                                              *
********************************************************************************
*
RANDXY:
D3BD      20 A8 D3       JSR RANDOM                                ;GET A RANDOM NUMBER IN A
D3C0      C9 12          CMP #MINY                                 ;MAKE SURE THAT IT IS WITHIN X
RANGE
D3C2      90 F9          BCC RANDXY                                ;GO AGAIN IF OFF SCREEN
D3C4      C9 9C          CMP #MAXX
D3C6      B0 F5          BCS RANDXY
D3C8      85 C3          STA RANDOMX
```

```
RANDXY1:
D3CA      20 A8 D3       JSR RANDOM                                  ;GET ANOTHER RANDOM NUMBER FOR
Y
D3CD      C9 12          CMP #MINY                                   ;LIKEWISE Y IN RANGE
D3CF      90 F9          BCC RANDXY1                                 ;IF OUT OF BOUNDS FIND ANOTHER
Y
D3D1      C9 AE          CMP #MAXY
D3D3      B0 F5          BCS RANDXY1
D3D5      85 C4          STA RANDOMY

RANDXYDN:
D3D7      60             RTS                                         ;COMPLETELY CHECKS OUT

*
********************************************************************************
*                                                                              *
*      RANDXYBX -- RETURNS RANDOM X AND Y POSITIONS NOT TOO CLOSE TO CENTER     *
*                 USEFUL FOR PLACING OBJECTS ON SCREEN AT START OF WAVE         *
*                 RETURNS VALUES IN RANDOMX AND RANDOMY.                        *
*                                                                              *
********************************************************************************
*
RANDXYBX:
D3D8      20 A8 D3       JSR RANDOM                                  ;GET A RANDOM NUMBER IN A
D3DB      C9 02          CMP #MINX                                   ;MAKE SURE THAT IT IS WITHIN X
RANGE
D3DD      90 F9          BCC RANDXYBX                                ;GO AGAIN IF OFF SCREEN
D3DF      C9 95          CMP #MAXX-HWID                              ;SUBTRACT HWID SO A LARGE OBJ
DOESN'T HANG OFF
D3E1      B0 F5          BCS RANDXYBX
D3E3      85 C3          STA RANDOMX

RANDXYB1:
D3E5      20 A8 D3       JSR RANDOM                                  ;GET ANOTHER RANDOM NUMBER FOR
Y
D3E8      C9 12          CMP #MINY                                   ;LIKEWISE Y IN RANGE
D3EA      90 F9          BCC RANDXYB1                                ;IF OUT OF BOUNDS FIND ANOTHER
Y
D3EC      C9 AE          CMP #MAXY-HHEIGHT
D3EE      B0 F5          BCS RANDXYB1
D3F0      85 C4          STA RANDOMY

*          NOW CHECK THAT X,Y IS NOT IN CENTER BOX

D3F2      C9 40          CMP #SBOXMINY-HHEIGHT                       ;Y-POS ALREADY IN A
D3F4      90 0E          BCC RANDXYBD                                ;OUTSIDE BOX
D3F6      C9 80          CMP #SBOXMAXY
D3F8      B0 0A          BCS RANDXYBD                                ;OUTSIDE BOX
D3FA      A5 C3          LDA RANDOMX                                 ;THE X-POS WE GENERATED
D3FC      C9 28          CMP #SBOXMINX-HWID
D3FE      90 04          BCC RANDXYBD                                ;OUTSIDE BOX
D400      C9 6B          CMP #SBOXMAXX
D402      90 D4          BCC RANDXYBX                                ;TESTS FAIL SO X,Y IS INSIDE
BOX

RANDXYBD:
D404      60             RTS                                         ;COMPLETELY CHECKS OUT
*

*******************************************************************************
*                                                                             *
*         RANDPM  --  RANDOM PLUS/MINUS                                       *
*                 RETURNS A RANDOM VALUE IN THE RANGE -TEMP4 TO +TEMP4        *
*          TEMP4 MUST BE 1,3,7,F,1F,3F, ETC.. BECAUSE IT IS USED AS A MASK    *
*                                                                             *
*******************************************************************************
*
RANDPM:
D405      20 A8 D3       JSR RANDOM                                  ;GET A RANDOM NUMBER 0 - 255
D408      6A             ROR A                                       ;SET CARRY TO SOMETHING RANDOM
D409      25 A4          AND TEMP4
D40B      90 06          BCC RANDPMDN                                ;OK - LEAVE IT POSITIVE
```

```
*         MAKE A INTO -A

D40D     85 7C          STA TEMP5
D40F     A9 00          LDA #$00
D411     E5 7C          SBC TEMP5                                   ;GET -A  --  CARRY WAS
ALREADY SET

RANDPMDN:
D413     60             RTS

*
**************************************************************************
*                                                                        *
*         RAND2   --  RANDOMLY RETURNS 0, 1 OR 2 IN THE ACCUMULATOR      *
*                         USEFUL FOR CHOOSING RANDOM ANIMATION STEPS     *
*                                                                        *
**************************************************************************
*
RAND2
D414     20 A8 D3       JSR RANDOM
D417     29 03          AND #MASK2
D419     C9 03          CMP #$03
D41B     D0 02          BNE RAND20
D41D     A9 01          LDA #$01

RAND20:
D41F     60             RTS

*********************************************************
*                                                       *
*         RESETSC --  RESET SCORES FOR BOTH PLAYERS     *
*                                                       *
*********************************************************
*
RESETSC:
D420     A9 00          LDA #$00
D422     A2 04          LDX $04
D424     95 40          STA SCORE1L
D426     95 44          STA SCORE1M
D428     CA             DEX
D429     10 F9          BPL $F9
D42B     A9 00          LDA #$00
D42D     85 4C          STA SCORE1H
D42F     85 4D          STA SCORE1V
D431     A9 02          LDA #$02
D433     85 4A          STA SCORE2L
D435     85 4B          STA SCORE2M
D437     A9 50          LDA #$50
D439     85 48          STA SCORE2H
D43B     85 49          STA SCORE2V
D43D     60             RTS
*
**********************************************************************
*                                                                    *
*     FSCORE        PUT THE CORRECT FAMILY SCORE FOR THE FAMILY      *
*                   ONTO THE SCREEN AND CREDIT IT TO THE PLAYER      *
*                                                                    *
**********************************************************************
*

FSCORE
D43E     A5 D4          LDA FAMLEVEL                                ;GET THE CURRENT FAMILY POINT
VALUE
D440     18             CLC
D441     69 01          ADC #$01                                    ;INCREMENT FAMILY VALUE
D443     C9 05          CMP #$05                                    ;SEE IF OVER 5
D445     30 02          BMI ZFSCORE                                 ;A VERY TRANSIENT SYMBOL
D447     A9 05          LDA #$05                                    ;RESET TO 5

ZFSCORE:
D449     85 D4          STA FAMLEVEL                                ;REMEMBER THE POINT VALUE

;HERE MUNGE THE ANIMATION TO GET THE NUMBER
```

```
D44B     18            CLC
D44C     0A            ASL A
D44D     0A            ASL A
D44E     0A            ASL A
D44F     0A            ASL A

;Disassembly of $D450-$D62F compliments of Dan Boris & "Scotty"
D450     85 B7         STA FRMCNT
D452     86 A1         STX TEMP1
D454     A2 00         LDX $00
D456     A4 61         LDY $61
D458     F0 02         BEQ $D45C
D45A     A2 04         LDX $04
D45C     18            CLC
D45D     F8            SED
D45E     A5 B7         LDA FRMCNT
D460     75 41         ADC $41,X
D462     95 41         STA $41,X
D464     B5 42         LDA $42,X
D466     69 00         ADC #$00
D468     95 42         STA $42,X
D46A     B5 43         LDA $43,X
D46C     69 00         ADC #$00
D46E     95 43         STA $43,X
D470     A4 61         LDY $61
D472     20 87 D4      JSR $D487
D475     D8            CLD
D476     A6 A1         LDX TEMP1
D478     A9 01         LDA #$01                                ;Play Family Pick-up Sound
D47A     20 95 E3      JSR DOTUNE_$E395
D47D     84 B6         STY TEMP22
D47F     20 55 D2      JSR $D255
D482     A4 B6         LDY TEMP22
D484     A5 D4         LDA $D4
D486     60            RTS

D487     A5 64         LDA SKILL
D489     C9 04         CMP #$04
D48B     90 01         BCC $D48E
D48D     60            RTS
;
;Check for earning of extra lives
;
D48E     B5 43         LDA $43,X                               ;Get high byte of score
D490     D9 4C 00      CMP $004C,Y                             ;Compare with next bonus
D493     90 40         BCC $D4D5                               ;Branch if score is less then
next bonus score
D495     D0 10         BNE $D4A7                               ;Branch if score is greater
then next bonus score
D497     B5 42         LDA $42,X                               ;Get next lowest byte
D499     D9 4A 00      CMP $004A,Y                             ;Compare with next bonus
D49C     90 37         BCC $D4D5                               ;Branch if score is less then
next bonus score
D49E     D0 07         BNE $D4A7                               ;Branch if score is greater
then next bonus score
D4A0     B5 41         LDA $41,X                               ;Get next lowest byte
D4A2     D9 48 00      CMP $0048,Y                             ;Compute with next bonus
D4A5     90 2E         BCC $D4D5                               ;Branch if score is less then
next bonus score
D4A7     18            CLC                                     ;Earned a bonus life
D4A8     B9 48 00      LDA $0048,Y                             ;Increment next bonus life
score by 25000
D4AB     69 50         ADC #$50                                ;
D4AD     99 48 00      STA $0048,Y                             ;
D4B0     B9 4A 00      LDA $004A,Y                             ;
D4B3     69 02         ADC #$02                                ;
D4B5     99 4A 00      STA $004A,Y                             ;
D4B8     B9 4C 00      LDA $004C,Y                             ;
D4BB     69 00         ADC #$00                                ;
D4BD     99 4C 00      STA $004C,Y                             ;
D4C0     A6 61         LDX $61                                 ;Get current player
D4C2     B5 E0         LDA $E0,X                               ;Get number of lives
```

```
D4C4      C9 7F          CMP #$7F                                        ;
D4C6      B0 02          BCS $D4CA                                       ;If 127 lives or more then
don't give another new life
D4C8      F6 E0          INC $E0,X                                       ;Add another life
D4CA      20 56 E3       JSR $E356                                       ;Sound?
D4CD      A9 05          LDA #$05                                        ;Play Extra Man Sound
D4CF      20 95 E3       JSR DOTUNE_$E395
D4D2      20 26 D3       JSR $D326
D4D5      60             RTS

D4D6      A5 62          LDA CURPLAYERS                                  ;Get current number of player
D4D8      D0 03          BNE $D4DD                                       ;Branch if 2 Players
D4DA      C6 E0          DEC $E0                                         ;Subtract a life from Player 1
D4DC      60             RTS

D4DD      A2 0F          LDX $0F                                         ;Swap active sprite counts
with inactive counts
D4DF      BD 06 19       LDA OBJS_PER_LEVEL_$1906,X                      ;
D4E2      85 A0          STA TEMP0                                       ;
D4E4      BD 16 19       LDA OBJS_PER_WAVE_OTHER_PLAYER_$1916,X          ;
D4E7      9D 06 19       STA OBJS_PER_LEVEL_$1906,X                      ;
D4EA      A5 A0          LDA TEMP0                                       ;
D4EC      9D 16 19       STA OBJS_PER_WAVE_OTHER_PLAYER_$1916,X          ;
D4EF      CA             DEX                                             ;
D4F0      10 ED          BPL $D4DF                                       ;

D4F2      C6 61          DEC $61                                         ;Switch to player 0
D4F4      D0 1B          BNE $D511
D4F6      A9 0F          LDA #$0F
D4F8      85 65          STA $65
D4FA      A9 36          LDA #$36
D4FC      85 66          STA $66
D4FE      C6 E0          DEC $E0

D500      A5 D5          LDA $D5
D502      85 4F          STA $4F
D504      A5 4E          LDA $4E
D506      85 D5          STA $D5

D508      A5 6E          LDA $6E
D50A      85 70          STA $70
D50C      A5 6F          LDA $6F
D50E      85 6E          STA $6E
D510      60             RTS

D511      A2 01          LDX $01
D513      86 61          STX $61
D515      A9 36          LDA #$36
D517      85 65          STA $65
D519      A9 0F          LDA #$0F
D51B      85 66          STA $66
D51D      D6 E0          DEC $E0,X
D51F      A5 D5          LDA $D5
D521      85 4E          STA $4E
D523      A5 4F          LDA $4F
D525      85 D5          STA $D5
D527      A5 6E          LDA $6E
D529      85 6F          STA $6F
D52B      A5 70          LDA $70
D52D      85 6E          STA $6E
D52F      60             RTS
;
;Setup hardware registers and enable DMA
;
D530      A9 00          LDA #$00                                        ;
D532      8D 81 02       STA $0281                                       ;Set Joystick port to input
D535      85 38          STA $38                                         ;Set OFFSET to zero
D537      A9 00          LDA #$00                                        ;
D539      85 20          STA $20                                         ;Set Background color
D53B      20 30 D6       JSR $D630                                       ;Setup color registers
D53E      A9 A8          LDA #$A8                                        ;
D540      85 34          STA $34                                         ;CHARBASE = $A800
D542      A9 04          LDA #$04                                        ;
```

```
D544      85 30          STA $30                                      ;
D546      A9 18          LDA #$18                                     ;
D548      85 2C          STA $2C                                      ;DLL = $1804
D54A      A9 B1          LDA #$B1                                     ;Setup some variables
D54C      85 59          STA $59                                      ;Set DLI pointer to $D5B1
D54E      A9 D5          LDA #$D5                                     ;
D550      85 5A          STA $5A                                      ;
D552      85 24          STA $24                                      ;Wait for Sync
D554      24 28          BIT $28                                      ;
D556      30 FA          BMI $D552                                    ;Wait for end of vblank
D558      24 28          BIT $28                                      ;
D55A      30 F6          BMI $D552                                    ;Check VBLANK again
D55C      85 24          STA $24                                      ;Wait for Sync
D55E      24 28          BIT $28                                      ;
D560      10 FA          BPL $D55C                                    ;Wait for start of vblank
D562      24 28          BIT $28                                      ;Check again
D564      10 F6          BPL $D55C                                    ;
D566      A9 40          LDA #$40                                     ;
D568      85 3C          STA $3C                                      ;Enable DMA   RM = 0
D56A      60             RTS                                          ;
;
;NMI
;
D56B      85 7E          STA $7E                                      ;Save A
D56D      86 7F          STX $7F                                      ;Save X
D56F      BA             TSX                                          ;Save Status
D570      CA             DEX                                          ;
D571      BD 00 01       LDA $0100,X
D574      85 D0          STA $D0
D576      6C 59 00       JMP ($0059)
D579      A5 7E          LDA $7E                                      ;Restore A
D57B      A6 7F          LDX $7F                                      ;Restore X
D57D      40             RTI

D57E      98             TYA                                          ;Save Y
D57F      48             PHA                                          ;
D580      A9 D2          LDA #$D2                                     ;
D582      85 59          STA $59
D584      A9 D5          LDA #$D5
D586      85 5A          STA $5A
D588      A9 00          LDA #$00
D58A      85 E4          STA $E4
D58C      85 24          STA $24
D58E      A5 65          LDA $65
D590      85 3D          STA $3D
D592      A5 66          LDA $66
D594      85 3E          STA $3E
D596      A9 36          LDA #$36
D598      85 3F          STA $3F
D59A      20 A8 D3       JSR RANDOM_$D3A8
D59D      E6 CA          INC $CA
D59F      E6 E5          INC $E5
D5A1      D0 02          BNE $D5A5
D5A3      E6 E6          INC $E6
D5A5      A5 5C          LDA $5C
D5A7      D0 03          BNE $D5AC
D5A9      20 9E E4       JSR $E49E
D5AC      68             PLA
D5AD      A8             TAY
D5AE      4C 79 D5       JMP $D579

D5B1      A9 7E          LDA #$7E
D5B3      85 59          STA $59
D5B5      A9 D5          LDA #$D5
D5B7      85 5A          STA $5A
D5B9      A9 01          LDA #$01
D5BB      85 E4          STA $E4
D5BD      85 24          STA $24
D5BF      A9 ED          LDA #$ED
D5C1      85 3D          STA $3D
D5C3      A9 35          LDA #$35
D5C5      85 3E          STA $3E
D5C7      A9 97          LDA #$97
```

```
D5C9     85 3F          STA $3F
D5CB     A9 A8          LDA #$A8
D5CD     85 34          STA $34
D5CF     4C 79 D5       JMP $D579

D5D2     85 24          STA $24
D5D4     85 24          STA $24
D5D6     85 24          STA $24
D5D8     A5 28          LDA $28
D5DA     C5 28          CMP $28
D5DC     D0 FA          BNE $D5D8
D5DE     0A             ASL A
D5DF     90 08          BCC $D5E9
D5E1     A9 B1          LDA #$B1
D5E3     85 59          STA $59
D5E5     A9 D5          LDA #$D5
D5E7     85 5A          STA $5A
D5E9     4C 79 D5       JMP $D579

D5EC     A9 04          LDA #$04
D5EE     85 59          STA $59
D5F0     A9 D6          LDA #$D6
D5F2     85 5A          STA $5A
D5F4     85 24          STA $24
D5F6     20 91 D6       JSR $D691
D5F9     A9 A8          LDA #$A8
D5FB     85 34          STA $34
D5FD     A9 40          LDA #$40                                    ;Enable DMA,RM=0
D5FF     85 3C          STA $3C                                     ;
D601     4C 79 D5       JMP $D579

D604     A9 EC          LDA #$EC
D606     85 59          STA $59
D608     A9 D5          LDA #$D5
D60A     85 5A          STA $5A
D60C     85 24          STA $24
D60E     A9 39          LDA #$39
D610     85 34          STA $34
D612     A9 4B          LDA #$4B                                    ;Enable DMA,RM=3
D614     85 3C          STA $3C
D616     4C 79 D5       JMP $D579

D619     A9 EC          LDA #$EC
D61B     85 59          STA $59
D61D     A9 D5          LDA #$D5
D61F     85 5A          STA $5A
D621     E6 E5          INC $E5
D623     20 BB F6       JSR $F6BB
D626     60             RTS
D627     48             PHA
D628     A9 42          LDA #$42
D62A     85 20          STA $20
D62C     D0 FE          BNE $D62C
D62E     68             PLA
D62F     40             RTI

*
******************************************
*                                        *
*         PALINIT - INIT PALETTES        *
*                                        *
******************************************
*
*         PALETTE 0 - BACKGROUND
PALINIT:
D630     A9 00          LDA #$00                                    ;
D632     85 20          STA P0C0                                    ;SET BACKGROUND COLOR

*         PALETTE 0 - MUTANT CLONE
D634     A9 0F          LDA #$0F                                    ;WHITE
D636     85 21          STA P0C1                                    ;BODY AND FEET
D638     A9 69          LDA #$69                                    ;PURPLE
D63A     85 22          STA P0C2                                    ;ARMS AND HEAD (BEANIE)
```

```
*         LEAVE COLOR 3 ALONE - IT CYCLES

*         PALETTE 1 - GRUNT AND MIKEY
D63C     A9 0F          LDA #$0F                                     ;WHITE
D63E     85 25          STA P1C1                                     ;NECK STRIPE
D640     A9 36          LDA #$36                                     ;RED
D642     85 26          STA P1C2                                     ;BODY
D644     A9 1F          LDA #$1F                                     ;YELLOW
D646     85 27          STA P1C3                                     ;HEAD AND FEET

*         PALETTE 2 - MOMMY
D648     A9 1D          LDA #$FF                                     ;BLONDE
D64A     85 29          STA P2C1                                     ;HAIR AND LEGS
D64C     A9 59          LDA #$4F                                     ;HOT PINK
D64E     85 2A          STA P2C2                                     ;DRESS
D650     A9 FA          LDA #$DF                                     ;POCKETBOOK GREEN
D652     85 2B          STA P2C3                                     ;POCKETBOOK

*         PALETTE 3 - DADDY
D654     A9 96          LDA #$99                                     ;BLUE
D656     85 2D          STA P3C1                                     ;SUIT
D658     A9 14          LDA #$1A                                     ;DIRTY BLONDE
D65A     85 2E          STA P3C2                                     ;HAIR
D65C     A9 1A          LDA #$14                                     ;BROWNISH
D65E     85 2F          STA P3C3                                     ;FACE AND ATTACHE CASE

*         PALETTE 4 - HULKS
D660     A9 35          LDA #$35                                     ;RED
D662     85 31          STA P4C1                                     ;HEAD AND LEGS
D664     A9 ED          LDA #$ED                                     ;LIGHT GREEN
D666     85 32          STA P4C2                                     ;BODY
D668     A9 00          LDA #$00                                     ;UNIMPORTANT
D66A     85 33          STA P4C3                                     ;ARMS - COLOR CYCLE

*         PALETTE 5 - SKULL AND CROSSBONES, SCORE NUMBERS, ELECTRODES
D66C     A9 0F          LDA #$0F                                     ;WHITE
D66E     85 35          STA P5C1                                     ;SKULL
D670     A9 FF          LDA #$00                                     ;UNIMPORTANT
D672     85 36          STA P5C2                                     ;THIS COLOR CYCLES
D674     A9 FF          LDA #$00                                     ;UNIMPORTANT
D676     85 37          STA P5C3                                     ;THIS COLOR CYCLES

*         PALETTE 6 - BRAINS
D678     A9 00          LDA #$00                                     ;UNIMPORTANT
D67A     85 39          STA P6C1                                     ;EYES AND SOME OF BRAIN - COLOR
CYCLES
D67C     A9 97          LDA #$97                                     ;BLUE
D67E     85 3A          STA P6C2                                     ;BRAIN
D680     A9 EB          LDA #$EB                                     ;GREEN
D682     85 3B          STA P6C3                                     ;BRAINS' FEET

;Disassembly of $D684-$D6B9 compliments of Dan Boris & "Scotty"
D684     A5 65          LDA $65                                      ;
D686     85 3D          STA $3D                                      ;P7C1 from variable
D688     A5 66          LDA $66                                      ;
D68A     85 3E          STA $3E                                      ;P7C2 from variable
D68C     A9 36          LDA #$36                                     ;
D68E     85 3F          STA $3F                                      ;P7C3
D690     60             RTS                                          ;

D691     A9 0F          LDA #$0F
D693     85 35          STA $35
D695     A9 1F          LDA #$1F
D697     85 36          STA $36
D699     A5 68          LDA $68
D69B     85 37          STA $37
D69D     A9 0F          LDA #$0F
D69F     85 39          STA $39
D6A1     A9 1F          LDA #$1F
D6A3     85 3A          STA $3A
D6A5     A9 36          LDA #$36
D6A7     85 3B          STA $3B
```

```
D6A9     A9 0F          LDA #$0F
D6AB     85 3D          STA $3D
D6AD     A9 0F          LDA #$0F
D6AF     85 3E          STA $3E
D6B1     A9 0F          LDA #$0F
D6B3     85 3F          STA $3F
D6B5     A9 00          LDA #$00
D6B7     85 20          STA $20
D6B9     60             RTS

;Disassembly of $D6BA-$D71B compliments of Dan Boris & "Scotty"
;*************************************************************
;Clear score, lives, and level display
MCSEND:
D6BA     A2 07          LDX $07                                     ;7 bytes
D6BC     A9 00          LDA #$00                                    ;

MCSTEMP1:
D6BE     9D 00 21       STA $2100,X                                 ;Clear player 1 score display
D6C1     9D 07 21       STA $2107,X                                 ;Clear player 2 score display
D6C4     9D 0E 21       STA $210E,X                                 ;Clear player 1 lives display
D6C7     9D 15 21       STA $2115,X                                 ;Clear player 2 lives display
D6CA     CA             DEX                                         ;
D6CB     10 F1          MCSTEMP1                                    ;


D6CD     A9 00          LDA #$00                                    ;
D6CF     A2 01          LDX $01                                     ;

MCSTEMP2:
D6D1     9D 1C 21       STA $211C,X                                 ;Clear Level
D6D4     9D 23 21       STA $2123,X
D6D7     CA             DEX                                         ;
D6D8     10 F7          BPL MCSTEMP2                                ;
D6DA     60             RTS                                         ;

;**************************************************************
;Scoring
SCORING:
D6DB     86 B7          STX FRMCNT                                  ;Save X
D6DD     A2 00          LDX $00                                     ;Index of player 0 score
D6DF     A4 61          LDY $61                                     ;Get player #
D6E1     F0 02          BEQ $D6E5                                   ;Player 0?
D6E3     A2 04          LDX $04                                     ;If not then set index to
player 1 score
D6E5     0A             ASL A                                       ; *2
D6E6     A8             TAY                                         ;Move to index
D6E7     18             CLC                                         ;Get ready for math
D6E8     F8             SED                                         ;
D6E9     B9 9B F1       LDA SCORETBL+1,Y                            ;GET LOW 2 DIGITS OF SCORE
D6EC     75 40          ADC SCORE1L,X                               ;Add to score
D6EE     95 40          STA SCORE1L,X                               ;And score
D6F0     B9 9A F1       LDA SCORETBL,Y                              ;HIGH 2 DIGITS OF SCORE
D6F3     75 41          ADC SCORE1M                                 ;Add to score
D6F5     95 41          STA SCORE1M                                 ;And store
D6F7     B5 42          LDA $42,X                                   ;Get next byte of score
D6F9     69 00          ADC #$00                                    ;Add carry
D6FB     95 42          STA $42,X                                   ;And store
D6FD     B5 43          LDA $43,X                                   ;Get top byte of score
D6FF     69 00          ADC #$00                                    ;Add carry
D701     95 43          STA $43,X                                   ;Store
D703     A4 61          LDY $61                                     ;Get player #
D705     20 87 D4       JSR $D487                                   ;Check for bonus lives
D708     D8             CLD                                         ;Turn off decimal mode
D709     A6 B7          LDX FRMCNT
D70B     20 55 D2       JSR $D255
;
; Change state of a sprite, setting bit 1 of state = dying
; expects x = index of sprite
D70E     A9 02          LDA #$02
D710     1D 91 1F       ORA SPRITE_STATE_$1F91,X
D713     9D 91 1F       STA SPRITE_STATE_$1F91,X
D716     A9 00          LDA #$00
```

```
D718      9D 7D 1B       STA MTTBL,X
D71B      60             RTS
;
; Reduce enemy count and kill an object
; expects x = index of object to kill
;
D71C      C6 C9          DEC CRELEFT                             ;Decrement "number of enemies
on screen"
                                                                        : counter
;
; Marks a given object as truly dead
;
D71E      A9 00          LDA #$00
D720      9D 91 1F       STA SPRITE_STATE_$1F91,X
D723      A5 EF          LDA $EF
D725      86 EF          STX $EF
D727      9D 30 1D       STA MISCTBL_$1D30,X
D72A      20 AF E1       JSR $E1AF
D72D      A9 00          LDA #$00
D72F      9D 8C 1E       STA SPRITE_TYPE_$1E8C,X
D732      60             RTS

*
********************************************************************************
*                                                                              *
*         MCSHOOT -- MOVE MC SHOTS, CHECK FOR ANY COLLISIONS                   *
*                    ALSO ADD A NEW SHOT IN DIRECTION OF FIRE CONTROL          *
*                      IF LESS THAN 4 ARE OUT AND PLAYER IS FIRING             *
*               NOTE THAT SHOTS USE 4-BIT DIR CODE, WITH 0 IN SDIRTBL = NULL   *
*                                                                              *
********************************************************************************
*
MCSHOOT:
D733      A2 00          LDX $00                                 ;INITIALIZE X TO START AT
FIRST SHOT
                                                                       ;X POINTS TO SHOTS 0 - 3
IN THIS LOOP
D735      CE 30 1D       DEC MISCTBL_$1D30                       ;DECREMENT MC SHOT TIMER
D738      10 03          BPL MCSLOOP_$D73D                       ;IF IT DOESN'T GO TO 0, START
LOOPING
D73A      8E 30 1D       STX MISCTBL_$1D30                       ;IF HE IS READY TO SHOOT, KEEP
                                                                       ;  MISCTBL (MCSTMR?) AT
ZERO
MCSLOOP:
D73D      8A             TXA
D73E      C9 04          CMP #$04                                ;DONE MOVING SHOTS?
D740      30 03          BMI MCSL1_$D745
D742      4C 9D D7       JMP $D79D                               ;YES, IF Y IS PAST 4TH SHOT

MCSL1:
D745      A9 00          LDA #$00
D747      85 AE          STA TEMP14
D749      BD 27 1C       LDA SDIRTBL,X                           ;LOAD DIRECTION OF THIS SHOT
D74C      D0 17          BNE MCSGO_$D765                         ;IF DIR NOT 00, DON'T ADD
ANOTHER SHOT
D74E      AD 30 1D       LDA MISCTBL_$1D30
D751      F0 03          BEQ $D756
D753      4C 99 D7       JMP $D799

MCSL2:
D756      A9 07          LDA #MCSDELAY                           ;ADDING A NEW SHOT, SO RESET
TIMER
D758      8D 30 1D       STA MCSTMR_$1D30
D75B      20 47 D9       JSR $D947
D75E      A5 AE          LDA TEMP14
D760      F0 37          BEQ $D799
D762      4C 68 D7       JMP MCS1_$D768                          ;IF NOT ZERO, CAN'T ADD A SHOT

MCSGO:
D765      20 AA D9       JSR CHKSHOT                             ;ROUTINE USES X AND SHOT
TABLES

MCS1:
```

```
D768      20 B4 DA       JSR MOVESHOT                                ;MOVE SHOT POS'S AND EXTENTS,
REMOVE IF
                                                                    ; OFFSCREEN

********************************************************************************
*         NOW JSR TO A ROUTINE WHICH UPDATES THE SHOT IN DL AND ZONOBJC ********
*         X POINTS TO SHOT IN SHOT TABLES                               ********
*         NEW SHOT POSITIONS AND EXTENTS IN SHOT DATA TABLES           *********
*         IF SDIRTBL,X IF ZERO, AND SHOT HAS NO DL ENTRY, LEAVE IT ALONE.*******
*         IF SDIRTBL,X IS ZERO, AND SHOT HAS SPACE IN DL, REMOVE IT. ***********
*
*         SHOT DATA SHOULD BE UNCHANGED IN SHOT DATA TABLES
*
*         NOW ADVANCE POINTER INTO SHOT TABLES
*
D76B      8A             TXA
D76C      48             PHA
D76D      18             CLC
D76E      69 53          ADC #$53
D770      AA             TAX
D771      BD CF 1A       LDA SPRITE_X,X
D774      85 BE          STA XINTEND_BE
D776      BD E3 1E       LDA SPRITE_X_EXTENT,X
D779      85 C0          STA XXINTEND_C0
D77B      BD 26 1B       LDA SPRITE_Y,X
D77E      85 BF          STA YINTEND_BF
D780      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
D783      85 C1          STA YYINTEND_C1
D785      A5 AE          LDA TEMP14
D787      F0 0B          BEQ $D794
D789      BD 91 1F       LDA SPRITE_STATE_$1F91,X
D78C      F0 09          BEQ $D797
D78E      20 36 E1       JSR $E136
D791      4C 97 D7       JMP $D797
D794      20 AF E1       JSR $E1AF

D797      68             PLA
D798      AA             TAX

*         NOW ADVANCE POINTER INTO SHOT TABLES

MCSCONT:
D799      E8             INX                                         ;POINT TO NEXT SHOT
D79A      4C 3D D7       JMP MCSLOOP_$D73D                           ;ON TO NEXT SHOT

D79D      60             RTS

*
*******************************************************************************
*                                                                             *
*         MCMOV   --  MOVE MUTANT CLONE ACCORDING TO MOVEMENT JOYSTICK        *
*                     ALSO CHECK FOR COLLISIONS                               *
*                                                                             *
*******************************************************************************
*
MCMOV:
D79E      CE 7D 1B       DEC MCMTMR                                  ;CHECK IF TIME TO MOVE
D7A1      10 63          BPL MCMOV1                                  ;SKIP PAST IF TOO EARLY
D7A3      EE 7D 1B       INC MCMTMR                                  ;RESET MCMTMR
D7A6      EE 7D 1B       INC MCMTMR                                  ;NOW, HAVE MC MOVE EVERY FRAME
D7A9      18             CLC
D7AA      A5 E3          LDA $E3
D7AC      F0 03          BEQ $D7B1
D7AE      4C 03 F9       JMP $F903

D7B1      AD 80 02       LDA SWCHA                                   ;GET MOVE CONTROL
D7B4      29 F0          AND #MASKH                                  ;GET CORRECT BITS
D7B6      4A             LSR A
D7B7      4A             LSR A
D7B8      4A             LSR A
D7B9      4A             LSR A                                       ;GET IT INTO BOTTOM BITS
D7BA      C9 0F          CMP #$0F                                    ;WILL BE $F IF MC ISN'T MOVING
D7BC      F0 48          BEQ MCMOV1                                  ;SKIP MOVING MC
```

```
*  MC IS MOVING THIS FRAME
D7BE      8D B4 1B       STA MCDIR                                       ;PUT AWAY MOVEMENT DIR FROM
JOYSTICK

*  CHANGE ANIMATION STEP
D7C1      CE D9 1C       DEC MCSA                                        ;DECREMENT STEP
D7C4      10 05          BPL MCMOV2                                      ;BRANCH IF STILL A VALID STEP
D7C6      A9 03          LDA #$03                                        ;RESET ANIMATION - LOAD
HIGHEST ANIMATION STEP
D7C8      8D D9 1C       STA MCSA

MCMOV2:
D7CB      AD D4 1B       LDA MCDIR                                       ;RESTORE MOVEMENT DIRECTION

*  COMPUTE CHANGES IN MC'S POSITION
D7CE      AA             TAX                                             ;PUT IT IN X TO INDEX
D7CF      BD 3D EC       LDA XDIRTBL4,X                                  ;RETURNS 0, +STEP OR -STEP
FROM 4-BITS
D7D2      6D CF 1A       ADC MCXPOS                                      ;MOVE 1 STEP
D7D5      85 BE          STA MCXPOS                                      ;STORE NEW POSITION
D7D7      18             CLC
D7D8      69 05          ADC #MCWID
D7DA      85 C0          STA MCXEX                                       ;STORE NEW X EXTENT
D7DC      BD 4D EC       LDA YDIRTBL4,X                                  ;RETURN CHANGE IN Y POSITION
D7DF      18             CLC
D7E0      6D 26 1B       ADC MCYPOS                                      ;MOVE 1 STEP IN  Y DIRECTION
D7E3      85 BF          STA MCYPOS                                      ; STORE NEW Y POSITION
D7E5      18             CLC
D7E6      69 0B          ADC #MCHEIGHT
D7E8      85 C1          STA MCYEX                                       ;STORE NEW Y EXTENT
D7EA      A2 00          LDX #$00                                        ;LOOK AT MC ENTRIES IN OBJECT
TABLES
D7EC      20 B4 D1       JSR CHKOBJBD                                    ;CHECK IF OFF SCREEN

;Disassembly of $D7EF-$D803 compliments of Dan Boris & "Scotty"
D7EF      20 AF E1       JSR $E1AF
D7F2      A5 BE          LDA XINTEND_BE
D7F4      8D CF 1A       STA SPRITE_X                                    ;X POSITION OF PLAYER
D7F7      A5 C0          LDA XXINTEND_C0
D7F9      8D E3 1E       STA SPRITE_X_EXTENT
D7FC      A5 BF          LDA YINTEND_BF
D7FE      8D 26 1B       STA SPRITE_Y                                    ;Y POSITION OF PLAYER
D801      A5 C1          LDA YYINTEND_C1
D803      8D 3A 1F       STA SPRITE_Y_EXTENT

MCMOV1:                  ;LOOK FOR MC COLLIDING WITH AN OBJECT
D806      C6 C8          DEC MCCTMR                                      ;DECREMENT COLLISION TIMER
D808      30 03          BMI MCMOV11                                     ;MUST CHECK COLLISIONS NOW
D80A      4C 46 D9       JMP MCOK                                        ;SKIP COLLISIONS THIS FRAME

MCMOV11:
D80D      A9 02          LDA #$02
D80F      85 C8          STA MCCTMR                                      ;RESET COLLISION TIMER

*          CHECK FOR COLLISIONS

***********  HERE NEED TO CHECK THROUGH ELECTRODES FOR COLLISIONS  *********

*          SEARCH THROUGH OBJECTS IN MC'S ZONE
*  SET UP TEMP0 AND TEMP1 TO BE THE ADDRESS OF THE MC'S ZONE'S ENTRY IN ZONOBJC
*  (THE FIRST IF MC IS IN 2 ZONES).  THEN SET Y TO 27 (OR 55 IF MC
*  IS IN 2 ZONES)  AND LOOP THRU ZONOBJC, DECREMENTING Y UNTIL IT GOES TO 0.

D811      A0 1B          LDY #27                                         ;SET UP Y FOR 1-ZONE CASE

*          COMPUTE MC'S FIRST ZONE, CHECK IF IN 1 OR 2 ZONES, SET UP Y
D813      AD 26 1B       LDA MCYPOS                                      ;LOAD MC Y POSITION
D816      4A             LSR A
D817      4A             LSR A
D818      4A             LSR A
D819      4A             LSR A                                           ;GET ZONE # FROM Y POSITION
D81A      85 C2          STA TEMPZON
```

```
*         CHECK IF IN 2 ZONES
D81C      AD 3A 1F       LDA MCYEX                                   ;Y POSITION OF MC LOWER EDGE
D81F      4A             LSR A
D820      4A             LSR A
D821      4A             LSR A
D822      4A             LSR A                                       ;GET ZONE # OF MC'S LOWER EDGE
D823      C5 C2          CMP TEMPZON                                 ;IS IT SAME AS MC TOP ZONE?
D825      F0 02          BEQ MCC1                                    ;MC IN ONLY 1 ZONE, LEAVE Y AS
31
D827      A0 37          LDY #55                                     ;LOAD Y WITH 55 - WE WILL INDEX
                                                                     ;  THRU 2 ZONES

*         GET ABSOLUTE ADDRESS OF START OF MC'S ZONE IN ZONOBJC
*                 ZONE NUMBER IS IN TEMPZON

MCC1:                                                                ;NOW Y IS SET UP
D829      84 A2          STY TEMP2                                   ;PUT Y AWAY FOR NOW
D82B      A4 C2          LDY TEMPZON                                 ;ZONE NUMBER WILL BE USED TO
INDEX
D82D      B9 EC F1       LDA ZONOBJLH,Y                              ;GET HIGH BYTE OF ABS ADDRESS
D830      85 BD          STA TEMP1
D832      B9 E0 F1       LDA ZONOBJLL,Y                              ;GET LOW BYTE OF ABS ADDRESS
D835      85 BC          STA TEMP0
D837      A4 A2          LDY TEMP2                                   ;RECOVER Y - IT WILL INDEX THRU
ZONOBJC

*         NOW WE HAVE THE ABSOLUTE ADDRESS OF THE ZONOBJC LISTING
*                 FOR THE MC'S (FIRST) ZONE IN TEMP0 AND TEMP1

MCCLOOP:
D839      B1 BC          LDA (TEMP0),Y                               ;GET OBJECT # FROM CORRECT
PART OF ZONOBJC
D83B      F0 3E          BEQ MCCNEXT                                 ;IF 0, A NULL ENTRY
D83D      AA             TAX                                         ;PUT OBJECT INDEX IN X
D83E      BD 8C 1E       LDA CRTBL,X                                 ;LOAD CREATURE TYPE
D841      29 1F          AND #$1F
D843      F0 36          BEQ MCCNEXT                                 ;IF THIS IS 0, ALSO A NULL
OBJECT
D845      C9 0F          CMP #MCSCODE                                ;CHECK IF AN MC SHOT
D847      F0 32          BEQ MCCNEXT                                 ;MC CAN'T HIT HIS OWN SHOT
D849      BD 91 1F       LDA STTBL,X
D84C      29 03          AND #$03
D84E      F0 2B          BEQ MCCNEXT                                 ;CAN'T HIT DEAD OBJECT
D850      29 0F          AND #MASKL
D852      C9 03          CMP #$03
D854      F0 25          BEQ MCCNEXT                                 ;CAN'T HIT DYING OBJECT

*         CHECK FOR A COLLISION WITH THE OBJECT POINTED TO BY X.
*                 IF A FAMILY, HANDLE IT AND THEN CONTINUE
*         IF A COLLISION, THE MC BITES IT
*
*         MC X,Y ARE IN MCXPOS AND MCYPOS
*         MC X,Y EXTENTS ARE IN MCXEX AND MCYEX
*         OBJECT X,Y ARE IN XTBL,X AND YTBL,X
*         OBJECT EXTENTS ARE IN XEXTBL,X AND YEXTBL,X
*                  IF A MISS, BRANCH TO MCCNEXT.  IF A HIT, FALL THROUGH…
*
D856      18             CLC
D857      AD E3 1E       LDA MCXEX
D85A      DD CF 1A       CMP XTBL,X
D85D      90 1C          BCC MCCNEXT                                 ;BRANCH IF LESS THAN
D85F      AD CF 1A       LDA MCXPOS
D862      DD E3 1E       CMP XEXTBL,X
D865      F0 02          BEQ $D869
D867      B0 12          BCS MCCNEXT                                 ;BRANCH IF > OR =
D869      AD 3A 1F       LDA MCYEX
D86C      DD 26 1B       CMP YTBL,X
D86F      90 0A          BCC MCCNEXT                                 ;BRANCH ON LESS THAN
D871      AD 26 1B       LDA MCYPOS
D874      DD 3A 1F       CMP YEXTBL,X
D877      90 05          BCC $D87E                                   ;BRANCH ON GREATER THAN
D879      F0 03          BEQ $D87E
```

```
D87B      4C 40 D9       JMP $D940

*         HIT!    CHECK IF WITH FAMILY

D87E      BD 8C 1E       LDA CRTBL,X                                 ;GET CREATURE TYPE
D881      29 1F          AND #$1F
D883      C9 02          CMP #MOCODE
D885      F0 08          BEQ MCCF
D887      C9 03          CMP #DCODE
D889      F0 04          BEQ MCCF
D88B      C9 04          CMP #MICODE
D88D      D0 22          BNE MCC2

MCCF:              ;MC COLLIDED WITH A FAMILY MEMBER
*         NOW:  ENTER PICKUP SOUND INTO QUEUE.  UPDATE SCORE AND SET FAMILY
*    ANIMATION TO THE CORRECT NUMBER (1,2,3,4 OR 5) WITH THE HIGH BIT SET
*    SET THE OBJECT CODE TO #MOCODE.  SET THE DYING BIT IN STTBL.
*    SET THE OBJECT DIRECTION CODE TO 8.  FINALLY, RESET MTTBL.

*         HERE ENTER PICKUP SOUND INTO SOUND QUEUE                      ********
*         HERE CALL THE ROUTINE TO UPDATE SCORE - IT SHOULD RETURN WITH
*                 THE CORRECT NUMBER ANIMATION IN A  (1 - 5 FOR 1000 - 5000 )

D88F      20 3E D4       JSR FSCORE                                  ;THIS RETURNS WITH 1 - 5 IN A
D892      09 80          ORA #$80                                    ;SET HIGH BIT
D894      9D D9 1C       STA SATBL,X                                 ;STORE NEW ANIMATION STEP
D897      A9 02          LDA #MOCODE
D899      9D 8C 1E       STA CRTBL,X                                 ;STORE NEW CREATURE TYPE
D89C      A9 02          LDA #$02                                    ;BIT 1 IS ON
D89E      1D 91 1F       ORA STTBL,X
D8A1      9D 91 1F       STA STTBL,X                                 ;SET BIT 1 IN STATUS ENTRY
D8A4      A9 08          LDA #$08
D8A6      9D D4 1B       STA DXTBL,X                                 ;SET DIR CODE
D8A9      A9 00          LDA #$00
D8AB      9D 7D 1B       STA MTTBL,X                                 ;RESET MOVEMENT TIMER
D8AE      4C 40 D9       JMP MCCNEXT                                 ;CONTINUE COLLISION CHECKING

MCC2:             ;MC DIDN'T HIT A FAMILY, SO HE HIT SOMETHING HE DIDN'T WANT TO
;Disassembly of $D8B1-$D951 compliments of Dan Boris & "Scotty"
D8B1      C9 0D          CMP #$0D
D8B3      D0 3B          BNE $D8F0
D8B5      BD 30 1D       LDA MISCTBL_$1D30,X
D8B8      85 B7          STA FRMCNT
D8BA      4A             LSR A
D8BB      4A             LSR A
D8BC      4A             LSR A
D8BD      4A             LSR A                                       ; * 16
D8BE      18             CLC
D8BF      7D CF 1A       ADC SPRITE_X,X
D8C2      85 AA          STA TEMP10
D8C4      85 AB          STA TEMP11
D8C6      A5 B7          LDA FRMCNT
D8C8      29 0F          AND #$0F
D8CA      18             CLC
D8CB      7D 26 1B       ADC SPRITE_Y,X
D8CE      85 AC          STA TEMP12
D8D0      69 01          ADC #$01
D8D2      85 AD          STA TEMP13
D8D4      AD CF 1A       LDA SPRITE_X
D8D7      85 B8          STA TEMPX
D8D9      AD 26 1B       LDA SPRITE_Y
D8DC      85 B9          STA TEMPY
D8DE      AD E3 1E       LDA SPRITE_X_EXTENT
D8E1      85 A4          STA TEMP4
D8E3      AD 3A 1F       LDA SPRITE_Y_EXTENT
D8E6      85 A5          STA TEMP5
D8E8      20 F2 DB       JSR $DBF2
D8EB      F0 53          BEQ $D940
D8ED      4C 11 D0       JMP MCDEATH                                 ;MC DEATH ROUTINE

D8F0      BD CF 1A       LDA SPRITE_X,X
D8F3      85 AA          STA TEMP10
D8F5      BD E3 1E       LDA SPRITE_X_EXTENT,X
```

```
D8F8      85 AB          STA TEMP11
D8FA      BD 26 1B       LDA SPRITE_Y,X
D8FD      85 AC          STA TEMP12
D8FF      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
D902      85 AD          STA TEMP13
D904      86 AE          STX TEMP14                                 ;Save current object index in
$AE as
                                                                    ; check overlap function
expects it
D906      A2 00          LDX $00                                    ;Compare with player's index
D908      20 13 DC       JSR $DC13                                  ;Are player's sprites and
enemy sprites
                                                                    ; colliding?
D90B      F0 33          BEQ $D940                                  ;No, the Z flag is set
D90D      A6 AE          LDX TEMP14                                 ;Restore object index
D90F      BD 8C 1E       LDA CRTBL,X
D912      C9 06          CMP #$06                                   ;Spheroid?
D914      F0 07          BEQ $D91D
D916      C9 07          CMP #$07                                   ;Quark?
D918      F0 07          BEQ $D921
D91A      4C 11 D0       JMP $D011
D91D      A9 03          LDA #$03                                   ;Frame 3
D91F      D0 02          BNE $D923
D921      A9 06          LDA #$06                                   ;Frame 6
D923      9D D9 1C       STA SATBL,X
D926      BD CF 1A       LDA SPRITE_X,X
D929      85 BE          STA XINTEND_BE
D92B      BD 26 1B       LDA SPRITE_Y,X
D92E      85 BF          STA YINTEND_BF
D930      BD E3 1E       LDA SPRITE_X_EXTENT,X
D933      85 C0          STA XXINTEND_C0
D935      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
D938      85 C1          STA YYINTEND_C1
D93A      20 AF E1       JSR $E1AF
D93D      4C 11 D0       JMP $D011

D940      88             DEY
D941      30 03          BMI $D946
D943      4C 39 D8       JMP $D839
D946      60             RTS

D947      A9 00          LDA #$00
D949      85 AE          STA TEMP14
D94B      A5 E3          LDA $E3
D94D      F0 03          BEQ $D952
D94F      4C BB F8       JMP $F8BB

*
****************************************
*                                      *
*     SUBROUTINES USED BY MCSHOOT:     *
*                                      *
****************************************

*************************************************************************************
*                                                                                   *
*         CRESHOT -- ROUTINE TO CREATE (POSSIBLY) A NEW SHOT                        *
*                                                                                   *
*         GIVEN SHOT NUMBER IN X, PRESERVES X                                       *
*         SDIRTBL,X SHOULD BE 0 FOR THIS TO BE CALLED - A NULL SHOT DOES EXIST      *
*         IF A NEW SHOT IS CALLED FOR, SETS EACH SHOT TABLE,X                       *
*         ENDS WITH RTS, AFTER SETTING UP SHOT TABLES FOR SHOT NUMBER X             *
*                                                                                   *
*************************************************************************************
*
CRESHOT:
D952      AD 80 02       LDA $0280                                  ;LOOK AT JOYSTICK TO FIND
DIRECTION
D955      29 0F          AND #MASKL                                 ;MASK OFF RIGHT JOYSTICK
D957      C9 0F          CMP #$0F                                   ;PUSHED IN ANY DIRECTION?
(WHEN EQUAL TO #$0F = NO)
D959      D0 17          BNE CRESHOT1                               ;BRANCH IF FIRING
D95B      A5 0C          LDA $0C                                          ;READ TRIGGER
```

```
D95D      29 80          AND #$80                                    ;
D95F      D0 10          BNE $D971                                   ;BRANCH IF NOT PUSHED
D961      AD 80 02       LDA $0280                                   ;READ JOYSTICK
D964      29 F0          AND #$F0                                    ;MASK OFF LEFT JOYSTICK
D966      C9 F0          CMP #$F0                                    ;PUSHED IN ANY DIRECTION?
D968      F0 07          BEQ $D971                                   ;BRANCH IF NOT
D96A      4A             LSR A                                             ;SHIFT TO BOTTOM
D96B      4A             LSR A                                             ;
D96C      4A             LSR A                                             ;
D96D      4A             LSR A                                             ;
D96E      4C 72 D9       JMP CRESHOT1                                ;
D971      60             RTS                                               ;LEAVE THIS SHOT NULL,
CONTINUE
                                    ; SHOT MOVE ROUTINE WILL NOT MOVE THIS SINCE IT IS NULL

;PLAYER FIRED, BOTTOM 4 BITS OF THE ACCUMULATOR CONTAIN DIRECTION
CRESHOT1:
D972      9D 27 1C       STA SDIRTBL,X                               ;SAVE JOYSTICK POSITION
D975      AD A0 FC       LDA CRETODST+$0F                            ;BYPASS GETSTAMPS FOR SPEED. F
IS THE TYPE
D978      18             CLC                                         ;THAT LOAD JUST GOT THE TYPE
POINTER
D979      7D 27 1C       ADC SDIRTBL,X                               ;ADD IN THE DIRECTION FOR
POINTERS TO EXTENTS
D97C      A8             TAY
D97D      AD CF 1A       LDA MCXPOS                                  ;GET MC'S HORIZONTAL POSITION
D980      18             CLC
D981      69 02          ADC #MCWID/2                                ;COMPUTE CENTER OF MC'S BODY
D983      9D 22 1B       STA SXTBL,X                                 ;STORE SHOT XPOS = CENTER OF MC
D986      18             CLC
D987      79 AD FE       ADC #MCHEIGHT/2                             ;COMPUTE CENTER OF MC'S BODY
D98A      9D 36 1F       STA SYTBL,X                                 ;STORE SHOT YPOS = CENTER OF MC
D98D      AD 26 1B       LDA MCYPOS                                  ;GET MC'S VERTICAL POSITION
D990      18             CLC
D991      69 05          ADC #$05
D993      9D 79 1B       STA SYTBL,X
D996      18             CLC
D997      79 F9 FD       ADC STAMPHGH,Y
D99A      9D 8D 1F       STA SYEXTBL,X
D99D      A9 01          LDA #$01
D99F      9D E4 1F       STA $1FE4,X

          ; NOW SHOT EXISTS, AND HAS THE DIRECTION OF THE MC'S FIRING CONTROL
*         START MC SHOOTING SOUND

D9A2      A9 00          LDA #SMCS                                   ;Play Mutant Clone Shooting
Sound
D9A4      20 95 E3       JSR DOTUNE_$E395                            ;
D9A7      E6 AE          INC TEMP14
D9A9      60             RTS                                         ;NOW RETURN, AND THIS SHOT
SHOULD THEN BE MOVED

*********************************************************************************
*                                                                               *
*         CHKSHOT -- CHECK FOR COLLISIONS VERSUS ONE SHOT                       *
*                 SHOT NUMBER GIVEN IN X, USE SHOT DATA TABLES                  *
*                 END WITH AN RTS, PRESERVE X                                   *
*                 IF NO COLLISION, DON'T CHANGE ANYTHING                        *
*                 IF A COLLISION, REMOVE SHOT BY NULLING OUT SDIRTBL,X          *
*                   AND MODIFY OBJECT DATA TABLES TO SHOW CREATURE              *
*                   WHICH WAS HIT AS DYING                                      *
*                                                                               *
*********************************************************************************
*
CHKSHOT
*         GET SHOT POSITION AND EXTENTS

D9AA      BD 22 1B       LDA SXTBL,X                                 ;GET SHOT XPOS
D9AD      85 B8          STA TEMPX
D9AF      BD 36 1F       LDA SXEXTBL,X                               ;GET SHOT X EXTENT
D9B2      85 A4          STA TEMP4
D9B4      BD 79 1B       LDA SYTBL,X                                 ;GET SHOT YPOS
D9B7      85 B9          STA TEMPY
```

```
D9B9      BD 8D 1F       LDA SYEXTBL,X                          ;GET SHOT Y EXTENT
D9BC      85 A5          STA TEMP5

*          LOOP THROUGH OBJECTS TO CHECK FOR COLLISIONS WITH A SHOT
*          SEARCH THROUGH OBJECTS IN THE SHOT'S ZONE
* SET UP TEMP0 AND TEMP1 TO BE THE ADDRESS OF THE SHOT'S ZONE'S ENTRY IN ZONOBJC
*          (THE FIRST IF SHOT IS IN 2 ZONES).  THEN SET Y TO 27 (OR 55 IF SHOT
* IS IN 2 ZONES)  AND LOOP THRU ZONOBJC, DECREMENTING Y UNTIL IT GOES TO 0.

D9BE      A0 1B          LDY #27                                ;SET UP Y FOR 1-ZONE CASE

*    COMPUTE SHOT'S FIRST ZONE, CHECK IF 1 OR 2 ZONES, SET UP Y
*
*          CHECK IF IN 2 ZONES

D9C0      BD 79 1B       LDA SYTBL,X                            ;LOAD SHOT Y POSITION
D9C3      4A             LSR A
D9C4      4A             LSR A
D9C5      4A             LSR A
D9C6      4A             LSR A                                  ;GET ZONE # FROM Y POSITION
D9C7      85 C2          STA TEMPZON
D9C9      BD 8D 1F       LDA SYEXTBL,X                          ;Y POSITION OF SHOT LOWER EDGE
D9CC      4A             LSR A
D9CD      4A             LSR A
D9CE      4A             LSR A
D9CF      4A             LSR A                                  ;GET ZONE # OF SHOT'S LOWER
EDGE
D9D0      C5 C2          CMP TEMPZON                            ;IS IT SAME AS SHOT TOP ZONE?
D9D2      F0 02          BEQ MCSC1                              ;SHOT IN ONLY 1 ZONE, LEAVE Y
AS 31
D9D4      A0 37          LDY #55                                ;LOAD Y WITH 63 - WE WILL INDEX
THRU 2 ZONES

MCSC1:             ;NOW Y IS SET UP
*          GET ABSOLUTE ADDRESS OF START OF SHOT'S ZONE IN ZONOBJC
*                  ZONE NUMBER IS IN TEMPZON
D9D6      84 A2          STY TEMP2                              ;PUT Y AWAY FOR NOW
D9D8      A4 C2          LDY TEMPZON
D9DA      B9 EC F1       LDA ZONOBJLH,Y                         ;GET HIGH BYTE
D9DD      85 A1          STA TEMP1
D9DF      B9 E0 F1       LDA ZONOBJLL,Y                         ;GET LOW BYTE OF ABS ADDRESS
D9E2      85 A0          STA TEMP0
D9E4      A4 A2          LDY TEMP2                              ;RECOVER Y - IT IS INDEX THRU
ZONOBJC

*          NOW WE HAVE THE ABSOLUTE ADDRESS OF THE ZONOBJC LISTING
*                  FOR THE SHOT'S (FIRST) ZONE IN TEMP0 AND TEMP1
*
*          SAVE X  -  IT CURRENTLY POINTS TO CURRENT SHOT IN SHOT TABLES

D9E6      86 A8          STX TEMP8

MCSCLOOP:
D9E8      B1 A0          LDA (TEMP0),Y                          ;GET OBJECT NUMBER FROM
ZONOBJC
D9EA      F0 6C          BEQ MCSCNEXT                           ;IF 0, A NULL ENTRY
D9EC      AA             TAX                                    ;PUT OBJECT NUMBER IN X
D9ED      BD 8C 1E       LDA CRTBL,X                            ;GET CREATURE CODE
D9F0      29 1F          AND #$1F
D9F2      F0 64          BEQ MCSCNEXT                           ;IF THIS IS 0, ALSO A NULL
OBJECT
D9F4      C9 0F          CMP #MCSCODE                           ;IS IT AN MC SHOT?
D9F6      F0 60          BEQ MCSCNEXT                           ;SHOT CAN'T COLLIDE WITH A
SHOT
D9F8      BD 91 1F       LDA STTBL,X
D9FB      29 03          AND #MASKL
D9FD      F0 59          BEQ MCSCNEXT                           ;DON'T COLLIDE WITH A DEAD
OBJECT
D9FF      C9 03          CMP #$03                               ;CODE IN STTBL FOR 'DYING'
OBJECT
DA01      F0 55          BEQ MCSCNEXT                           ;DON'T COLLIDE WIT

;Disassembly of $DA03-$DA59 compliments of Dan Boris & "Scotty"
```

```
DA03      BD 8C 1E       LDA CRTBL,X
DA06      29 1F          AND #$1F
DA08      C9 0D          CMP #$0D
DA0A      D0 26          BNE $DA32
DA0C      BD 30 1D       LDA MISCTBL_$1D30,X
DA0F      85 B7          STA FRMCNT
DA11      4A             LSR A
DA12      4A             LSR A
DA13      4A             LSR A
DA14      4A             LSR A
DA15      18             CLC
DA16      7D CF 1A       ADC SPRITE_X,X
DA19      85 AA          STA TEMP10
DA1B      85 AB          STA TEMP11
DA1D      A5 B7          LDA FRMCNT
DA1F      29 0F          AND #$0F
DA21      18             CLC
DA22      7D 26 1B       ADC SPRITE_Y,X
DA25      85 AC          STA TEMP12
DA27      69 01          ADC #$01
DA29      85 AD          STA TEMP13
DA2B      20 F2 DB       JSR $DBF2
DA2E      F0 28          BEQ $DA58
DA30      D0 1C          BNE $DA4E
DA32      A5 A4          LDA TEMP4
DA34      DD CF 1A       CMP SPRITE_X,X
DA37      90 1F          BCC $DA58
DA39      BD E3 1E       LDA SPRITE_X_EXTENT,X
DA3C      C5 B8          CMP TEMPX
DA3E      90 18          BCC $DA58
DA40      A5 A5          LDA TEMP5
DA42      DD 26 1B       CMP SPRITE_Y,X
DA45      90 11          BCC $DA58
DA47      BD 3A 1F       LDA SPRITE_Y_EXTENT,X
DA4A      C5 B9          CMP TEMPY
DA4C      90 0A          BCC $DA58
DA4E      20 5E DA       JSR MCSHIT
DA51      A6 A8          LDX TEMP8
DA53      BD 27 1C       LDA SHOT_DIR_TBL_$1C27,X
DA56      F0 05          BEQ $DA5D
DA58      A6 A8          LDX TEMP8

*         CONTINUE HERE IF NO COLLISION WITH CURRENT OBJECT
MCCNEXT:
DA5A      88             DEY                                         ;ON TO NEXT ENTRY IN ZONE LIST
DA5B      10 8B          BPL MCCLOOP                                 ;IF Y NON-NEGATIVE, KEEP GOING

*         WE HAVE CHECKED EVERYTHING IN THE MC'S ZONE(S)
MCOK:                                                                ;NO COLLISION WITH MC THIS
TIME
DA5D      60             RTS                                         ;MC MAKES IT THROUGH YET
ANOTHER FRAME

*********************************************************************************
*                                                                               *
*         MCSHIT  -  MC SHOT HIT SOMETHING                                      *
*         A COLLISION!  REPLACE OBJECT WITH NULL OBJECT                         *
*                    AND SHOT WITH A NULL SHOT IF A LEGITIMATE COLLISION.       *
*         THE SHOT CAN BE FOUND FROM TEMP8.  THE OBJECT THAT WAS HIT IS         *
*          INDEXED BY X.  DO AN STZ CRTBL,X TO ZERO THE OBJECT CODE.            *
*                                                                               *
*********************************************************************************
*
MCSHIT:
DA5E      BD 8C 1E       LDA CRTBL,X                                 ;GET OBJECT CODE THAT SHOT HIT

*         CHECK IF COLLISION WITH FAMILY.  IF SO, PRETEND AS IF NO COLLISION

DA61      29 1F          AND #$1F
DA63      C9 02          CMP #MOCODE                                 ;IS IT MOMMY?
DA65      F0 46          BEQ MCSHITNO_$DAB3                          ;SHOT GOES THROUGH HER
DA67      C9 03          CMP #DCODE                                  ;IS IT DADDY?
DA69      F0 42          BEQ MCSHITNO_$DAB3                          ;SHOT GOES THROUGH HIM
```

```
DA6B      C9 04          CMP #MICODE                                   ;IS IT MIKEY?
DA6D      F0 3E          BEQ MCSHITNO_$DAB3                            ;SHOT GOES THROUGH THE LITTLE
TWERP

*          SOMETHING WAS REALLY SHOT...
*          FIRST REMOVE MC SHOT WHICH CAUSED THIS
DA6F      84 A9          STY TEMP9                                     ;TEMP9 NOW HOLDS INDEX INTO
ZONOBJC
DA71      A4 A8          LDY TEMP8                                     ;SHOT NUMBER WAS SAVED HERE
DA73      B9 27 1C       LDA SDIRTBL,Y
DA76      85 A0          STA TEMP0                                     ;PUT DIR AWAY IN CASE A HULK IS
HIT
DA78      A9 00          LDA #$00
DA7A      99 27 1C       STA SDIRTBL,Y                                 ;ZERO THIS SHOT'S DIRECTION
                                                                       ;A NEW SHOT WILL SOON BE ADDED
DA7D      99 E4 1F       STA $1FE4,Y

*          UPDATE SCORE AND REMOVE OBJECT IF OBJECT IS DESTRUCTIBLE
DA80      BD 8C 1E       LDA CRTBL,X                                   ;GET CREATURE OBJECT CODE
DA83      29 1F          AND #$1F
DA85      C9 05          CMP #HCODE                                    ;IS IT A HULK?
DA87      F0 08          BNE KILLOBJ                                   ;IF NOT, GET RID OF THE OBJECT

*          ITS A HULK    ;HULK WAS SHOT
DA89      C9 10          CMP #$10
DA8B      D0 21          BPL $DAA2
DA8D      20 0E D7       JSR $D70E
DA90      60             RTS

;Disassembly of $DA91-$D9FF compliments of Dan Boris & "Scotty"
KILLOBJ:
DA91      38             SEC
DA92      A5 E5          LDA $E5
DA94      FD 2B 1C       SBC SPRITE_DELTA_Y_$1C2B,X
DA97      C9 08          CMP #$08
DA99      90 16          BCC $DAB1
DA9B      A5 E5          LDA $E5
DA9D      9D 2B 1C       STA SPRITE_DELTA_Y_$1C2B,X

*          SET OBJECT TO MOVE NEXT FRAME SO IT CAN START DYING
DAA0      A5 A0          LDA TEMP0                                     ;LOAD ORIGINAL DIR OF CURRENT
SHOT
DAA2      9D 30 1D       STA MISCTBL,X                                 ;SET HULK'S DIR-TO-JUMP TO
SHOT DIR
DAA5      A9 00          LDA #$00
DAA7      9D 7D 1B       STA MTTBL,X                                   ;FORCE HULK TO MOVE NEXT FRAME
DAAA      4C B1 DA       JMP MCSHIT1_$DAB1                             ;ON TO NEXT SHOT
DAAD      60             RTS

DAAE      20 DB D6       JSR SCORING_$D6DB

*          HIT ROUTINE IS FINISHED.  WE MAY HAVE ZEROED THE CURRENT SHOT DIR

MCSHIT1:
DAB1      A4 A9          LDY TEMP9                                     ;RESTORE ZONOBJC TO Y
                  ; CODE AFTER THE JSR TO MCSHIT WILL RESTORE X AS SHOT INDEX

MCSHITNO:
DAB3      60             RTS

**********************************************************************************
*                                                                                *
*          MOVESHOT  - MOVE A MC SHOT                                            *
*                  GIVEN SHOT NUMBER IN X                                        *
*                  THIS CHANGES THE SHOT DATA IN THE SHOT TABLES                 *
*                  STORE NEW X,Y POSITIONS, THEN COMPUTE AND STORE EXTENTS       *
*                  WILL NULL OUT DIR CODE IF A BOUNDARY IS ENCOUNTERED           *
*                                                                                *
**********************************************************************************
*
MOVESHOT:
DAB4      BC 27 01       LDY SDIRTBL,X                                 ;X HAS DIRECTION CODE
DAB7      D0 01          BNE MOVES0                                    ;MOVE THIS SHOT; IT IS ALIVE
```

```
DAB9      60             RTS                                              ;DON'T MOVE THIS SHOT; IT IS
NULL

MOVES0:
DABA      B9 3D EC       LDA XDIRTBL4,Y                                   ;A HAS 1, 0, OR FF FOR X CHANGE
DABD      F0 32          BEQ MCSY_$DAF1                                   ;DON'T CHANGE X POSITION
DABF      30 18          BMI MCSXMI_DAD9                                  ;BRANCH ON NEGATIVE DELTA X
DAC1      BD 22 1B       LDA SXTBL,X                                      ;LOAD SHOT X POSITION
DAC4      18             CLC
DAC5      69 04          ADC #SHOTSTX                                     ;INCREMENT X POS BY X SHOTSTEP
DAC7      9D 22 1B       STA SXTBL,X
DACA      C9 98          CMP #MAXX-SHOTWID
DACC      B0 5D          BCS SHOTEND                                      ;IF SHOT IS OUT OF BOUNDS
DESTROY IT
DACE      BD 36 1F       LDA SXEXTBL,X                                    ;GET THE EXTENT
DAD1      69 04          ADC #SHOTSTX                                     ;MOVE THE OTHER EDGE
DAD3      9D 36 1F       STA SXEXTBL,X
DAD6      4C F1 DA       JMP MCSY

MCSXMI:
DAD9      BD 22 1B       LDA SXTBL,X                                      ;LOAD SHOT X POSITION
DADC      38             SEC
DADD      E9 04          SBC #SHOTSTX                                     ;DECREMENT X POS BY X SHOTSTEP
DADF      9D 22 1B       STA SXTBL,X                                      ;STORE NEW POS IN TABLES
DAE2      18             CLC
DAE3      69 10          ADC #$10                                         ;ADD FOR BORDER CHECKING
DAE5      C9 12          CMP #MINX+$10
DAE7      90 42          BCC SHOTEND_$DB2B                                ;IF SHOT IS OUT OF BOUNDS
DESTROY IT
DAE9      BD 36 1F       LDA SXEXTBL,X                                    ;GET THE EXTENT
DAEC      E9 04          SBC #SHOTSTX                                     ;MOVE THE OTHER EDGE
DAEE      9D 36 1F       STA SXEXTBL,X

*          NOW HANDLE Y CHANGE
MCSY:
DAF1      B9 4D EC       LDA YDIRTBL4,Y                                   ;GET 0, POSITIVE OR NEG # FOR
DELTA Y
DAF4      F0 3D          BEQ MOVESD_$DB33                                 ;NO Y CHANGE, GO ON AND DO
EXTENTS
DAF6      30 18          BMI MCSYMI_$DB10                                 ;BRANCH IF NEGATIVE DELTA Y
DAF8      BD 79 1B       LDA SYTBL,X                                      ;GET SHOT Y POSITION
DAFB      18             CLC
DAFC      69 08          ADC #SHOTSTY                                     ;INCREMENT Y POS BY Y SHOTSTEP
DAFE      9D 79 1B       STA SYTBL,X
DB01      C9 B4          CMP #MAXY-SHOTHT
DB03      B0 26          BCS SHOTEND                                      ;IF SHOT IS OUT OF BOUNDS
DESTROY IT
DB05      BD 8D 1F       LDA SYEXTBL,X                                    ;GET THE EXTENT
DB08      69 08          ADC #SHOTSTY                                     ;MOVE THE OTHER EDGE
DB0A      9D 8D 1F       STA SYEXTBL,X
DB0D      4C 33 DB       JMP MOVESD

MCSYMI:
DB10      BD 79 1B       LDA SYTBL,X                                      ;LOAD SHOT Y POSITION
DB13      38             SEC
DB14      E9 08          SBC #SHOTSTY                                     ;DECREMENT Y POS BY Y SHOTSTEP
DB16      9D 79 1B       STA SYTBL,X                                      ;SAVE NEW POS IN TABLES
DB19      18             CLC
DB1A      69 10          ADC #$10                                         ;ADD FOR BORDER CHECKING
DB1C      C9 22          CMP #MINY+$10
DB1E      90 0B          BCC SHOTEND_$DB2B
DB20      BD 8D 1F       LDA SYEXTBL,X                                    ;GET THE EXTENT
DB23      E9 08          SBC #SHOTSTY                                     ;MOVE THE OTHER EDGE
DB25      9D 8D 1F       STA SYEXTBL,X
DB28      4C 33 DB       JMP MOVESD                                       ;DONE WITH MOVING SHOT

*          NOTE: IN THE BORDER CHECKING ABOVE IT IS OK TO USE SHOTWID AND SHOTHT
*       INSTEAD OF ACTUAL EXTENTS - SHOTS ARE NEVER PARALLEL TO BORDERS THEY HIT
*
*          A SHOT HIT A WALL  --  REMOVE IT
SHOTEND:
DB2B      A9 00          LDA #$00
DB2D      9D 27 1C       STA SDIRTBL,X                                    ;ZERO ITS DIRECTION
```

```
DB30      9D E4 1F       STA $1FE4,X

*          NOW GO ON, AND THE SHOT LOADING ROUTINE MUST REMOVE THIS SINCE
*                   ITS DIRECTION IS 0.
*
*          CONTINUE AFTER MOVING SHOT...
MOVESD:
DB33      60             RTS

;Disassembly of $DB34-$E394 compliments of Dan Boris & "Scotty"
DB34      A5 E3          LDA $E3
DB36      F0 08          BEQ $DB40
DB38      20 22 F5       JSR $F522
DB3B      F0 03          BEQ $DB40
DB3D      4C 95 F3       JMP $F395

DB40      A5 5E          LDA $5E
DB42      F0 05          BEQ $DB49
DB44      C6 5E          DEC $5E
DB46      4C CF DB       JMP $DBCF

DB49      A5 5D          LDA $5D
DB4B      29 01          AND #$01
DB4D      D0 0E          BNE $DB5D
DB4F      AD 82 02       LDA $0282
DB52      29 01          AND #$01
DB54      F0 07          BEQ $DB5D
DB56      A9 00          LDA #$00
DB58      85 E3          STA $E3
DB5A      4C 03 90       JMP $9003

DB5D      A5 5D          LDA $5D
DB5F      29 02          AND #$02
DB61      D0 0A          BNE $DB6D
DB63      AD 82 02       LDA $0282
DB66      29 02          AND #$02
DB68      F0 03          BEQ $DB6D
DB6A      4C 95 F3       JMP $F395

DB6D      A5 5D          LDA $5D
DB6F      29 08          AND #$08
DB71      D0 57          BNE $DBCA
DB73      AD 82 02       LDA $0282
DB76      29 08          AND #$08
DB78      F0 50          BEQ $DBCA
DB7A      A9 02          LDA #$02
DB7C      85 5E          STA $5E
DB7E      A5 5C          LDA $5C
DB80      49 01          EOR #$01
DB82      C9 03          CMP #$03
DB84      D0 07          BNE $DB8D
DB86      A9 00          LDA #$00
DB88      85 5C          STA $5C
DB8A      4C B1 DB       JMP $DBB1

DB8D      85 5C          STA $5C
DB8F      F0 18          BEQ $DBA9
DB91      A5 E5          LDA $E5
DB93      85 A1          STA TEMP1
DB95      A5 E6          LDA $E6
DB97      85 A2          STA TEMP2
DB99      A9 00          LDA #$00
DB9B      85 E5          STA $E5
DB9D      A9 01          LDA #$01
DB9F      85 E6          STA $E6
DBA1      A9 00          LDA #$00
DBA3      85 19          STA $19
DBA5      85 1A          STA $1A
DBA7      F0 21          BEQ $DBCA
DBA9      A5 5C          LDA $5C
DBAB      F0 15          BEQ $DBC2
DBAD      A9 01          LDA #$01
DBAF      85 5C          STA $5C
```

```
DBB1      20 45 F5       JSR $F545
DBB4      A9 40          LDA #$40                                  ;Enable DMA, RM=0
DBB6      85 3C          STA $3C                                         ;
DBB8      A0 10          LDY $10
DBBA      20 38 F5       JSR $F538
DBBD      88             DEY
DBBE      10 FA          BPL $DBBA
DBC0      F0 08          BEQ $DBCA
DBC2      A5 A1          LDA TEMP1
DBC4      85 E5          STA $E5
DBC6      A5 A2          LDA TEMP2
DBC8      85 E6          STA $E6
DBCA      AD 82 02       LDA $0282
DBCD      85 5D          STA $5D
DBCF      A5 5C          LDA $5C
DBD1      F0 1E          BEQ $DBF1
DBD3      A5 E6          LDA $E6
DBD5      D0 08          BNE $DBDF
DBD7      A9 7F          LDA #$7F                                  ;Disable DMA
DBD9      85 3C          STA $3C                                         ;
DBDB      A9 02          LDA #$02
DBDD      85 5C          STA $5C
DBDF      20 22 F5       JSR $F522
DBE2      D0 03          BNE $DBE7
DBE4      4C 34 DB       JMP $DB34

DBE7      A5 5C          LDA $5C
DBE9      38             SEC
DBEA      E9 01          SBC #$01
DBEC      85 5C          STA $5C
DBEE      4C A9 DB       JMP $DBA9
DBF1      60             RTS

DBF2      A5 AB          LDA TEMP11
DBF4      C5 B8          CMP TEMPX
DBF6      90 16          BCC $DC0E
DBF8      A5 A4          LDA TEMP4
DBFA      C5 AA          CMP TEMP10
DBFC      90 10          BCC $DC0E
DBFE      A5 AD          LDA TEMP13
DC00      C5 B9          CMP TEMPY
DC02      90 0A          BCC $DC0E
DC04      A5 A5          LDA TEMP5
DC06      C5 AC          CMP TEMP12
DC08      90 04          BCC $DC0E
DC0A      A9 01          LDA #$01
DC0C      D0 02          BNE $DC10
DC0E      A9 00          LDA #$00
DC10      85 B2          STA TEMP18
DC12      60             RTS

; Measures distance of object A from object B using the
; Inputs
; x = index of sprite A
;
; $AA = X of sprite B
; $AB = X extent of sprite B
; $AC = Y of sprite B
; $AD = Y extent of sprite B
; $AE = object index
;
; Outputs
; $A9 = X distance
; $A7 = Y distance
; $B2 = 0 or 1. 0 = definite overlap between sprites somewhere, 1 = probable non-overlap
; Z flag set if overlap

DC13      BD CF 1A       LDA SPRITE_X,X                            ;Is A.X (object A's X coord) >
B.X (object B's X coord) ?
DC16      C5 AA          CMP TEMP10
DC18      B0 09          BCS $DC23                                 ;Yes
DC1A      38             SEC
DC1B      BD E3 1E       LDA SPRITE_X_EXTENT,X                     ;No, so compute A.ExtentX -
```

```
B.ExtentX
DC1E      E5 AA          SBC TEMP10

DC20      4C 29 DC       JMP $DC29
DC23      38             SEC
DC24      A5 AB          LDA TEMP11                           ;Compute B.ExtentX - A.X
DC26      FD CF 1A       SBC SPRITE_X,X
DC29      85 A9          STA TEMP9                            ;Save X distance
DC2B      BD 26 1B       LDA SPRITE_Y,X
DC2E      C5 AC          CMP TEMP12
DC30      B0 09          BCS $DC3B
DC32      38             SEC
DC33      BD 3A 1F       LDA SPRITE_Y_EXTENT,X                ;Compute A.ExtentY - B.Y
DC36      E5 AC          SBC TEMP12                           ;Result = Y distance
DC38      4C 41 DC       JMP $DC41                            ;And jump to where it gets
stored
DC3B      38             SEC
DC3C      A5 AD          LDA TEMP13                           ;Get B.ExtentY
DC3E      FD 26 1B       SBC SPRITE_Y,X                       ;Subtract A.Y
DC41      85 A7          STA TEMP7                            ;Save the result as the Y
distance
DC43      C9 07          CMP #$07                             ;Is Y distance more than 7?
DC45      B0 11          BCS $DC58                            ;Yes
DC47      A5 A9          LDA TEMP9                            ;Is X distance more than 4?
DC49      C9 04          CMP #$04
DC4B      B0 0B          BCS $DC58                            ;Yes
DC4D      18             CLC                                  ;If we get here, Y distance <=
7 and X
DC4E      65 A7          ADC TEMP7                            ; distance <=4 add X distance
to Y dist
DC50      C9 08          CMP #$08                             ; is it more than 8?  This is
a quick
DC52      B0 04          BCS $DC58                            ; overlap check.  Yes, it's
more than 8
DC54      A9 00          LDA #$00                             ;Store 0 in $B2 to signify an
overlap
DC56      F0 02          BEQ $DC5A                            ; in both objects.
DC58      A9 01          LDA #$01
DC5A      85 B2          STA TEMP18                           ;$B2 holds 1 to signify the
quick overlap
                                                              ;  check didn't yield

anything
DC5C      60             RTS

;See if one sprite overlaps another
;         x = index of one sprite
;         y = index of other sprite
;         returns $B7 0 = not overlapping, 1 = overlapping

DC5D      A9 00          LDA #$00                             ;Set collision result to 0
initially
DC5F      85 B7          STA FRMCNT
DC61      BD 26 1B       LDA SPRITE_Y,X                       ;Get sprite top Y-pos
DC64      D9 3A 1F       CMP SPRITE_Y_EXTENT,Y                ;Compare with Y-pos of bott of
prev. sprite
DC67      B0 1C          BCS $DC85                            ;If top of current sprite is
lower on the
                                                              ; screen then previous then

done
DC69      BD 3A 1F       LDA SPRITE_Y_EXTENT,X                ;Get sprite bottom Y-pos
DC6C      D9 26 1B       CMP SPRITE_Y,Y                       ;Compare with Y-pos of bottom
of previous
DC6F      90 14          BCC $DC85                            ; sprite.
DC71      BD CF 1A       LDA SPRITE_X,X
DC74      D9 E3 1E       CMP SPRITE_X_EXTENT,Y
DC77      B0 0C          BCS $DC85
DC79      BD E3 1E       LDA SPRITE_X_EXTENT,X
DC7C      D9 CF 1A       CMP SPRITE_X,Y
DC7F      90 04          BCC $DC85
DC81      A9 01          LDA #$01                             ;Collision has occurred

DC83      85 B7          STA FRMCNT                           ;Save TRUE value here
DC85      60             RTS
```

```
;
;Setup DLL and DLs
;
DC86      A9 04          LDA #$04                                    ;Pointer to $1804
DC88      85 BC          STA TADDR1L                                 ;
DC8A      A9 18          LDA #$18                                    ;
DC8C      85 BD          STA TADDR1H                                 ;
DC8E      A9 0C          LDA #$0C                                    ;
DC90      85 A1          STA TEMP1                                   ;Index into DLL

DC92      A2 01          LDX $01                                     ;
DC94      BD BC F1       LDA $F1BC,X                                 ;Table of DL pointers
DC97      85 BA          STA TADDRL                                  ;
DC99      BD C8 F1       LDA $F1C8,X                                 ;
DC9C      85 BB          STA TADDRH                                  ;
DC9E      A9 00          LDA #$00                                    ;Clear each $80 byte block
DCA0      A8             TAY                                         ;
DCA1      91 BA          STA (TADDRL),Y                              ;
DCA3      C8             INY                                         ;
DCA4      C0 80          CPY #$80                                    ;
DCA6      90 F9          BCC $DCA1                                   ;

DCA8      A4 A1          LDY TEMP1                                   ;Build DLL entry starting at
$1810
DCAA      A9 4F          LDA #$4F                                    ;DLL control H16 on, Offset =
16
DCAC      91 BC          STA (TADDR1L),Y                             ;
DCAE      C8             INY                                         ;Next byte
DCAF      A5 BB          LDA TADDRH                                  ;Set DL pointer
DCB1      91 BC          STA (TADDR1L),Y                             ;
DCB3      C8             INY                                         ;
DCB4      A5 BA          LDA TADDRL                                  ;
DCB6      91 BC          STA (TADDR1L),Y                             ;
DCB8      C8             INY                                         ;
DCB9      84 A1          STY TEMP1                                   ;Save DLL index
DCBB      E8             INX                                         ;
DCBC      E0 0C          CPX #$0C                                    ;
DCBE      90 D4          BCC $DC94                                   ;Setup 11 DLL entries
;
;Clear $194F - $1ACF
;
DCC0      A9 19          LDA #$19                                    ;Setup pointer to $194F
DCC2      85 BB          STA TADDRH                                  ;
DCC4      A9 4F          LDA #$4F                                    ;
DCC6      85 BA          STA TADDRL                                  ;
DCC8      A2 01          LDX $01                                     ;
DCCA      A9 00          LDA #$00                                    ;
DCCC      A8             TAY                                         ;
DCCD      91 BA          STA (TADDRL),Y                              ;Clear RAM
DCCF      C8             INY                                         ;
DCD0      C0 C0          CPY #$C0                                    ;
DCD2      90 F9          BCC $DCCD                                   ;

DCD4      A5 BA          LDA TADDRL                                  ;Move Pointer to $1A0F
DCD6      18             CLC                                         ;
DCD7      69 C0          ADC #$C0                                    ;
DCD9      85 BA          STA TADDRL                                  ;
DCDB      A5 BB          LDA TADDRH                                  ;
DCDD      69 00          ADC #$00                                    ;
DCDF      85 BB          STA TADDRH                                  ;
DCE1      CA             DEX                                         ;
DCE2      10 E6          BPL $DCCA                                   ;
;
;Setup top 4 DLL entires
;
DCE4      A0 00          LDY $00                                     ;
DCE6      A2 00          LDX $00                                     ;
DCE8      BD 1F F2       LDA $F21F,X                                 ;Read data from table
DCEB      91 BC          STA (TADDR1L),Y                             ;Write into DLL
DCED      C8             INY                                         ;
DCEE      E8             INX                                         ;
DCEF      C0 0C          CPY #$0C                                    ;copy 12 bytes
DCF1      90 F5          BCC $DCE8                                   ;
```

```
;
;Setup bottom 3 DLL entries
;
DCF3     A0 2D        LDY $2D                                     ;Index to end of DLL
DCF5     BD 1F F2     LDA $F21F,X                                 ;Read data from table
DCF8     91 BC        STA (TADDR1L),Y                             ;Store in DLL
DCFA     C8           INY                                         ;
DCFB     E8           INX                                         ;
DCFC     E0 15        CPX #$15                                    ;Copy 9 more bytes
DCFE     90 F5        BCC $DCF5                                   ;

DD00     A0 0C        LDY $0C                                     ;
DD02     A9 CF        LDA #$CF                                    ;Set DLI on 5th DLL entry + H16
DD04     91 BC        STA (TADDR1L),Y                             ;

DD06     A9 00        LDA #$00                                    ;Clear $1800-$1803
DD08     A2 03        LDX $03                                     ;
DD0A     9D 00 18     STA $1800,X                                 ;
DD0D     CA           DEX                                         ;
DD0E     10 FA        BPL $DD0A                                   ;

DD10     A2 0B        LDX $0B
DD12     A9 00        LDA #$00
DD14     9D 63 19     STA $1963,X
DD17     9D 25 21     STA $2125,X
DD1A     9D 31 21     STA $2131,X
DD1D     CA           DEX
DD1E     10 F4        BPL $DD14

DD20     20 34 DD     JSR $DD34                                   ;Setup some DL entries

DD23     A9 00        LDA #$00
DD25     85 50        STA $50
DD27     85 E5        STA $E5
DD29     A9 20        LDA #$20
DD2B     85 51        STA $51
DD2D     A9 21        LDA #$21
DD2F     85 52        STA $52
DD31     85 53        STA $53
DD33     60           RTS
;
;Setup DL entry at $2253
;
DD34     A2 16        LDX $16                                     ;Copy DL data from table to RAM
DD36     BD F8 F1     LDA $F1F8,X                                 ;
DD39     9D 53 22     STA $2253,X                                 ;
DD3C     CA           DEX                                         ;
DD3D     10 F7        BPL $DD36                                   ;
;
;Setup DL entry at $226C
;
DD3F     A2 11        LDX $11                                     ;Copy DL data from table to RAM
DD41     BD 0E F2     LDA $F20E,X                                 ;
DD44     9D 6C 22     STA $226C,X                                 ;
DD47     CA           DEX                                         ;
DD48     10 F7        BPL $DD41                                   ;

DD4A     A2 00        LDX $00
DD4C     A9 16        LDA #$16
DD4E     18           CLC
DD4F     9D 1E 21     STA $211E,X
DD52     69 01        ADC #$01
DD54     E8           INX
DD55     E0 05        CPX #$05
DD57     90 F6        BCC $DD4F
DD59     60           RTS

DD5A     A2 01        LDX $01
DD5C     BD BC F1     LDA $F1BC,X
DD5F     85 BA        STA TADDRL
DD61     BD C8 F1     LDA $F1C8,X
DD64     85 BB        STA TADDRH
DD66     A0 01        LDY $01
```

```
DD68      A9 00          LDA #$00
DD6A      91 BA          STA (TADDRL),Y
DD6C      C8             INY
DD6D      C8             INY
DD6E      C8             INY
DD6F      C8             INY
DD70      C0 80          CPY #$80
DD72      90 F6          BCC $DD6A
DD74      E8             INX
DD75      E0 0C          CPX #$0C
DD77      90 E3          BCC $DD5C
DD79      A2 0B          LDX $0B
DD7B      A9 00          LDA #$00
DD7D      9D 63 19       STA $1963,X
DD80      9D 25 21       STA $2125,X
DD83      9D 31 21       STA $2131,X
DD86      CA             DEX
DD87      10 F4          BPL $DD7D
DD89      A9 00          LDA #$00
DD8B      85 50          STA $50
DD8D      85 E5          STA $E5
DD8F      A9 20          LDA #$20
DD91      85 51          STA $51
DD93      A9 21          LDA #$21
DD95      85 52          STA $52
DD97      85 53          STA $53
DD99      60             RTS

DD9A      20 45 F5       JSR $F545                                 ;Wait for next VBLANK
DD9D      A9 04          LDA #$04                                  ;Set pointer to DLL
DD9F      85 BC          STA TADDR1L                               ;
DDA1      A9 18          LDA #$18                                  ;
DDA3      85 BD          STA TADDR1H                               ;
DDA5      A9 0C          LDA #$0C                                  ;
DDA7      85 A1          STA TEMP1
DDA9      A2 00          LDX $00
DDAB      BD BA F2       LDA $F2BA,X
DDAE      86 A2          STX TEMP2
DDB0      AA             TAX
DDB1      BD 90 F2       LDA $F290,X                               ;Setup a pointer
DDB4      85 BA          STA TADDRL                                ;
DDB6      BD A5 F2       STA $F2A5,X                               ;
DDB9      85 BB          STA TADDRH                                ;
DDBB      E0 14          CPX $14
DDBD      F0 0A          BEQ $DDC9
DDBF      A9 00          LDA #$00
DDC1      A8             TAY
DDC2      91 BA          STA (TADDRL),Y
DDC4      C8             INY
DDC5      C0 56          CPY #$56
DDC7      90 F9          BCC $DDC2
DDC9      A4 A1          LDY TEMP1
DDCB      A9 07          LDA #$07
DDCD      91 BC          STA (TADDR1L),Y
DDCF      C8             INY
DDD0      A5 BB          LDA TADDRH
DDD2      91 BC          STA (TADDR1L),Y
DDD4      C8             INY
DDD5      A5 BA          LDA TADDRL
DDD7      91 BC          STA (TADDR1L),Y
DDD9      C8             INY
DDDA      84 A1          STY $A1
DDDC      A6 A2          LDX TEMP2
DDDE      E8             INX
DDDF      E0 16          CPX $16
DDE1      90 C8          BCC $DDAB

DDE3      A0 4E          LDY $4E
DDE5      A2 00          LDX $00
DDE7      BD 2B F2       LDA $F22B,X
DDEA      91 BC          STA (TADDR1L),Y
DDEC      C8             INY
DDED      E8             INX
```

```
DDEE      E0 09          CPX $09
DDF0      90 F5          BCC $DDE7
DDF2      A0 0C          LDY $0C
DDF4      A9 87          LDA #$87
DDF6      91 BC          STA (TADDR1L),Y
DDF8      A9 00          LDA #$00
DDFA      A2 03          LDX $03
DDFC      9D 00 18       STA $1800,X
DDFF      CA             DEX
DE00      10 FA          BPL $DDFC
DE02      60             RTS

DE03      A6 77          LDX $77
DE05      BD 91 1F       LDA SPRITE_STATE_$1F91,X
DE08      F0 33          BEQ $DE3D
DE0A      BD 8C 1E       LDA SPRITE_TYPE_$1E8C,X
DE0D      C5 76          CMP $76
DE0F      D0 2C          BNE $DE3D
DE11      20 36 E1       JSR $E136
DE14      AD 3E 21       LDA $213E
DE17      D0 06          BNE $DE1F
DE19      9D E8 1F       STA $1FE8,X
DE1C      4C 47 DE       JMP $DE47
DE1F      A9 00          LDA #$00
DE21      9D 91 1F       STA SPRITE_STATE_$1F91,X
DE24      BD 8C 1E       LDA SPRITE_TYPE_$1E8C,X
DE27      29 1F          AND #$1F
DE29      A8             TAY
DE2A      A9 00          LDA #$00
DE2C      9D 8C 1E       STA SPRITE_TYPE_$1E8C,X
DE2F      C0 0B          CPY #$0B
DE31      B0 0A          BCS $DE3D
DE33      C0 06          CPY #$06
DE35      B0 04          BCS $DE3B
DE37      C0 01          CPY #$01
DE39      D0 02          BNE $DE3D
DE3B      C6 C9          DEC CRELEFT
DE3D      E8             INX
DE3E      E0 57          CPX #$57
DE40      D0 C3          BNE $DE05
DE42      A9 01          LDA #$01
DE44      85 78          STA $78
DE46      60             RTS

DE47      E8             INX
DE48      E0 57          CPX #$57
DE4A      D0 05          BNE $DE51
DE4C      A9 01          LDA #$01
DE4E      85 78          STA $78
DE50      60             RTS

DE51      86 77          STX $77
DE53      60             RTS

*
***********************************************************************
*                                                                     *
*    ZONELOAD  --  LOAD A ZONE DISPLAY LIST ENTRY WITH OBJECT DATA    *
*                                                                     *
***********************************************************************
*
;****************************************************
; Write a DL entry
;         $B3 = Low address of data
;         $AE = DL region to write to
;         $AF = palette and width
;         X = sprite number to write
;
; Outputs
; $BA, $BB = display list address
;
ZONELOAD:
DE54      86 AD          STX TEMP13                                  ;Save sprite number
```

```
DE56     A6 AE          LDX TEMP14                                 ;Display list index
DE58     BD BC F1       LDA $F1BC,X                                ;Get display list address
DE5B     85 BA          STA TADDRL                                 ;
DE5D     BD C8 F1       LDA $F1C8,X                                ;
DE60     85 BB          STA TADDRH                                 ;

DE62     BD E0 F1       LDA $F1E0,X                                ;Pointer table
DE65     85 BC          STA TADDR1L                                ;
DE67     BD EC F1       LDA $F1EC,X                                ;
DE6A     85 BD          STA TADDR1H                                ;

DE6C     BD 31 21       LDA $2131,X                                ;Next available DL slot
DE6F     FE 31 21       INC $2131,X                                ;Increment it
DE72     A8             TAY                                        ;
DE73     A5 AD          LDA TEMP13                                 ;Get sprite number
DE75     9D 25 21       STA $2125,X
DE78     91 BC          STA (TADDR1L),Y
DE7A     98             TYA                                        ;Get DL index
DE7B     0A             ASL A                                      ; *4
DE7C     0A             ASL A                                      ;
DE7D     65 BA          ADC TADDRL                                 ;Add to DL start pointer
DE7F     85 BA          STA TADDRL                                 ;
DE81     A6 AD          LDX TEMP13                                 ;Get sprite number
DE83     C6 50          DEC $50
DE85     86 AD          STX TEMP13
DE87     A0 00          LDY $00                                    ;Index into DL entry
DE89     A5 B3          LDA TEMP19                                 ;Low address of data
DE8B     91 BA          STA (TADDRL),Y                             ;Write address low
DE8D     A5 AF          LDA TEMP15                                 ;
DE8F     C8             INY                                        ;
DE90     91 BA          STA (TADDRL),Y                             ;Write palette and width
DE92     C8             INY                                        ;
DE93     38             SEC                                        ;
DE94     BD 26 1B       LDA SPRITE_Y,X                             ;Get player vertical position
DE97     A6 AE          LDX TEMP14                                 ;DL region to write to
DE99     FD D4 F1       SBC $F1D4,X                                ;Subtract position of start of
this DL region
DE9C     18             CLC                                        ;
DE9D     65 B4          ADC TEMP20                                 ;Add to high byte of data
address
DE9F     91 BA          STA (TADDRL),Y                             ;Write address high
DEA1     C8             INY                                        ;
DEA2     A6 AD          LDX TEMP13                                 ;Get sprite number
DEA4     BD CF 1A       LDA SPRITE_X,X                             ;Get player x position
DEA7     91 BA          STA (TADDRL),Y                             ;Write position
DEA9     60             RTS

DEAA     A4 53          LDY $53
DEAC     C4 52          CPY $52
DEAE     D0 03          BNE $DEB3
DEB0     4C FD DF       JMP $DFFD

DEB3     A5 E4          LDA $E4
DEB5     D0 FC          BNE $DEB3
DEB7     A4 53          LDY $53
DEB9     BE 00 22       LDX $2200,Y
DEBC     88             DEY
DEBD     10 02          BPL $DEC1
DEBF     A0 21          LDY $21
DEC1     84 53          STY $53
DEC3     BD 91 1F       LDA SPRITE_STATE_$1F91,X                   ;Sprite enabled table
DEC6     85 A0          STA TEMP0                                  ;Save
DEC8     D0 25          BNE $DEEF                                  ;Branch if enabled
DECA     BD E8 1F       LDA $1FE8,X
DECD     29 F0          AND #$F0
DECF     F0 0D          BEQ $DEDE
DED1     4A             LSR A
DED2     4A             LSR A
DED3     4A             LSR A
DED4     4A             LSR A
DED5     85 AE          STA TEMP14
DED7     A9 01          LDA #$01
DED9     85 A1          STA TEMP1
```

```
DEDB     20 05 E0       JSR $E005
DEDE     BD E8 1F       LDA $1FE8,X
DEE1     29 0F          AND #$0F
DEE3     85 AE          STA TEMP14
DEE5     A9 00          LDA #$00
DEE7     85 A1          STA TEMP1
DEE9     20 05 E0       JSR $E005
DEEC     4C DB DF       JMP $DFDB

DEEF     A5 A0          LDA TEMP0
DEF1     29 08          AND #$08
DEF3     F0 17          BEQ $DF0C
DEF5     BD E8 1F       LDA $1FE8,X
DEF8     29 0F          AND #$0F
DEFA     85 AE          STA TEMP14
DEFC     20 EA D0       JSR GETSTAMP_$D0EA
DEFF     BD 87 1D       LDA $1D87,X
DF02     85 BB          STA TADDRH
DF04     BD DE 1D       LDA $1DDE,X
DF07     85 BA          STA TADDRL
DF09     20 83 DE       JSR $DE83

DF0C     A5 A0          LDA TEMP0
DF0E     29 04          AND #$04
DF10     F0 1F          BEQ $DF31
DF12     BD E8 1F       LDA $1FE8,X
DF15     4A             LSR A
DF16     4A             LSR A
DF17     4A             LSR A
DF18     4A             LSR A
DF19     85 AE          STA TEMP14
DF1B     20 EA D0       JSR GETSTAMP_$D0EA
DF1E     BD 35 1E       LDA $1E35,X
DF21     18             CLC
DF22     7D DE 1D       ADC $1DDE,X
DF25     85 BA          STA TADDRL
DF27     BD 87 1D       LDA $1D87,X
DF2A     69 00          ADC #$00
DF2C     85 BB          STA TADDRH
DF2E     20 83 DE       JSR $DE83
DF31     A5 A0          LDA TEMP0
DF33     29 10          AND #$10
DF35     F0 10          BEQ $DF47
DF37     BD E8 1F       LDA $1FE8,X
DF3A     4A             LSR A
DF3B     4A             LSR A
DF3C     4A             LSR A
DF3D     4A             LSR A
DF3E     85 AE          STA TEMP14
DF40     A9 01          LDA #$01
DF42     85 A1          STA TEMP1
DF44     20 05 E0       JSR $E005
DF47     A5 A0          LDA TEMP0
DF49     29 20          AND #$20
DF4B     F0 0E          BEQ $DF5B
DF4D     BD E8 1F       LDA $1FE8,X
DF50     29 0F          AND #$0F
DF52     85 AE          STA TEMP14
DF54     A9 00          LDA #$00
DF56     85 A1          STA TEMP1
DF58     20 05 E0       JSR $E005
DF5B     A5 A0          LDA TEMP0
DF5D     29 80          AND #$80
DF5F     F0 49          BEQ $DFAA

DF61     A9 00          LDA #$00
DF63     85 A3          STA TEMP3
DF65     BD E8 1F       LDA $1FE8,X
DF68     F0 07          BEQ $DF71
DF6A     85 A3          STA TEMP3
DF6C     BD DE 1D       LDA $1DDE,X
DF6F     85 A2          STA TEMP2
DF71     BD 26 1B       LDA SPRITE_Y,X                              ;Get sprite y position
```

```
DF74     4A             LSR A                                        ;Get top 4 bits
DF75     4A             LSR A                                        ;
DF76     4A             LSR A                                        ;
DF77     4A             LSR A                                        ;
DF78     85 AE          STA TEMP14                                   ;DL region to write to
DF7A     20 EA D0       JSR GETSTAMP_$D0EA                           ;Determine address of data
DF7D     20 54 DE       JSR $DE54                                    ;Write DL
DF80     BD E8 1F       LDA $1FE8,X
DF83     29 F0          AND #$F0
DF85     05 AE          ORA TEMP14
DF87     9D E8 1F       STA $1FE8,X
DF8A     A5 BA          LDA TADDRL
DF8C     9D DE 1D       STA $1DDE,X
DF8F     A5 BB          LDA TADDRH
DF91     9D 87 1D       STA $1D87,X
DF94     A5 A3          LDA TEMP3
DF96     F0 12          BEQ $DFAA
DF98     0A             ASL A
DF99     0A             ASL A
DF9A     0A             ASL A
DF9B     0A             ASL A
DF9C     1D E8 1F       ORA $1FE8,X
DF9F     9D E8 1F       STA $1FE8,X
DFA2     A5 A2          LDA TEMP2
DFA4     38             SEC
DFA5     E5 BA          SBC TADDRL
DFA7     9D 35 1E       STA $1E35,X
DFAA     A5 A0          LDA TEMP0
DFAC     29 40          AND #$40
DFAE     F0 2B          BEQ $DFDB
DFB0     BD 3A 1F       LDA SPRITE_Y_EXTENT,X
DFB3     4A             LSR A
DFB4     4A             LSR A
DFB5     4A             LSR A
DFB6     4A             LSR A
DFB7     85 AE          STA TEMP14
DFB9     20 EA D0       JSR GETSTAMP_$D0EA
DFBC     20 54 DE       JSR $DE54
DFBF     BD E8 1F       LDA $1FE8,X
DFC2     29 0F          AND #$0F
DFC4     85 A2          STA TEMP2
DFC6     BD 3A 1F       LDA SPRITE_Y_EXTENT,X
DFC9     29 F0          AND #$F0
DFCB     05 A2          ORA TEMP2
DFCD     9D E8 1F       STA $1FE8,X
DFD0     38             SEC
DFD1     A5 BA          LDA TADDRL
DFD3     FD DE 1D       SBC $1DDE,X
DFD6     9D 35 1E       STA $1E35,X
DFD9     D0 00          BNE $DFDB
DFDB     BD 91 1F       LDA SPRITE_STATE_$1F91,X
DFDE     29 03          AND #$03
DFE0     9D 91 1F       STA SPRITE_STATE_$1F91,X
DFE3     F8             SED
DFE4     18             CLC
DFE5     A5 5F          LDA $5F
DFE7     69 01          ADC #$01
DFE9     85 5F          STA $5F
DFEB     D8             CLD
DFEC     A4 53          LDY $53
DFEE     C4 52          CPY $52
DFF0     F0 07          BEQ $DFF9
DFF2     A5 E4          LDA $E4
DFF4     D0 07          BNE $DFFD
DFF6     4C B9 DE       JMP $DEB9
DFF9     A9 00          LDA #$00
DFFB     85 50          STA $50
DFFD     A0 01          LDX $01
DFFF     84 E4          STY $E4
E001     88             DEY
E002     84 54          STY $54
E004     60             RTS
```

```
E005      C6 50          DEC $50
E007      C6 50          DEC $50
E009      A4 AE          LDY TEMP14
E00B      86 A5          STX TEMP5
E00D      BD 87 1D       LDA $1D87,X
E010      85 BB          STA TADDRH
E012      BD DE 1D       LDA $1DDE,X
E015      85 BA          STA TADDRL
E017      A5 A1          LDA TEMP1
E019      F0 0E          BEQ $E029
E01B      BD 35 1E       LDA $1E35,X
E01E      18             CLC
E01F      65 BA          ADC TADDRL
E021      85 BA          STA TADDRL
E023      A5 BB          LDA TADDRH
E025      69 00          ADC #$00
E027      85 BB          STA TADDRH
E029      B9 31 21       LDA $2131,Y
E02C      C9 01          CMP #$01
E02E      D0 0C          BNE $E03C
E030      A9 00          LDA #$00
E032      99 31 21       STA $2131,Y
E035      A0 01          LDY $01
E037      91 BA          STA (TADDRL),Y
E039      4C FA E0       JMP $E0FA
E03C      A5 BA          LDA TADDRL
E03E      38             SEC
E03F      F9 BC F1       SBC $F1BC,Y
E042      4A             LSR A
E043      4A             LSR A
E044      85 A4          STA TEMP4
E046      B9 25 21       LDA $2125,Y
E049      85 A2          STA TEMP2
E04B      A8             TAY
E04C      A9 00          LDA #$00
E04E      85 A8          STA TEMP8
E050      C4 A5          CPY TEMP5
E052      D0 0E          BNE $E062
E054      A9 01          LDA #$01
E056      85 A8          STA TEMP8
E058      A5 BA          LDA TADDRL
E05A      85 BC          STA TADDR1L
E05C      A5 BB          LDA TADDRH
E05E      85 BD          STA TADDR1H
E060      D0 36          BNE $E098
E062      A9 00          LDA #$00
E064      85 A3          STA TEMP3
E066      B9 E8 1F       LDA $1FE8,Y
E069      29 0F          AND #$0F
E06B      C5 AE          CMP TEMP14
E06D      F0 04          BEQ $E073
E06F      A9 01          LDA #$01
E071      85 A3          STA TEMP3
E073      B9 87 1D       LDA $1D87,Y
E076      85 BD          STA TADDR1H
E078      B9 DE 1D       LDA $1DDE,Y
E07B      85 BC          STA TADDR1L
E07D      A5 A3          LDA TEMP3
E07F      F0 0E          BEQ $E08F
E081      B9 35 1E       LDA $1E35,Y
E084      18             CLC
E085      65 BC          ADC TADDR1L
E087      85 BC          STA TADDR1L
E089      A5 BD          LDA TADDR1H
E08B      69 00          ADC #$00
E08D      85 BD          STA TADDR1H
E08F      A0 03          LDY $03
E091      B1 BC          LDA (TADDR1L),Y
E093      91 BA          STA (TADDRL),Y
E095      88             DEY
E096      10 F9          BPL $E091
E098      A0 01          LDY $01
E09A      A9 00          LDA #$00
```

```
E09C     91 BC          STA (TADDR1L),Y
E09E     A6 AE          LDX TEMP14
E0A0     BD E0 F1       LDA $F1E0,X
E0A3     85 BC          STA TADDR1L
E0A5     BD EC F1       LDA $F1EC,X
E0A8     85 BD          STA TADDR1H
E0AA     DE 31 21       DEC $2131,X
E0AD     BC 31 21       LDY $2131,X
E0B0     A9 00          LDA #$00
E0B2     91 BC          STA (TADDR1L),Y
E0B4     88             DEY
E0B5     B1 BC          LDA (TADDR1L),Y
E0B7     C5 A5          CMP TEMP5
E0B9     D0 02          BNE $E0BD
E0BB     A5 A2          LDA TEMP2
E0BD     9D 25 21       STA $2125,X
E0C0     A4 A4          LDY TEMP4
E0C2     A5 A8          LDA TEMP8
E0C4     D0 34          BNE $E0FA
E0C6     A5 A2          LDA TEMP2
E0C8     91 BC          STA (TADDR1L),Y
E0CA     AA             TAX
E0CB     A5 A3          LDA TEMP3
E0CD     D0 22          BNE $E0F1
E0CF     BD DE 1D       LDA $1DDE,X
E0D2     85 A6          STA TEMP6
E0D4     A5 BA          LDA TADDRL
E0D6     9D DE 1D       STA $1DDE,X
E0D9     A5 BB          LDA TADDRH
E0DB     9D 87 1D       STA $1D87,X
E0DE     BD 35 1E       LDA $1E35,X
E0E1     F0 17          BEQ $E0FA
E0E3     A5 A6          LDA TEMP6
E0E5     38             SEC
E0E6     E5 BA          SBC TADDRL
E0E8     18             CLC
E0E9     7D 35 1E       ADC $1E35,X
E0EC     9D 35 1E       STA $1E35,X
E0EF     D0 09          BNE $E0FA
E0F1     A5 BA          LDA TADDRL
E0F3     38             SEC
E0F4     FD DE 1D       SBC $1DDE,X
E0F7     9D 35 1E       STA $1E35,X
E0FA     A6 A5          LDX TEMP5
E0FC     A5 A1          LDA TEMP1
E0FE     D0 28          BNE $E128
E100     BD 35 1E       LDA $1E35,X
E103     F0 1D          BEQ $E122
E105     18             CLC
E106     7D DE 1D       ADC $1DDE,X
E109     9D DE 1D       STA $1DDE,X
E10C     A9 00          LDA #$00
E10E     9D 35 1E       STA $1E35,X
E111     7D 87 1D       ADC $1D87,X
E114     9D 87 1D       STA $1D87,X
E117     BD E8 1F       LDA $1FE8,X
E11A     4A             LSR A
E11B     4A             LSR A
E11C     4A             LSR A
E11D     4A             LSR A
E11E     9D E8 1F       STA $1FE8,X
E121     60             RTS
E122     A9 00          LDA #$00
E124     9D E8 1F       STA $1FE8,X
E127     60             RTS

E128     A9 00          LDA #$00
E12A     9D 35 1E       STA $1E35,X
E12D     BD E8 1F       LDA $1FE8,X
E130     29 0F          AND #$0F
E132     9D E8 1F       STA $1FE8,X
E135     60             RTS
;
```

```
; Expects
; x = object index
;
;
; Returns
; $213E is 0 if the object has been set up OK, nonzero otherwise
;
;
E136      A9 01         LDA #$01
E138      85 A0         STA TEMP0
E13A      A9 00         LDA #$00
E13C      85 B7         STA FRMCNT
E13E      BD 8C 1E      LDA SPRITE_TYPE_$1E8C,X                         ;Get current type
E141      29 1F         AND #$1F
E143      F0 08         BEQ $E14D                                       ; 0? (meaning null sprite
type)
E145      C9 0F         CMP #$0F                                        ;Electrode style (this is a
placeholder, not strictly a sprite type)?
E147      F0 04         BEQ $E14D
E149      A9 00         LDA #$00
E14B      85 A0         STA TEMP0
E14D      BD 26 1B      LDA SPRITE_Y,X                                  ;Get sprite Y
E150      4A            LSR A
E151      4A            LSR A
E152      4A            LSR A
E153      4A            LSR A                                           ;Divide Y by 16
E154      85 A1         STA TEMP1                                       ;Save result
E156      A8            TAY                                             ;Y = result
E157      A5 A0         LDA TEMP0
E159      D0 07         BNE $E162
E15B      B9 63 19      LDA $1963,Y
E15E      C9 17         CMP #$17
E160      B0 19         BCS $E17B
E162      BD 3A 1F      LDA SPRITE_Y_EXTENT,X                           ;Get sprite X
E165      4A            LSR A
E166      4A            LSR A
E167      4A            LSR A
E168      4A            LSR A                                           ;Divide X by 16
E169      85 A2         STA TEMP2                                       ;Save result
E16B      C5 A1         CMP TEMP1
E16D      F0 18         BEQ $E187
E16F      A8            TAY
E170      A5 A0         LDA TEMP0
E172      D0 0D         BNE $E181
E174      B9 63 19      LDA $1963,y
E177      C9 17         CMP #$17
E179      90 06         BCC $E181
E17B      A9 01         LDA #$01                                        ;Something is wrong
E17D      8D 3E 21      STA $213E                                       ;Set return value
E180      60            RTS

E181      A9 C0         LDA #$C0
E183      E6 B7         INC FRMCNT
E185      D0 02         BNE $E189
E187      A9 80         LDA #$80
E189      1D 91 1F      ORA SPRITE_STATE_$1F91,X
E18C      9D 91 1F      STA SPRITE_STATE_$1F91,X
E18F      E6 B7         INC FRMCNT
E191      A9 00         LDA #$00
E193      8D 3E 21      STA $213E
E196      A5 A0         LDA TEMP0
E198      D0 12         BNE $E1AC
E19A      86 AA         STX TEMP10
E19C      A6 A1         LDX TEMP1
E19E      FE 63 19      INC $1963,X
E1A1      A6 A2         LDX TEMP2
E1A3      E4 A1         CPX TEMP1
E1A5      F0 03         BEQ $E1AA
E1A7      FE 63 19      INC $1963,X
E1AA      A6 AA         LDX TEMP10
E1AC      4C 39 E3      JMP $E339
;
; This is called whenever the enemy in question has been finished processing.
```

```
:   I think it's to actually draw the sprite.
;   But, I'm not sure what all the zero page variables are for just yet.
;
; Expects
; $BE = Sprite X
; $BF = Sprite Y
; $C0 = Sprite X Extent
; $C1 = Sprite Y Extent
;
; Returns
; $213E is 0 if success
; Non zero if fail
;
E1AF     A0 00          LDY $00
E1B1     8C 3E 21       STY $213E
E1B4     84 A1          STY TEMP1
E1B6     84 A2          STY TEMP2
E1B8     84 A3          STY TEMP3
E1BA     84 A4          STY TEMP4
E1BC     84 B7          STY FRMCNT
E1BE     BD E8 1F       LDA $1FE8,X
E1C1     29 0F          AND #$0F
E1C3     85 A5          STA TEMP5
E1C5     BD E8 1F       LDA $1FE8,X
E1C8     4A             LSR A
E1C9     4A             LSR A
E1CA     4A             LSR A
E1CB     4A             LSR A
E1CC     85 A6          STA TEMP6
E1CE     A9 03          LDA #$03                              ;Mask off lower 2 bits of
SPRITE_STATE
E1D0     3D 91 1F       AND SPRITE_STATE_$1F91,X
E1D3     9D 91 1F       STA SPRITE_STATE_$1F91,X
E1D6     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E1D9     D0 03          BNE $E1DE
E1DB     4C 79 E2       JMP $E279
E1DE     C9 03          CMP #$03
E1E0     D0 03          BNE $E1E5
E1E2     4C 9F E2       JMP $E29F

E1E5     A5 BF          LDA YINTEND_BF                        ;Divide sprite Y by 16
E1E7     4A             LSR A
E1E8     4A             LSR A
E1E9     4A             LSR A
E1EA     4A             LSR A
E1EB     85 A7          STA TEMP7                             ; $A7 = result
E1ED     A5 C1          LDA YYINTEND_C1                       ;Divide sprite Y Extent by 16
E1EF     4A             LSR A
E1F0     4A             LSR A
E1F1     4A             LSR A
E1F2     4A             LSR A
E1F3     85 A8          STA TEMP8                             ; $A8 = result
E1F5     C5 A5          CMP TEMP5
E1F7     F0 18          BEQ $E211
E1F9     A5 A5          LDA TEMP5
E1FB     C5 A7          CMP TEMP7
E1FD     F0 12          BEQ $E211
E1FF     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E202     09 20          ORA #$20
E204     9D 91 1F       STA SPRITE_STATE_$1F91,X
E207     E6 B7          INC FRMCNT
E209     E6 B7          INC FRMCNT
E20B     A9 FF          LDA #$FF
E20D     85 A1          STA TEMP1
E20F     D0 0A          BNE $E21B
E211     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E214     09 08          ORA #$08
E216     9D 91 1F       STA SPRITE_STATE_$1F91,X
E219     E6 B7          INC FRMCNT
E21B     A5 A7          LDA TEMP7
E21D     C5 A5          CMP TEMP5
E21F     F0 12          BEQ $E233
E221     C5 A6          CMP TEMP6
```

```
E223     F0 0E          BEQ $E233
E225     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E228     09 80          ORA #$80
E22A     9D 91 1F       STA SPRITE_STATE_$1F91,X
E22D     E6 B7          INC FRMCNT
E22F     A9 01          LDA #$01
E231     85 A3          STA TEMP3
E233     A5 A6          LDA TEMP6
E235     F0 24          BEQ $E25B
E237     C5 A7          CMP TEMP7
E239     F0 16          BEQ $E251
E23B     C5 A8          CMP TEMP8
E23D     F0 12          BEQ $E251
E23F     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E242     09 10          ORA #$10
E244     9D 91 1F       STA SPRITE_STATE_$1F91,X
E247     E6 B7          INC FRMCNT
E249     E6 B7          INC FRMCNT
E24B     A9 FF          LDA #$FF
E24D     85 A2          STA TEMP2
E24F     D0 0A          BNE $E25B
E251     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E254     09 04          ORA #$04
E256     9D 91 1F       STA SPRITE_STATE_$1F91,X
E259     E6 B7          INC FRMCNT
E25B     A5 A8          LDA TEMP8
E25D     C5 A7          CMP TEMP7
E25F     F0 59          BEQ $E2BA
E261     C5 A5          CMP TEMP5
E263     F0 55          BEQ $E2BA
E265     C5 A6          CMP TEMP6
E267     F0 51          BEQ $E2BA
E269     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E26C     09 40          ORA #$40
E26E     9D 91 1F       STA SPRITE_STATE_$1F91,X
E271     E6 B7          INC FRMCNT
E273     A9 01          LDA #$01
E275     85 A4          STA TEMP4
E277     D0 41          BNE $E2BA
E279     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E27C     09 20          ORA #$20
E27E     9D 91 1F       STA SPRITE_STATE_$1F91,X
E281     E6 B7          INC FRMCNT
E283     E6 B7          INC FRMCNT
E285     A9 FF          LDA #$FF
E287     85 A1          STA TEMP1
E289     A5 A6          LDA TEMP6
E28B     F0 2D          BEQ $E2BA
E28D     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E290     09 10          ORA #$10
E292     9D 91 1F       STA SPRITE_STATE_$1F91,X
E295     E6 B7          INC FRMCNT
E297     E6 B7          INC FRMCNT
E299     A9 FF          LDA #$FF
E29B     85 A2          STA TEMP2
E29D     D0 1B          BNE $E2BA
E29F     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E2A2     09 08          ORA #$08
E2A4     9D 91 1F       STA SPRITE_STATE_$1F91,X
E2A7     E6 B7          INC FRMCNT
E2A9     A5 A6          LDA TEMP6
E2AB     F0 0A          BEQ $E2B7
E2AD     BD 91 1F       LDA SPRITE_STATE_$1F91,X
E2B0     09 04          ORA #$04
E2B2     9D 91 1F       STA SPRITE_STATE_$1F91,X
E2B5     E6 B7          INC FRMCNT
E2B7     4C 39 E3       JMP $E339
E2BA     BD 8C 1E       LDA SPRITE_TYPE_$1E8C,X
E2BD     29 1F          AND #$1F
E2BF     F0 4E          BEQ $E30F
E2C1     C9 0F          CMP #$0F
E2C3     F0 4A          BEQ $E30F
E2C5     A5 A3          LDA TEMP3
```

```
E2C7     F0 0B          BEQ $E2D4
E2C9     30 09          MNI $E2D4
E2CB     A4 A7          LDY TEMP7
E2CD     B9 63 19       LDA $1963,Y
E2D0     C9 17          CMP #$17
E2D2     B0 4D          BCS $E321
E2D4     A5 A4          LDA TEMP4
E2D6     F0 0B          BEQ $E2E3
E2D8     30 09          BMI $E2E3
E2DA     A4 A8          LDY TEMP8
E2DC     B9 63 19       LDA $1963,Y
E2DF     C9 17          CMP #$17
E2E1     B0 3E          BCS $E321
E2E3     A4 A8          LDY TEMP8
E2E5     18             CLC
E2E6     B9 63 19       LDA $1963,Y
E2E9     65 A4          ADC TEMP4
E2EB     99 63 19       STA $1963,Y
E2EE     A4 A7          LDY TEMP7
E2F0     18             CLC
E2F1     B9 63 19       LDA $1963,Y
E2F4     65 A3          ADC TEMP3
E2F6     99 63 19       STA $1963,Y
E2F9     A4 A5          LDY TEMP5
E2FB     18             CLC
E2FC     B9 63 19       LDA $1963,Y
E2FF     65 A1          ADC TEMP1
E301     99 63 19       STA $1963,Y
E304     A4 A6          LDY TEMP6
E306     18             CLC
E307     B9 63 19       LDA $1963,Y
E30A     65 A2          ADC TEMP2
E30C     99 63 19       CTA $1963,Y
E30F     A0 0B          LDY $0B
E311     B9 63 19       LDA $1963,Y
E314     10 05          BPL $E31B
E316     A9 00          LDA #$00
E318     99 63 19       STA $1963,Y
E31B     88             DEY
E31C     D0 F3          BNE $E311
E31E     4C 39 E3       JMP $E339
E321     BD 26 1B       LDA SPRITE_Y,X
E324     85 BF          STA YINTEND_BF
E326     BD 3A 1F       LDA SPRITE_Y_EXTENT,X
E329     85 C1          STA YYINTEND_C1
E32B     A9 03          LDA #$03

E32D     3D 91 1F       AND SPRITE_STATE_$1F91,X
E330     9D 91 1F       STA SPRITE_STATE_$1F91,X
E333     A9 01          LDA #$01                                    ;Fail!!!!
E335     8D 3E 21       STA $213E
E338     60             RTS

E339     8A             TXA
E33A     A4 52          LDY $52
E33C     99 00 22       STA $2200,Y
E33F     88             DEY
E340     10 02          BPL $E344
E342     A0 21          LDY $21
E344     84 52          STY $52
E346     A5 50          LDA $50
E348     18             CLC
E349     65 B7          ADC FRMCNT
E34B     85 50          STA $50
E34D     C5 51          CMP $51
E34F     90 04          BCC $E355
E351     A9 01          LDA #$01
E353     85 54          STA $54
E355     60             RTS
;
;Turn off sound
;
E356     98             TYA                                         ;Save A
```

```
E357     48             PHA                                              ;
E358     8A             TXA                                              ;Save X
E359     48             PHA                                              ;
E35A     A0 00          LDY $00                                          ;Set sound volume to 0
E35C     84 19          STY $19                                          ;
E35E     84 1A          STY $1A                                          ;
E360     8C 40 19       STY $1940
E363     8C 41 19       STY $1941
E366     88             DEY
E367     8C 38 19       STY $1938
E36A     8C 39 19       STY $1939
E36D     68             PLA
E36E     AA             TAX
E36F     68             PLA
E370     A8             TAY
E371     60             RTS

E372     CD 38 19       CMP $1938
E375     D0 0D          BNE $E384
E377     A9 00          LDA #$00
E379     85 19          STA $19
E37B     8D 40 19       STA $1940
E37E     A9 FF          LDA #$FF
E380     8D 38 19       STA $1938
E383     60             RTS

E384     38             SEC
E385     ED 39 19       SBC $1939
E388     D0 0A          BNE $E394
E38A     85 1A          STA $1A
E38C     8D 41 19       STA $1941
E38F     A9 FF          LDA #$FF
E391     8D 39 19       STA $1939
E394     60             RTS

*****************************************************************************
*                                                                           *
*                                                                           *
*    ROBOTRON    22-AUGUST-83                                               *
*                24-AUGUST-83           4:00                                *
*                                                                           *
*         RSOUNDS.S                 SOUND ROUTINES AND DATA                 *
*                                                                           *
*****************************************************************************
*
*    ORIGINALLY:    ALIEN;TUNES.S            - SOUND DRIVER

*  THERE ARE 3 EXTERNAL ROUTINES IN THIS PACKAGE:
*         DOTUNE  STARTS A TUNE, THE NUMBER OF THE TUNE IS IN THE ACCUMULATOR
*         KILLTUNE KILLS A TUNE, THE NUMBER OF THE TUNE IS IN THE ACCUMULATOR
*         CLEARTUN CLEARS OUT ALL TUNES, INCLUDING BACKED UP TUNES
*  NOTE THAT THESE ALL PRESERVE X AND Y REGISTERS.
*  IN ADDITION, THERE IS A ROUTINE CALLED 'TUNER' WHICH SHOULD BE CALLED ONCE
*  PER FRAME (PREFERABLY AT ABOUT THE SAME TIME EACH FRAME).  NOTE THAT IT
*  CAN BE CALLED LESS FREQUENTLY (SAY ONCE PER TWO FRAMES) IF THE DURATION
*  DATA IS HALVED AND YOU ARE WILLING TO LIVE WITH THE DECREASED DURATION
*  RESOLUTION.

*  A BRIEF DESCRIPTION OF THE DRIVER.  THIS IS A PRIORITY BASED TUNE DRIVER
*  WITH TWO BACK-UP CHANNELS.  WHENEVER A DOTUNE IS EXECUTED, A CLEAR CHANNEL
*  IS LOOKED FOR.  IF IT IS NOT AVAILABLE, PRIORITIES ARE CHECKED TO SEE IF
*  THE NEW TUNE SHOULD PREEMPT A LOWER PRIORITY TUNE.  WHICHEVER TUNE LOSES
*  IS STORED IN ONE OF THE BACK-UP CHANNELS AND RESTARTED WHENEVER A CHANNEL
*  FREES ITSELF.  IF MORE THAN ONE IS BACKED UP, THE HIGHER PRIORITY ONE IS
*  RESTARTED.  CONCEPTUALLY THERE ARE 4 SOUND CHANNELS, OF WHICH CHANNELS 0
*  AND 1 ARE ACTIVE AND CHANNELS 2 AND 3 ARE INACTIVE.  THERE ARE PROVISIONS
*  FOR INFINITE TUNES AND TUNES WHICH INVOKE OTHER TUNES WHEN THEY ARE FINISHED.

*  LET'S START WITH THE DATA.  NOTE THAT I HAVE LEFT A SELECTION OF TUNES
*  AND SOUND EFFECTS FROM MS PAC-MAN IN HERE TO SHOW HOW THE DATA IS ORGANIZED.
*  THERE ARE 5 TABLES WHICH CONTROL THE TUNES.  THESE ARE INDEXED BY THE TUNE
*  TUMBER.  THESE ARE:
*         TBASE   LOW BYTE OF BASE ADDRESS OF TUNE DATA
*         TBASE1  HI BYTE
*         TCTLOFF OFFSET INTO TUNE DATA WHERE CTL DATA STARTS
*         TVOLOFF OFFSET INTO TUNE DATA WHERE VOLUME DATA STARTS
*         TPRIOR  PRIORITY OF THIS TUNE (FROM 0 TO $7F, 0 IS LOWEST)
*  THESE TABLES TELL WHERE TO FIND AND HOW TO INTERPRET THE 'TUNE DATA' FOR
*  EACH TUNE.  THE TUNE DATA IS ORGANIZED AS FOLLOWS:
*         FREQUENCY INFORMATION:  THIS IS A SET OF PAIRS 'FREQ,DUR' WHERE
*                 FREQ IS THE VALUE TO STUFF INTO AUDF0 AND DUR IS THE NUMBER
*                 OF FRAMES TO LEAVE IT THERE.  THE TUNE IS TERMINATED WHEN
*                 A NEGATIVE FREQ IS ENCOUNTERED (THERE ARE ONLY 5 SIGNIFICANT
*                 BITS OF FREQ, SO THIS DOES NOT LIMIT THE TUNES).  THERE ARE
*                 THE FOLLOWING WAYS TO TERMINATE A TUNE:
*                   $FF - THE TUNE IS OVER
*                   $FE - REPEAT THE TUNE (MAKES THE TUNE INFINITE)
*                   $FD,TUNENUM - IMMEDIATELY START TUNENUM
*         CONTROL INFORMATION:  THIS IS A SET OF PAIRS 'CTL,DUR' WHERE CTL IS
*                 THE VALUE TO STUFF INTO AUDC0 AND DUR IS THE NUMBER OF FRAMES
*                 TO LEAVE IT THERE.  A DUR OF '$00' WILL MAKE THE CTL LAST THE
*                 ENTIRE TUNE.
*         VOLUME INFORMATION:  THIS IS A SET OF PAIRS 'VOL,DUR' WHERE VOL IS
*                 THE VALUE TO STUFF INTO AUDV0 AND DUR IS THE NUMBER OF FRAMES
*                 TO LEAVE IT THERE.  AGAIN, A DUR OF '$00' WILL KEEP THE VOL
*                 FOR THE ENTIRE TUNE.
*  NOTE THAT THE TUNE DATA FOR A SINGLE TUNE CANNOT BE LARGER THAN A PAGE.
*  BREAK THE TUNE INTO TWO PARTS WITH ONE STARTING THE OTHER IF YOU NEED A
*  LARGER TUNE.  TAKE A LOOK AT THE EXAMPLES AT THE END OF THIS FILE IF THIS
*  DESCRIPTION DOESN'T MAKE SENSE, THERE ARE EXAMPLES OF INFINITE TUNES AND
*  TUNES CALLING EACH OTHER.  NOTE THAT AN INFINITE TUNE CAN BE STOPPED BY
*  A KILLTUN OR A CLEARTUN.

*  AND NOW TO THE VARIABLES NEEDED.  COPY OUT THESE VARIABLES (CHANGING THE
*  LOCATIONS AS NEEDED):

*
*         ALL VARIABLES USED ARE IN RMAIN.S
*
*  NOTE THAT SOUNDZP CAN PROBABLY BE PUT THE SAME PLACE AS YOUR LOADER TEMP
*  VARIABLES.
*  THE ONLY VARIABLES HERE THAT YOU MIGHT WISH TO LOOK AT FROM 'OUTSIDE' ARE
*  TUNON AND TUNINDEX.  TUNON IS 1 IF A CHANNEL IS USED, 0 IF IT IS FREE.
*  TUNINDEX IS THE TUNE NUMBER OF A USED CHANNEL, 0 IF THE CHANNEL IS FREE.
*  NOTE THAT IF TUNON IS 0, A ZERO VOLUME IS FORCED INTO THE APPROPRIATE VOLUME
*  REGISTER, SO SIMPLY ZEROING THE TUNE DATA WILL SHUT THE TIA UP (OF COURSE,
*  CLEARTUN ALSO DOES THIS).

*  THE SIZE OF THIS CODE IS ABOUT $200 BYTES.  NOTE THAT THE TUNE DATA WILL
*  PROBABLY RUN MUCH LARGER.

*  THIS ROUTINE ENTERS A TUNE INTO ONE OF THE SOUND CHANNELS IF IT CAN
*  INPUT:  TUNE NUMBER IN ACCUMULATOR
*  X AND Y ARE PRESERVED.

;Disassembly of $E395-$E5C7 compliments of Dan Boris & "Scotty"
; DoTune in original source
; Play a sound
; Expects accumulator to be set to the sound index
; x = an object index
; y = an object index
;
; returns
; x and y were as they were before

DOTUNE:
E395      85 58          STA $58                                        ;Save accumulator parameter
E397      A5 E3          LDA $E3                                        ;Are we in attract mode??
E399      F0 03          BEQ $E39E                                      ;No
E39B      4C 44 E4       JMP $E444                                      ;If in attract mode, then just
exit
                                                                        : (straight jump to an RTS)
E39E      98             TYA                                            ;Save x and y on the stack
E39F      48             PHA
E3A0      8A             TXA
E3A1      48             PHA
```

```
E3A2      A0 01          LDY $01
E3A4      A6 58          LDX $58
E3A6      BD 70 E6       LDA $E670,X
E3A9      F0 37          BEQ $E3E2
E3AB      85 58          STA $58
E3AD      BD 46 E6       LDA $E646,X
E3B0      CD 40 19       CMP $1940
E3B3      B0 03          BCS $E3B8
E3B5      4C 40 E4       JMP $E440

E3B8      CD 41 19       CMP $1941
E3BB      B0 03          BCS $E3C0
E3BD      4C 40 E4       JMP $E440

E3C0      A9 01          LDA #$01
E3C2      8D 43 19       STA $1943
E3C5      8A             TXA
E3C6      A8             TAY
E3C7      09 80          ORA #$80
E3C9      8D 38 19       STA $1938
E3CC      B9 46 E6       LDA $E646,Y
E3CF      8D 40 19       STA $1940
E3D2      A5 58          LDA $58
E3D4      A8             TAY
E3D5      09 80          ORA #$80
E3D7      8D 39 19       STA $1939
E3DA      B9 46 E6       LDA $E646,Y
E3DD      8D 41 19       STA $1941
E3E0      B0 5E          BCS $E440
E3E2      B9 38 19       LDA $1938,Y
E3E5      C9 FF          CMP #$FF
E3E7      F0 17          BEQ $E400
E3E9      88             DEY
E3EA      10 F6          BPL $E3E2
E3EC      BD 46 E6       LDA $E646,X
E3EF      CD 41 19       CMP $1941
E3F2      A0 01          LDY $01
E3F4      B0 0A          BCS $E400
E3F6      BD 46 E6       LDA $E646,X
E3F9      CD 40 19       CMP $1940
E3FC      90 42          BCC $E440
E3FE      A0 00          LDY $00
E400      98             TYA
E401      49 01          EOR #$01
E403      AA             TAX
E404      BD 38 19       LDA $1938,X
E407      C5 58          CMP $58
E409      D0 08          BNE $E413
E40B      8A             TXA
E40C      48             PHA
E40D      A8             TAY
E40E      68             PLA
E40F      AA             TAX
E410      4C 1E E4       JMP $E41E

E413      29 7F          AND #$7F
E415      C5 58          CMP $58
E417      D0 05          BNE $E41E
E419      8A             TXA
E41A      48             PHA
E41B      A8             TAY
E41C      68             PLA
E41D      AA             TAX
E41E      AD 43 19       LDA $1943
E421      F0 0A          BEQ $E42D
E423      A9 FF          LDA #$FF
E425      9D 38 19       STA $1938,X
E428      A9 00          LDA #$00
E42A      9D 40 19       STA $1940,X
E42D      A5 58          LDA $58
E42F      AA             TAX
E430      09 80          ORA #$80
E432      99 38 19       STA $1938,Y
```

```
E435      BD 46 E6       LDA $E646,X
E438      99 40 19       STA $1940,Y
E43B      A9 00          LDA #$00
E43D      8D 43 19       STA $1943
E440      68             PLA
E441      AA             TAX
E442      68             PLA
E443      A8             TAY
E444      60             RTS

E445      85 58          STA $58
E447      A5 E3          LDA $E3
E449      F0 03          BEQ $E44E
E44B      4C 9D E4       JMP $E49D

E44E      98             TYA
E44F      48             PHA
E450      8A             TXA
E451      48             PHA
E452      A0 01          LDY $01
E454      A6 58          LDX $58
E456      B9 38 19       LDA $1938,Y
E459      C9 FF          CMP #$FF
E45B      F0 17          BEQ $E474
E45D      88             DEY
E45E      10 F6          BPL $E456
E460      BD 46 E6       LDA $E646,X
E463      CD 41 19       CMP $1941
E466      A0 01          LDY $01
E468      B0 0A          BCS $E474
E46A      BD 46 E6       LDA $E646,X
E46D      CD 40 19       CMP $1940
E470      90 27          BCC $E499
E472      A0 00          LDY $00
E474      8A             TXA
E475      CD 38 19       CMP $1938
E478      F0 1F          BEQ $E499
E47A      CD 39 19       CMP $1939
E47D      F0 1A          BEQ $E499
E47F      09 80          ORA #$80
E481      CD 38 19       CMP $1938
E484      F0 13          BEQ $E499
E486      CD 39 19       CMP $1939
E489      F0 0E          BEQ $E499
E48B      99 38 19       STA $1938,Y
E48E      BD 46 E6       LDA $E646,X
E491      99 40 19       STA $1940,Y
E494      A9 00          LDA #$00
E496      8D 43 19       STA $1943
E499      68             PLA
E49A      AA             TAX
E49B      68             PLA
E49C      A8             TAY
E49D      60             RTS

E49E      A2 01          LDX $01
E4A0      BC 38 19       LDY $1938,X
E4A3      C8             INY
E4A4      D0 03          BNE $E4A9
E4A6      4C BC E5       JMP $E5BC

E4A9      88             DEY
E4AA      10 41          BPL $E4ED
E4AC      AD 42 19       LDA $1942
E4AF      F0 19          BEQ $E4CA
E4B1      AD 43 19       LDA $1943
E4B4      D0 14          BNE $E4CA
E4B6      A9 FF          LDA #$FF
E4B8      8D 38 19       STA $1938
E4BB      8D 39 19       STA $1939
E4BE      A9 00          LDA #$00
E4C0      85 19          STA $19
E4C2      85 1A          STA $1A
```

```
E4C4     8D 40 19       STA $1940
E4C7     8D 41 19       STA $1941
E4CA     98             TYA
E4CB     29 7F          AND #$7F
E4CD     9D 38 19       STA $1938,X
E4D0     A8             TAY
E4D1     A9 01          LDA #$01
E4D3     9D 36 19       STA $1936,X
E4D6     B9 46 E6       LDA $E646,Y
E4D9     9D 40 19       STA $1940,X
E4DC     A9 FF          LDA #$FF
E4DE     9D 3A 19       STA $193A,X
E4E1     9D 3C 19       STA $193C,X
E4E4     9D 3E 19       STA $193E,X
E4E7     AD 43 19       LDA $1943
E4EA     8D 42 19       STA $1942
E4ED     B9 C8 E5       LDA $E5C8,Y
E4F0     85 56          STA $56
E4F2     B9 DD E5       LDA $E5DD,Y
E4F5     85 57          STA $57
E4F7     DE 36 19       DEC $1936,X
E4FA     D0 AA          BNE $E4A6
E4FC     B9 5B E6       LDA $E65B,Y
E4FF     9D 36 19       STA $1936,X
E502     BC 3A 19       LDY $193A,X
E505     FE 3C 19       INC $193C,X
E508     FE 3E 19       INC $193E,X
E50B     C8             INY
E50C     B1 56          LDA ($56),Y
E50E     C9 FF          CMP #$FF
E510     D0 0F          BNE $E521
E512     9D 38 19       STA $1938,X
E515     A9 00          LDA #$00
E517     95 19          STA $19,X
E519     9D 40 19       STA $1940,X
E51C     8D 42 19       STA $1942
E51F     F0 85          BEQ $E4A6
E521     C9 FE          CMP #$FE
E523     F0 29          BEQ $E54E
E525     C9 FD          CMP #$FD
E527     D0 2E          BNE $E557
E529     C8             INY
E52A     B1 56          LDA ($56),Y
E52C     9D 38 19       STA $1938,X
E52F     A8             TAY
E530     B9 C8 E5       LDA $E5C8,Y
E533     85 56          STA $56
E535     B9 DD E5       LDA $E5DD,Y
E538     85 57          STA $57
E53A     B9 5B E6       LDA $E65B,Y
E53D     9D 36 19       STA $1936,X
E540     A0 00          LDY $00
E542     98             TYA
E543     9D 3C 19       STA $193C,X
E546     9D 3E 19       STA $193E,X
E549     B1 56          LDA ($56),Y
E54B     4C 57 E5       JMP $E557

E54E     C8             INY
E54F     B1 56          LDA ($56),Y
E551     9D 40 19       STA $1940,X
E554     C8             INY
E555     B1 56          LDA ($56),Y
E557     10 0D          BPL $E566
E559     C8             INY
E55A     B1 56          LDA ($56),Y
E55C     9D 36 19       STA $1936,X
E55F     88             DEY
E560     B1 56          LDA ($56),Y
E562     C8             INY
E563     4C 80 E5       JMP $E580

E566     0A             ASL A
```

```
E567      10 16          BPL $E57F
E569      4A             LSR A
E56A      29 BF          AND #$BF
E56C      9D 36 19       STA $1936,X
E56F      98             TYA
E570      9D 3A 19       STA $193A,X
E573      DE 3E 19       DEC $193E,X
E576      DE 3C 19       DEC $193C,X
E579      A9 00          LDA #$00
E57B      95 19          STA $19,X
E57D      F0 3D          BEQ $E5BC
E57F      4A             LSR A
E580      95 17          STA $17,X
E582      98             TYA
E583      9D 3A 19       STA $193A,X
E586      BC 38 19       LDY $1938,X
E589      B9 1C E6       LDA $E61C,Y
E58C      85 56          STA $56
E58E      B9 31 E6       LDA $E631,Y
E591      85 57          STA $57
E593      BC 3E 19       LDY $193E,X
E596      B1 56          LDA ($56),Y
E598      10 01          BPL $E59B
E59A      88             DEY
E59B      95 19          STA $19,X
E59D      98             TYA
E59E      9D 3E 19       STA $193E,X
E5A1      BC 38 19       LDY $1938,X
E5A4      B9 F2 E5       LDA $E5F2,Y
E5A7      85 56          STA $56
E5A9      B9 07 E6       LDA $E607,Y
E5AC      85 57          LDA $57
E5AE      BC 3C 19       LDY $193C,X
E5B1      B1 56          LDA ($56),Y
E5B3      10 01          BPL $E5B6
E5B5      88             DEY
E5B6      95 15          STA $15,X
E5B8      98             TYA
E5B9      9D 3C 19       STA $193C,X
E5BC      CA             DEX
E5BD      30 03          BMI $E5C2
E5BF      4C A0 E4       JMP $E4A0

E5C2      A9 00          LDA #$00
E5C4      8D 43 19       STA $1943
E5C7      60             RTS

;UNKNOWN USAGE ($E5C8-$E679)
E5C8                     .BYTE $85,$90,$AD,$C3,$C7,$D0,$4D,$D0
E5D0                     .BYTE $42,$5D,$67,$7D,$8D,$2F,$45,$4F,$71,$93,$97,$DA,$B4,$E6,$E6,$E6
E5E0                     .BYTE $E6,$E6,$E6,$E7,$E7,$E8,$E8,$E8,$E8,$E8,$E9,$E9,$E9,$E9,$E9,$E9
E5F0                     .BYTE $E9,$EA,$8E,$A0,$B5,$C5,$CE,$0E,$8F,$09,$51,$62,$72,$8B,$DE,$3A

E600                     .BYTE $4A,$6F,$91,$95,$D8,$1A,$1C,$E6,$E6,$E6,$E6,$E6,$E7,$E7,$E8,$E8
E610                     .BYTE $E8,$E8,$E8,$E8,$E9,$E9,$E9,$E9,$E9,$E9,$EA,$EA,$8F,$AC,$BC,$C6
E620                     .BYTE $CF,$0F,$CF,$41,$52,$63,$73,$8C,$2E,$3B,$4B,$70,$92,$96,$D9,$1B
E630                     .BYTE $4D,$E6,$E6,$E6,$E6,$E6,$E7,$E7,$E8,$E8,$E8,$E8,$E8,$E9,$E9,$E9
E640                     .BYTE $E9,$E9,$E9,$E9,$EA,$EB,$07,$0F,$07,$0F,$0F,$19,$14,$14,$16,$06
E650                     .BYTE $08,$09,$07,$08,$06,$0F,$0F,$06,$08,$07,$19,$01,$03,$03,$06,$06
E660                     .BYTE $01,$02,$02,$0C,$01,$01,$01,$01,$01,$01,$05,$05,$2A,$03,$01,$01
E670                     .BYTE $00,$00,$00,$04,$03,$00,$00,$00,$00,$00

*
***************************************************
*                                                 *
*         TUNES -- CALLED BY DOTUNE $E395         *
*                                                 *
***************************************************
*
;SOUND CALLS
          ;D9A2          LDA #$00                                      ;Play Mutant Clone Shooting
Sound
          ;D478          LDA #$01                                      ;Play Family Pick-up Sound
```

```
         ;B75B          LDA #$02                                 ;Play Generic Explosion Sound
         ;9C26          LDA #$03                                 ;Play Family Death Sound

         ;D4CD          LDA #$05                                 ;Play Extra Man Sound
         ;D003          LDA #$06                                 ;Play Rack End Sound

         ;D014          LDA #$08                                 ;Play MC Die Sound

         ;BAF3          LDA #$0A                                 ;Play Enforcer Spark Sound
         ;971B          LDA #$0B                                 ;Play Death Noise
         ;B63F          LDA #$0C                                 ;Play Spark Fired Sound
         ;BB72          LDA #$0E                                 ;Play "Boing" Noise!!
         ;9B47          LDA #$0F                                 ;Start "Human being
programmed" Sound -
                                                                 ;   we're creating a
Prog!!!!!!!
         ;98C2          LDA #$11                                 ;Play Tank Birth Sound
         ;B9E1          LDA #$12                                 ;Play Cruise Missile Fired
Sound
         ;BCD2          LDA #$13                                 ;Play Tank Shot Life Sound
         ;925D          LDA #$14                                 ;???

*         SOUND ROUTINE VARIABLES (UNKNOWN IF STILL VALID IN 2013)

;TUNON     EQU     $1300;2 BYTES - WHETHER TUNE IS ACTIVE
;TUNINDEX  EQU     $1302;2 BYTES - WHAT TUNE IS PLAYING
;TUNPRIOR  EQU     $1304;2 BYTES - WHAT THE PRIORITY OF TUNE IS
;TUNBASE   EQU     $1306;2 BYTES - BASE ADDRESS OF TUNE DATA
;TUNBASE1  EQU     $1308;2 BYTES - HI BYTE OF BASE ADDRESS
;FREQOFF   EQU     $130A;2 BYTES - OFFSET INTO DATA FOR FREQ'S
;CTLOFF    EQU     $130C;2 BYTES - OFFSET INTO DATA FOR CTL'S
;VOLOFF    EQU     $130E;2 BYTES - OFFSET INTO DATA FOR VOL'S
;FREQTIME  EQU     $1310;2 BYTES - NUMBER FRAMES TILL NEXT FREQ
;CTLTIME   EQU     $1312;2 BYTES - NUMBER FRAMES TILL NEXT CTL
;VOLTIME   EQU     $1314;2 BYTES - NUMBER FRAMES TILL NEXT VOL
;TUNNUM    EQU     $1316;WHAT TUNE YOU WANT - PARAMETER
;TUNTEMP0  EQU     $1317;TEMP VALUE FOR TUNE DRIVER
;TUNTEMP1  EQU     $1318;TEMP VALUE FOR TUNE DRIVER


//THE BYTES SHOWN FOR FREQ/CTL/VOL WERE DETERMINED BY MASKING OFF THOSE BYTES//

*  TUNE 0 - MUTANT CLONE SHOOTING (ORIGINAL VALUES IN GREEN, FINAL VALUES IN BLUE)
SMCS      EQU     0
TMCS:                   ;.BYTE $02,$01,$03,$01,$04,$01,$05,$01,$06,$01                  ;FREQ
                        ;.BYTE $07,$01,$08,$01,$09,$01,$0A,$01,$0B,$01,$FF

                        ;.BYTE $06,$00                                                  ;CTL

                        ;.BYTE $09,$00                                                  ;VOL

E67A                    .BYTE $0D,$00,$00,$0A,$00,$10,$0F,$00,$00,$00                   ;FREQ
E684                    .BYTE $00,$03,$04,$05,$06,$07,$08

E68B                    .BYTE $09,$0A                                                   ;CTL

E68D                    .BYTE $FF,$86                                                   ;VOL


*  TUNE 1 - FAMILY PICKUP SOUND (ORIGINAL VALUES IN GREEN, FINAL VALUES IN BLUE)
SFPICK    EQU     1
TFPICK:  ;.BYTE $15,$09,$0D,$03,$15,$03,$14,$03,$0C,$03             ;FREQ
                        ;.BYTE $15,$03,$13,$03,$0B,$03,$15,$03,$12,$03
                        ;.BYTE $0A,$03,$FF

                        ;.BYTE $06,$06,$0D,$06,$06,$03,$0D,$06,$06,$03                  ;CTL
                        ;.BYTE $0D,$06,$06,$03,$0D,$06

                        ;.BYTE $09,$00                                                  ;VOL

E68F                    .BYTE $7F,$95,$06,$15,$0D,$15,$14,$0C,$FE,$0F                   ;FREQ
E699                    .BYTE $15,$13,$0B,$15,$12,$0A,$FF
```

```
E6A0                     .BYTE $06,$0D,$0D,$06,$0D,$0D,$06,$0D,$0D,$06                                    ;CTL
E6AA                     .BYTE $0D,$0D

E6AC                     .BYTE $8A,$02                                                                    ;VOL


*  TUNE 2 - GENERIC EXPLOSION (ORIGINAL VALUES IN GREEN, FINAL VALUES IN BLUE)
SCREDIE   EQU     2
TCREDIE: ;.BYTE $02,$03,$03,$03,$04,$03,$05,$03,$06,$03              ;FREQ
                         ;.BYTE $15,$03,$19,$03,$FF

                         ;.BYTE $08,$0F,$02,$06                                                          ;CTL

                         ;.BYTE $09,$03,$07,$03,$06,$03,$05,$03,$04,$03                                  ;VOL
                         ;.BYTE $06,$03,$03,$03

E6AE                     .BYTE $03,$04                                                                    ;FREQ
         [TERMINATED EFFORT, MUCH MORE WORK REQ'D]

*  TUNE 3 - SKULL AND CROSSBONES SCREAM CHANNEL 0
SSKULL0   EQU     3
TSKULL0: ;.BYTE $0C,$04,$0D,$04,$0E,$04,$0F,$04,$10,$04              ;FREQ
                         ;.BYTE $11,$04,$12,$04,$13,$04,$FF

                         ;.BYTE $04,$00                                                                  ;CTL

                         ;.BYTE $09,$00                                                                  ;VOL

*  TUNE 4 - SKULL AND CROSSBONES SCREAM CHANNEL 1
SSKULL1   EQU     4
TSKULL1: ;.BYTE $1F,$04,$1E,$04,$1D,$04,$1C,$04,$1B,$04              ;FREQ
                         ;.BYTE $1A,$04,$19,$04,$18,$04,$FF

                         ;.BYTE $04,$00                                                                  ;CTL

                         ;.BYTE $02,$04,$03,$04,$04,$04,$05,$04,$06,$04                                  ;VOL
                         ;.BYTE $05,$04,$04,$04,$03,$04,$FF


*  TUNE 5 - EXTRA MAN
SEXTRA    EQU     5
TEXTRA:  ;.BYTE $03,$02,$04,$02,$05,$02,$06,$02,$FF                  ;FREQ

                         ;.BYTE $0F,$00                                                                  ;CTL

                         ;.BYTE $09,$00                                                                  ;VOL

*  TUNE 6 - BETWEEN RACK SOUND PART A
SRACKA    EQU     6
TRACKA:  ;.BYTE $1F,$04,$1D,$04,$1B,$04,$1A,$04,$18,$04              ;FREQ
                         ;.BYTE $17,$04,$16,$04,$14,$04,$13,$04,$12,$04
                         ;.BYTE $11,$04,$10,$04,$0F,$04,$0E,$04,$0D,$04
                         ;.BYTE $0C,$04,$0B,$04,$0A,$04,$09,$02,$1F,$02
                         ;.BYTE $08,$02,$1D,$02,$07,$02,$1B,$02,$06,$02
                         ;.BYTE $1A,$02,$05,$02,$18,$02,$04,$02,$17,$02
                         ;.BYTE $15,$04,$14,$04,$13,$04,$12,$04,$11,$04
                         ;.BYTE $10,$04,$0F,$04,$0E,$04,$FD,$07


                         ;.BYTE $06,$02,$0D,$02,$06,$02,$0D,$02       ;CTL
                         ;.BYTE $06,$02,$0D,$02,$06,$02,$0D,$02
                         ;.BYTE $06,$02,$0D,$02,$06,$02,$0D,$02
                         ;.BYTE $06,$02,$0D,$02,$06,$02,$0D,$02
                         ;.BYTE $06,$02,$0D,$02,$06,$02,$0D,$02
                         ;.BYTE $06,$02,$0D,$02,$06,$02,$0D,$02
                         ;.BYTE $06,$02,$0D,$02,$06,$02,$0D,$02
                         ;.BYTE $06,$02,$0D,$02,$06,$02,$0D,$02
                         ;.BYTE $06,$02,$0D,$02,$06,$02,$0D,$02
                         ;.BYTE $06,$02,$04,$02,$06,$02,$04,$02
                         ;.BYTE $06,$02,$04,$02,$06,$02,$04,$02
                         ;.BYTE $06,$02,$04,$02,$06,$02,$04,$02
                         ;.BYTE $0D,$02,$04,$02,$0D,$02,$04,$02
                         ;.BYTE $0D,$02,$04,$02,$0D,$02,$04,$02
```

```
                        ;.BYTE $0D,$02,$04,$02,$0D,$02,$04,$02
                        ;.BYTE $0D,$02,$04,$02,$0D,$02,$04,$02

                        ;.BYTE $09,$00                                                  ;VOL

*  TUNE 7 - BETWEEN RACK SOUND PART B
SRACKB    EQU     7
TRACKB:  ;.BYTE $0D,$04,$09,$02                                                         ;FREQ
                        ;.BYTE $13,$02,$18,$02,$1D,$02,$0B,$02,$0F,$02
                        ;.BYTE $10,$02,$16,$02,$0A,$02,$12,$02,$15,$02
                        ;.BYTE $1D,$02,$0D,$02,$0E,$02,$09,$02,$13,$02
                        ;.BYTE $18,$02,$1D,$02,$0B,$02,$0F,$02,$10,$02
                        ;.BYTE $16,$02,$0A,$02,$12,$02,$15,$02,$1D,$02
                        ;.BYTE $0D,$02,$0E,$02,$09,$02,$13,$02,$18,$02
                        ;.BYTE $1D,$02,$0B,$02,$0F,$02,$10,$02,$16,$02
                        ;.BYTE $0A,$02,$12,$02,$15,$02,$1D,$02,$0D,$02
                        ;.BYTE $0E,$02,$09,$02,$13,$02,$18,$02,$1D,$02
                        ;.BYTE $0B,$02,$0F,$02,$10,$02,$16,$02,$0A,$02
                        ;.BYTE $12,$02,$15,$02,$1D,$02,$FF

                        ;.BYTE $0D,$02,$04,$04,$0D,$06,$06,$04                 ;CTL
                        ;.BYTE $04,$06,$0D,$06,$06,$04,$04,$02
                        ;.BYTE $0D,$06,$06,$04,$04,$06,$0D,$06
                        ;.BYTE $06,$04,$04,$02,$0D,$06,$06,$04
                        ;.BYTE $04,$06,$0D,$06,$06,$04,$04,$02
                        ;.BYTE $0D,$06,$06,$04,$04,$06,$0D,$06

                        ;.BYTE $09,$00                                         ;VOL

*  TUNE 8 - MC DEATH SOUND
SMCDIE    EQU     8
TMCDIE:  ;.BYTE $0A,$04,$0C,$04,$0F,$04,$1F,$80,$FF                  ;FREQ

                        ;.BYTE $08,$00                                                  ;CTL

                        ;.BYTE $09,$0C,$0F,$10,$0A,$10,$09,$10,$07,$10                  ;VOL
                        ;.BYTE $05,$10,$04,$10,$03,$0C,$02,$0C,$01,$0C

E6B0                    .BYTE $05,$06,$15,$19,$FF,$08,$08,$08,$08,$08,$02,$02,$08,$06,$05,$04
E6C0                    .BYTE $03,$05,$02,$0C,$FF,$84,$88,$19,$1A,$1B,$1C,$1D,$1E,$FF,$84,$86
E6D0                    .BYTE $00,$00,$00,$00,$00,$00,$0E,$0D,$0C,$0B,$0A,$09,$00,$0E,$0D,$0C
E6E0                    .BYTE $0B,$0A,$09,$00,$0E,$0D,$0C,$0B,$0A,$09,$00,$0E,$0D,$0C,$0B,$0A
E6F0                    .BYTE $09,$00,$0E,$0D,$0C,$0B,$0A,$09,$00,$0E,$0D,$0C,$0B,$0A,$09,$00

E700                    .BYTE $0E,$0D,$0C,$0B,$0A,$09,$00,$0E,$0D,$0C,$0B,$0A,$09,$FF,$84,$00
E710                    .BYTE $00,$00,$00,$00,$00,$0F,$0F,$0F,$0F,$0F,$0F,$00,$0F,$0F,$0F,$0F
E720                    .BYTE $0F,$0F,$00,$0F,$0F,$0F,$0F,$0F,$0F,$00,$0F,$0F,$0F,$0F,$0F,$0F
E730                    .BYTE $00,$0F,$0F,$0F,$0F,$0F,$0F,$00,$0F,$0F,$0F,$0F,$0F,$0F,$00,$0F
E740                    .BYTE $0F,$0F,$0F,$0F,$0F,$00,$0F,$0F,$0F,$0F,$0F,$0F,$00,$1F,$1F,$1D
E750                    .BYTE $1D,$1B,$1B,$1A,$1A,$18,$18,$17,$17,$16,$16,$14,$14,$13,$13,$12
E760                    .BYTE $12,$11,$11,$10,$10,$0F,$0F,$0E,$0E,$0D,$0D,$0C,$0C,$0B,$0B,$0A
E770                    .BYTE $0A,$09,$1F,$08,$1D,$07,$1B,$06,$1A,$05,$18,$04,$17,$15,$15,$14
E780                    .BYTE $14,$13,$13,$12,$12,$11,$11,$10,$10,$0F,$0F,$0E,$0E,$FD,$07,$06
E790                    .BYTE $0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06
E7A0                    .BYTE $0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06
E7B0                    .BYTE $0D,$06,$0D,$06,$04,$06,$04,$06,$04,$06,$04,$06,$04,$06,$04,$0D
E7C0                    .BYTE $04,$0D,$04,$0D,$04,$0D,$04,$0D,$04,$0D,$04,$0D,$04,$0D,$04,$89
E7D0                    .BYTE $0D,$0D,$09,$13,$18,$1D,$0B,$0F,$10,$16,$0A,$12,$15,$1D,$0D,$0E
E7E0                    .BYTE $09,$13,$18,$1D,$0B,$0F,$10,$16,$0A,$12,$15,$1D,$0D,$0E,$09,$13
E7F0                    .BYTE $18,$1D,$0B,$0F,$10,$16,$0A,$12,$15,$1D,$0D,$0E,$09,$13,$18,$1D

E800                    .BYTE $0B,$0F,$10,$16,$0A,$12,$15,$1D,$FF,$0D,$04,$04,$0D,$0D,$0D,$06
E810                    .BYTE $06,$04,$04,$04,$0D,$0D,$0D,$06,$06,$04,$0D,$0D,$0D,$06,$06,$04
E820                    .BYTE $04,$04,$0D,$0D,$0D,$06,$06,$04,$0D,$0D,$0D,$06,$06,$04,$04,$04
E830                    .BYTE $0D,$0D,$0D,$06,$06,$04,$0D,$0D,$0D,$06,$06,$04,$04,$04,$0D,$0D
E840                    .BYTE $0D,$89,$8A,$04,$8C,$04,$8F,$04,$1F,$1F,$1F,$1F,$1F,$1F,$1F,$1F
E850                    .BYTE $FF,$88,$09,$09,$09,$0F,$0A,$09,$07,$05,$04,$03,$02,$1F,$00,$1F
E860                    .BYTE $00,$FF,$83,$05,$00,$05,$00,$00,$00,$05,$05,$05,$04,$04,$04,$03
E870                    .BYTE $03,$FF,$84,$00,$00,$0E,$0E,$0E,$0E,$0E,$0E,$0E,$0E,$16,$13,$12
E880                    .BYTE $11,$0F,$0E,$0D,$0C,$0B,$0A,$09,$08,$07,$FF,$8D,$87,$05,$1F,$05
E890                    .BYTE $1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05
E8A0                    .BYTE $1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05
E8B0                    .BYTE $1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05
```

```
E8C0                     .BYTE $1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05
E8D0                     .BYTE $1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$05,$1F,$FF,$0A,$04
E8E0                     .BYTE $0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04
E8F0                     .BYTE $0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04

E900                     .BYTE $0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04
E910                     .BYTE $0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04
E920                     .BYTE $0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$0A,$04,$8B,$00
E930                     .BYTE $00,$0D,$0D,$0D,$0C,$0C,$0C,$0B,$0B,$FF,$84,$00,$00,$06,$06,$06
E940                     .BYTE $06,$06,$06,$06,$06,$02,$03,$04,$06,$FF,$88,$0A,$0A,$07,$05,$12
E950                     .BYTE $12,$12,$12,$12,$12,$12,$13,$13,$13,$13,$13,$14,$14,$14,$14,$1A
E960                     .BYTE $18,$17,$16,$14,$13,$12,$11,$10,$0F,$0E,$0D,$0C,$0B,$0A,$FF,$83
E970                     .BYTE $88,$1F,$1F,$1F,$1E,$1E,$1D,$1C,$1B,$1A,$19,$18,$17,$16,$15,$14
E980                     .BYTE $13,$12,$11,$10,$0F,$0E,$0D,$0C,$0B,$0A,$09,$08,$07,$06,$05,$04
E990                     .BYTE $FF,$81,$88,$12,$FF,$83,$86,$1F,$1F,$1F,$1F,$0C,$0D,$0E,$0F,$0F
E9A0                     .BYTE $10,$10,$10,$11,$11,$11,$12,$12,$12,$13,$13,$13,$14,$14,$14,$15
E9B0                     .BYTE $15,$15,$16,$16,$16,$17,$17,$17,$18,$18,$18,$19,$19,$19,$1A,$1A
E9C0                     .BYTE $1A,$1B,$1B,$1B,$1C,$1C,$1C,$1D,$1D,$1D,$1E,$1E,$1E,$1F,$1F,$1F
E9D0                     .BYTE $1F,$1F,$1F,$1F,$1F,$1F,$1F,$FF,$88,$88,$08,$09,$0A,$09,$0A,$0B
E9E0                     .BYTE $0A,$0B,$0C,$0B,$0C,$0D,$0C,$0D,$0E,$0D,$0E,$0F,$0E,$0F,$10,$0F
E9F0                     .BYTE $10,$11,$10,$11,$12,$13,$11,$12,$13,$14,$12,$13,$14,$15,$13,$14

EA00                     .BYTE $15,$16,$15,$16,$17,$18,$19,$17,$18,$19,$1A,$1B,$19,$1A,$1B,$1C
EA10                     .BYTE $1D,$1B,$1C,$1D,$1E,$1F,$1D,$1E,$1F,$FF,$84,$88,$06,$0D,$06,$0D
EA20                     .BYTE $06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D
EA30                     .BYTE $06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D,$06,$0D
EA40                     .BYTE $01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07
EA50                     .BYTE $01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07
EA60                     .BYTE $01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07
EA70                     .BYTE $01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07
EA80                     .BYTE $01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07
EA90                     .BYTE $01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07
EAA0                     .BYTE $01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07,$01,$07
EAB0                     .BYTE $01,$07,$01,$07,$1F,$1F,$1D,$1D,$1B,$1B,$1A,$1A,$18,$18,$17,$17
EAC0                     .BYTE $16,$16,$14,$14,$13,$13,$12,$12,$11,$11,$10,$10,$0F,$0F,$0E,$0E
EAD0                     .BYTE $0D,$0D,$0C,$0C,$0B,$0B,$0A,$0A,$1F,$1E,$1F,$1E,$1F,$1E,$1F,$1E
EAE0                     .BYTE $1D,$1E,$1D,$1E,$1C,$1D,$1C,$1D,$1B,$1C,$1B,$1C,$1A,$1B,$1A,$1B
EAF0                     .BYTE $19,$1A,$19,$1A,$18,$19,$18,$19,$17,$18,$17,$18,$16,$17,$16,$17

EB00                     .BYTE $15,$16,$15,$16,$14,$15,$14,$15,$13,$14,$13,$14,$12,$13,$12,$13
EB10                     .BYTE $11,$12,$11,$12,$10,$11,$10,$11,$0F,$10,$0F,$10,$0E,$0F,$0E,$0F
EB20                     .BYTE $0D,$0E,$0D,$0E,$0C,$0D,$0C,$0D,$0B,$0C,$0B,$0C,$0A,$0B,$0A,$0B
EB30                     .BYTE $09,$0A,$09,$0A,$08,$09,$08,$09,$07,$08,$07,$08,$06,$07,$06,$07
EB40                     .BYTE $05,$06,$06,$06,$04,$05,$04,$05,$03,$04,$03,$04,$FF,$0A,$0A,$0A
EB50                     .BYTE $0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A
EB60                     .BYTE $0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A
EB70                     .BYTE $0A,$0A,$0A,$0A,$0A,$09,$09,$09,$09,$08,$08,$08,$08,$08,$08,$08
EB80                     .BYTE $08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08
EB90                     .BYTE $08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08
EBA0                     .BYTE $08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08
EBB0                     .BYTE $08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08
EBC0                     .BYTE $08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08
EBD0                     .BYTE $08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08
EBE0                     .BYTE $08,$08,$08,$08,$08,$91,$93,$A0,$94,$A0,$94,$A0,$94,$C6,$95,$F8
EBF0                     .BYTE $96,$24,$98,$57,$99,$C7,$B4,$9A,$9A,$40,$B7,$47,$B6,$4D,$B8,$0D

EC00                     .BYTE $BB,$00,$00,$7D,$9C

*****************************************************************
*                                                               *
*                                                               *
*    ROBOTRON     6-JULY-83                                     *
*                18-JULY-83             3:30                    *
*                23-AUGUST-83          12:30                    *
*****************************************************************
*
*         RDATA.S
*
*****************************************************************
*                                                               *
*         ROBOTRON DATA                                         *
*                                                               *
*****************************************************************
```

```
*
*
*          THIS FOLLOWS RSOUNDS.S IN MEMORY... NO ORG NEEDED
*
*
******************************************
******************************************
*                                        *
*          MISCELLANEOUS TABLES          *
*                                        *
*                                        *
*                                        *
******************************************
*
*          MOVTBL  -- POINTS TO VARIOUS ROUTINES TO HANDLE
*                     MOVING EACH DIFFERENT TYPE OF OBJECT
*                     POINTERS ARE ADDRESSES IN LOW, HIGH FORMAT
*                              THE 14 OBJECT TYPES:
MOVTBL:
                         DW      GMOV                                ;GRUNTS
                         DW      FMOV                                ;MOMMIES
                         DW      FMOV                                ;DADDIES
                         DW      FMOV                                ;MIKEYS
                         DW      HMOV                                ;HULKS
                         DW      SMOV                                ;SPHEROIDS
                         DW      QMOV                                ;QUARKS
                         DW      ETMOV                               ;ENFORCERS
                         DW      ETMOV                               ;TANKS
                         DW      BMOV                                ;BRAINS
                         DW      PMOV                                ;PROGS
                         DW      MMOV                                ;ENFORCER MISSILES
                         DW      MMOV                                ;CRUISE MISSILES
                         DW      MMOV                                ;TANK MISSILES

*
**********************************************************************************
*                                                                                *
*      STICKTBL  -- FOR ANY OF THE 4-BIT DIR CODES, RETURN A 0-7  DIRECTION      *
*                                                                                *
**********************************************************************************
*
STICKTBL
EC05                     .BYTE $00                                   ;0 INVALID
EC06                     .BYTE $00                                   ;1 INVALID
EC07                     .BYTE $00                                   ;2 INVALID
EC08                     .BYTE $00                                   ;3 INVALID
EC09                     .BYTE $00                                   ;4 INVALID
EC0A                     .BYTE $05                                   ;5 SE
EC0B                     .BYTE $04                                   ;6 NE
EC0C                     .BYTE $02                                   ;7 E
EC0D                     .BYTE $00                                   ;8 INVALID
EC0E                     .BYTE $06                                   ;9 SW
EC0F                     .BYTE $07                                   ;A NW
EC10                     .BYTE $03                                   ;B W
EC11                     .BYTE $00                                   ;C INVALID
EC12                     .BYTE $01                                   ;D S
EC13                     .BYTE $00                                   ;E N
EC14                     .BYTE $00                                   ;F NONE

;UNKNOWN USAGE ($EC15-$EC1C)
EC15                     .BYTE $0E
EC16                     .BYTE $0D
EC17                     .BYTE $07
EC18                     .BYTE $0B
EC19                     .BYTE $06
EC1A                     .BYTE $05
EC1B                     .BYTE $09
EC1C                     .BYTE $0A

*
**********************************************************************************
*                                                                                *
*         X,YDIRTBL - RETURN X AND Y INCREMENTS GIVEN A DIRECTION ( 0 - 7 )      *
*                                                                                 *
***********************************************************************************
*
XDIRTBL:
EC1D                    .BYTE $00                                   ;N
EC1E                    .BYTE $00                                   ;S
EC1F                    .BYTE $01                                   ;E
EC20                    .BYTE $FF                                   ;W
EC21                    .BYTE $01                                   ;NE
EC22                    .BYTE $01                                   ;SE
EC23                    .BYTE $FF                                   ;SW
EC24                    .BYTE $FF                                   ;NW

YDIRTBL:
EC25                    .BYTE $FE                                   ;N
EC26                    .BYTE $02                                   ;S
EC27                    .BYTE $00                                   ;E
EC28                    .BYTE $00                                   ;W
EC29                    .BYTE $FE                                   ;NE
EC2A                    .BYTE $02                                   ;SE
EC2B                    .BYTE $02                                   ;SW
EC2C                    .BYTE $FE                                   ;NW

;UNKNOWN USAGE ($EC2D-$EC3C)
EC2D                    .BYTE $00
EC2E                    .BYTE $00
EC2F                    .BYTE $02
EC30                    .BYTE $FE
EC31                    .BYTE $02
EC32                    .BYTE $02
EC33                    .BYTE $FE
EC34                    .BYTE $FE
EC35                    .BYTE $FC
EC36                    .BYTE $04
EC37                    .BYTE $00
EC38                    .BYTE $00
EC39                    .BYTE $FC
EC3A                    .BYTE $04
EC3B                    .BYTE $04
EC3C                    .BYTE $FC

*
************************************************************************************
*                                                                                  *
*       X,YDIRTBL4  -- FOR ANY OF THE 4-BIT DIR CODES, RETURN X OR Y INCREMENT      *
*                                                                                  *
************************************************************************************
*
XDIRTBL4:
EC3D                    .BYTE $00                                   ;0 INVALID
EC3E                    .BYTE $00                                   ;1 INVALID
EC3F                    .BYTE $00                                   ;2 INVALID
EC40                    .BYTE $00                                   ;3 INVALID
EC41                    .BYTE $00                                   ;4 INVALID
EC42                    .BYTE $01                                   ;5 SE
EC43                    .BYTE $01                                   ;6 NE
EC44                    .BYTE $01                                   ;7 E
EC45                    .BYTE $00                                   ;8 INVALID
EC46                    .BYTE $FF                                   ;9 SW
EC47                    .BYTE $FF                                   ;A NW
EC48                    .BYTE $FF                                   ;B W
EC49                    .BYTE $00                                   ;C INVALID
EC4A                    .BYTE $00                                   ;D S
EC4B                    .BYTE $00                                   ;E N
EC4C                    .BYTE $00                                   ;F INVALID

YDIRTBL4:
EC4D                    .BYTE $00                                   ;0 INVALID
EC4E                    .BYTE $00                                   ;1 INVALID
EC4F                    .BYTE $00                                   ;2 INVALID
EC50                    .BYTE $00                                   ;3 INVALID
EC51                    .BYTE $00                                   ;4 INVALID
EC52                    .BYTE $02                                   ;5 SE
```

```
EC53                    .BYTE $FE                                     ;6 NE
EC54                    .BYTE $00                                     ;7 E
EC55                    .BYTE $00                                     ;8 INVALID
EC56                    .BYTE $02                                     ;9 SW
EC57                    .BYTE $FE                                     ;A NW
EC58                    .BYTE $00                                     ;B W
EC59                    .BYTE $00                                     ;C INVALID
EC5A                    .BYTE $02                                     ;D S
EC5B                    .BYTE $FE                                     ;E N
EC5C                    .BYTE $00                                     ;F INVALID

;UNKNOWN USAGE ($EC5D-$EF19), GRAPHICS DATA (PROBABLY) & TUNES DATA (MAYBE)
EC5D                    .BYTE $00,$00,$00
EC60                    .BYTE $FD,$00,$00,$00,$00,$FC,$00,$04,$04,$03,$00,$04,$00,$00,$00,$FD
EC70                    .BYTE $00,$00,$00,$00,$FC,$00,$FC,$00,$00,$00,$00,$00,$04,$05,$05,$05
EC80                    .BYTE $FB,$00,$00,$00,$00,$FB,$00,$00,$00,$00,$FB,$00,$00,$00,$00,$00
EC90                    .BYTE $0B,$16,$21,$2C,$01,$00,$03,$02,$06,$07,$04,$05,$00,$00,$03,$FD
ECA0                    .BYTE $03,$03,$FD,$FD,$FB,$05,$00,$00,$FB,$05,$05,$FB,$30,$65,$05,$51
ECB0                    .BYTE $5A,$1A,$11,$3B,$03,$63,$0E,$17,$09,$12,$16,$18,$63,$03,$01,$05
ECC0                    .BYTE $00,$02,$04,$06,$05,$17,$10,$63,$0B,$14,$19,$1B,$01,$0E,$63,$10
ECD0                    .BYTE $07,$0D,$0F,$11,$04,$16,$0F,$19,$0A,$13,$63,$1A,$06,$18,$11,$1B
ECE0                    .BYTE $0C,$15,$1A,$63,$00,$09,$07,$0B,$63,$08,$0A,$0C,$02,$12,$0D,$14
ECF0                    .BYTE $08,$63,$13,$15,$00,$09,$01,$02,$03,$04,$05,$06,$09,$07,$01,$02

ED00                    .BYTE $03,$04,$05,$06,$00,$07,$01,$09,$03,$04,$05,$06,$00,$07,$09,$02
ED10                    .BYTE $03,$04,$05,$06,$00,$07,$01,$02,$03,$04,$09,$06,$00,$07,$01,$02
ED20                    .BYTE $03,$04,$05,$09,$00,$07,$01,$02,$09,$04,$05,$06,$00,$07,$01,$02
ED30                    .BYTE $03,$09,$05,$06,$07,$07,$07,$05,$03,$06,$07,$0F,$09,$09,$09,$09
ED40                    .BYTE $0A,$0A,$0A,$07,$02,$03,$04,$0A,$10,$07,$06,$01,$09,$05,$01,$91
ED50                    .BYTE $01,$91,$1A,$1A,$AA,$AA,$00,$01,$01,$00,$02,$FE,$02,$FE,$02,$02
ED60                    .BYTE $FE,$FE,$00,$00,$02,$04,$02,$12,$10,$0E,$0C,$0E,$00,$0A,$14,$1E
ED70                    .BYTE $14,$05,$05,$03,$03,$01,$20,$20,$20,$20,$1E,$1E,$1E,$1C,$1C,$1C
ED80                    .BYTE $1C,$1A,$1A,$18,$18,$18,$16,$16,$16,$14,$14,$14,$12,$12,$12,$12
ED90                    .BYTE $12,$10,$10,$10,$0E,$0E,$0E,$0E,$0C,$0C,$0C,$0C,$0C,$0A,$0A,$0A
EDA0                    .BYTE $0A,$0A,$0A,$0A,$0A,$0A,$0A,$0A,$09,$09,$09,$09,$09,$09,$09,$09
EDB0                    .BYTE $09,$09,$09,$09,$09,$09,$09,$09,$09,$09,$09,$09,$16,$16,$16,$16
EDC0                    .BYTE $14,$14,$14,$14,$13,$13,$12,$12,$12,$12,$10,$10,$10,$0E,$0E,$0C
EDD0                    .BYTE $0B,$0B,$0B,$0B,$0A,$0A,$0A,$0A,$0A,$0A,$09,$09,$09,$09,$09,$09
EDE0                    .BYTE $09,$09,$09,$09,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08,$08
EDF0                    .BYTE $08,$08,$08,$08,$07,$07,$07,$07,$07,$07,$07,$07,$07,$07,$07,$07

EE00                    .BYTE $07,$07,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
EE10                    .BYTE $00,$00,$00,$00,$00,$00,$00,$01,$01,$01,$00,$02,$03,$02,$02,$00
EE20                    .BYTE $02,$04,$02,$04,$00,$02,$03,$03,$02,$02,$02,$04,$03,$04,$01,$03
EE30                    .BYTE $04,$03,$03,$01,$03,$04,$03,$05,$02,$03,$04,$03,$03,$02,$04,$04
EE40                    .BYTE $04,$07,$02,$04,$06,$04,$05,$02,$00,$05,$05,$05,$05,$05,$05,$05
EE50                    .BYTE $05,$05,$00,$04,$04,$04,$04,$04,$04,$04,$04,$04,$04,$04,$04,$03
EE60                    .BYTE $03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03
EE70                    .BYTE $03,$02,$03,$02,$03,$03,$02,$02,$02,$02,$02,$02,$02,$02,$02,$02
EE80                    .BYTE $02,$02,$02,$02,$02,$02,$02,$02,$02,$02,$02,$02,$02,$02,$06,$06
EE90                    .BYTE $06,$06,$06,$06,$06,$06,$06,$06,$06,$06,$06,$06,$06,$05,$05,$05
EEA0                    .BYTE $05,$05,$04,$04,$04,$04,$04,$04,$04,$04,$04,$04,$04,$04,$04,$04
EEB0                    .BYTE $04,$04,$04,$04,$04,$04,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03
EEC0                    .BYTE $03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03,$03
EED0                    .BYTE $03,$03,$03,$03,$00,$00,$00,$00,$0F,$00,$00,$00,$00,$0E,$00,$00
EEE0                    .BYTE $00,$00,$0D,$00,$00,$00,$00,$0C,$00,$00,$00,$00,$0A,$00,$00,$00
EEF0                    .BYTE $00,$09,$00,$00,$00,$00,$08,$00,$00,$00,$00,$08,$00,$00,$00,$00

EF00                    .BYTE $07,$00,$00,$00,$00,$07,$00,$00,$00,$00,$06,$00,$00,$00,$00,$06
EF10                    .BYTE $00,$00,$00,$00,$06,$00,$00,$00,$00,$06

*
*********************************************************************************
*                                                                               *
*         WAVETBL -- NUMBERS OF EACH OBJECT TO ALLOCATE FOR EACH WAVE           *
*                                                                               *
*         THERE IS A BLOCK OF SIXTEEN BYTES FOR EACH WAVE, AS FOLLOWS:          *
*                                                                               *
*   G, Mo, D, Mi, H, S, Q, E, T, B, P, Enf.M, Cr.M, TankM, ES, ELECTRODES       *
*                                                                               *
*********************************************************************************
*
;G=GRUNT,MO=MOMMY,D=DADDY,Mi=MIKEY,H=HULK,S=SPHEROID,Q=QUARK,E=ENFORCER,T=TANK
```

```
;B=BRAIN,P=PROG,ENF.M=ENFORCER MISSILE,Cr.M=CRUISE MISSILE,TankM=TANK MISSILE,
;ES=ELECTRODE STYLE, EL=ELECTRODE

WAVETBL:
               ;THIS IS JUST ANOTHER SYMBOL FOR WAVETBL - FOR EASY TYPING

*           G   Mo  D   Mi  H   S   Q   E   T   B   P   EM  CM  TM  ES  EL
EF1A    .BYTE $0F,$01,$01,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$05  ;WAVE 01
EF2A    .BYTE $11,$01,$01,$01,$05,$01,$00,$08,$00,$00,$00,$00,$00,$00,$01,$0D  ;WAVE 02
EF3A    .BYTE $16,$02,$02,$02,$06,$03,$00,$08,$00,$00,$00,$00,$00,$00,$03,$12  ;WAVE 03
EF4A    .BYTE $22,$02,$02,$02,$07,$04,$00,$08,$00,$00,$00,$00,$00,$00,$05,$14  ;WAVE 04
EF5A    .BYTE $14,$0E,$00,$01,$00,$01,$00,$08,$00,$0F,$00,$00,$00,$00,$04,$0F  ;WAVE 05
EF6A    .BYTE $1E,$03,$03,$03,$07,$04,$00,$08,$00,$00,$00,$00,$00,$00,$02,$14  ;WAVE 06
EF7A    .BYTE $00,$04,$04,$04,$0B,$00,$0B,$00,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 07
EF8A    .BYTE $23,$03,$02,$03,$08,$05,$00,$08,$00,$00,$00,$00,$00,$00,$06,$0C  ;WAVE 08
EF9A    .BYTE $30,$03,$03,$03,$04,$05,$00,$08,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 09
EFAA    .BYTE $18,$00,$14,$00,$00,$01,$00,$08,$00,$10,$00,$00,$00,$00,$07,$05  ;WAVE 10
EFBA    .BYTE $23,$03,$03,$03,$08,$05,$00,$08,$00,$00,$00,$00,$00,$00,$01,$08  ;WAVE 11
EFCA    .BYTE $00,$03,$03,$03,$0D,$00,$0D,$00,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 12
EFDA    .BYTE $23,$03,$03,$03,$08,$05,$00,$08,$00,$00,$00,$00,$00,$00,$00,$08  ;WAVE 13
EFEA    .BYTE $1A,$05,$05,$05,$12,$02,$00,$08,$00,$00,$00,$00,$00,$00,$01,$0A  ;WAVE 14
EFFA    .BYTE $1B,$00,$00,$16,$00,$01,$00,$08,$00,$15,$00,$00,$00,$00,$07,$05  ;WAVE 15
F00A    .BYTE $22,$03,$03,$03,$04,$05,$00,$08,$00,$00,$00,$00,$00,$00,$02,$0A  ;WAVE 16
F01A    .BYTE $00,$03,$03,$03,$0B,$00,$11,$00,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 17
F02A    .BYTE $21,$03,$03,$03,$08,$05,$00,$08,$00,$00,$00,$00,$00,$00,$03,$0F  ;WAVE 18
F03A    .BYTE $42,$03,$03,$03,$03,$04,$00,$08,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 19
F04A    .BYTE $00,$08,$08,$07,$02,$02,$00,$08,$00,$11,$00,$00,$00,$00,$07,$0A  ;WAVE 20
F05A    .BYTE $23,$03,$03,$03,$08,$05,$00,$08,$00,$00,$00,$00,$00,$00,$00,$14  ;WAVE 21
F06A    .BYTE $00,$03,$03,$03,$0D,$00,$0C,$00,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 22
F07A    .BYTE $23,$03,$03,$03,$08,$05,$00,$08,$00,$00,$00,$00,$00,$00,$03,$14  ;WAVE 23
F08A    .BYTE $00,$03,$03,$03,$0D,$06,$07,$08,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 24
F09A    .BYTE $00,$16,$00,$01,$01,$01,$00,$08,$00,$14,$00,$00,$00,$00,$04,$14  ;WAVE 25
F0AA    .BYTE $1E,$03,$03,$03,$08,$05,$00,$08,$00,$00,$00,$00,$00,$00,$02,$14  ;WAVE 26
F0BA    .BYTE $00,$03,$03,$03,$0D,$00,$0C,$00,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 27
F0CA    .BYTE $23,$03,$03,$03,$08,$05,$01,$08,$00,$00,$00,$00,$00,$00,$06,$14  ;WAVE 28
F0DA    .BYTE $3F,$03,$03,$03,$04,$05,$01,$08,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 29
F0EA    .BYTE $00,$00,$17,$00,$01,$01,$01,$08,$00,$14,$00,$00,$00,$00,$07,$0A  ;WAVE 30
F0FA    .BYTE $23,$03,$03,$03,$08,$05,$01,$08,$00,$00,$00,$00,$00,$00,$00,$14  ;WAVE 31
F10A    .BYTE $00,$03,$03,$03,$0C,$00,$0D,$00,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 32
F11A    .BYTE $23,$03,$03,$03,$08,$05,$01,$08,$00,$00,$00,$00,$00,$00,$03,$14  ;WAVE 33
F12A    .BYTE $1E,$03,$03,$03,$14,$02,$02,$08,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 34
F13A    .BYTE $00,$00,$00,$17,$02,$01,$02,$08,$00,$17,$00,$00,$00,$00,$04,$0F  ;WAVE 35
F14A    .BYTE $23,$03,$03,$03,$08,$05,$02,$08,$00,$00,$00,$00,$00,$00,$02,$14  ;WAVE 36
F15A    .BYTE $00,$03,$03,$03,$0C,$00,$0E,$00,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 37
F16A    .BYTE $23,$03,$03,$03,$08,$05,$02,$08,$00,$00,$00,$00,$00,$00,$06,$14  ;WAVE 38
F17A    .BYTE $3D,$03,$03,$03,$06,$05,$01,$08,$00,$00,$00,$00,$00,$00,$00,$00  ;WAVE 39
F18A    .BYTE $00,$09,$09,$09,$02,$01,$01,$08,$00,$17,$00,$00,$00,$00,$07,$0A  ;WAVE 40

***************************************************************
*                                                             *
*          SCORETBL -- SCORE FOR EACH OBJECT WHEN SHOT        *
*                                                             *
***************************************************************
*
SCORETBL:
F19A                    .BYTE $00,$00                           ;OBJECT ZERO IS NULL
F19C                    .BYTE $01,$00                           ;1 GRUNT  - 100
F19E                    .BYTE $00,$00                           ;2 MOMMY  - NOT APPLICABLE
HERE
F1A0                    .BYTE $00,$00                           ;3 DADDY  - NOT APPLICABLE
HERE
F1A2                    .BYTE $00,$00                           ;4 MIKEY  - NOT APPLICABLE
HERE
F1A4                    .BYTE $00,$00                           ;5 HULK   - CAN'T BE DESTROYED
F1A6                    .BYTE $10,$00                           ;6 SPHEROID  - 1000
F1A8                    .BYTE $10,$00                           ;7 QUARK     - 1000
F1AA                    .BYTE $01,$50                           ;8 ENFORCER - 150
F1AC                    .BYTE $02,$00                           ;9 TANK - 200
F1AE                    .BYTE $05,$00                           ;A BRAIN  - 500
F1B0                    .BYTE $01,$00                           ;B PROG   - 100
F1B2                    .BYTE $00,$25                           ;C ENFORCER MISSILE - 25
F1B4                    .BYTE $00,$25                           ;D CRUISE MISSILE - 25
F1B6                    .BYTE $00,$25                           ;E TANK MISSILE - 25
```

```
;UNKNOWN USAGE ($F1B8-$F1BB)
F1B8                    .BYTE $00,$00,$00,$00

*
*
******************************************
*                                        *
*           ZONE DATA TABLES             *
*                                        *
******************************************
*
ZONDLAL:
F1BC                    .BYTE $00                                  ;ZONE DISPLAY LIST LOW
ADDRESSES
F1BD                    .BYTE $80
F1BE                    .BYTE $00
F1BF                    .BYTE $80
F1C0                    .BYTE $00
F1C1                    .BYTE $80                                  ;6
F1C2                    .BYTE $00
F1C3                    .BYTE $80
F1C4                    .BYTE $00
F1C5                    .BYTE $80
F1C6                    .BYTE $00
F1C7                    .BYTE $80                                  ;12
*
ZONDLAH:
F1C8                    .BYTE $22                                  ;ZONE DISPLAY LIST ADDRESSES
HIGH
F1C9                    .BYTE $22
F1CA                    .BYTE $23
F1CB                    .BYTE $23
F1CC                    .BYTE $24
F1CD                    .BYTE $24                                  ;6
F1CE                    .BYTE $25
F1CF                    .BYTE $25
F1D0                    .BYTE $26
F1D1                    .BYTE $26
F1D2                    .BYTE $27
F1D3                    .BYTE $27                                  ;12
*
ZONLINE:
F1D4                    .BYTE $00 (#00)                            ;ZONE START LINE NUMBER (YPOS)
F1D5                    .BYTE $10 (#16)
F1D6                    .BYTE $20 (#32)
F1D7                    .BYTE $30 (#48)
F1D8                    .BYTE $40 (#64)
F1D9                    .BYTE $50 (#80)                            ;6
F1DA                    .BYTE $60 (#96)
F1DB                    .BYTE $70 (#112)
F1DC                    .BYTE $80 (#128)
F1DD                    .BYTE $90 (#144)
F1DE                    .BYTE $A0 (#160)
F1DF                    .BYTE $B0 (#176)                           ;12 (TO 192)
*
ZONOBJLL:
F1E0                    .BYTE $4F                                  ;ZONE OBJECT LIST ADDRESSES
(LOW)
F1E1                    .BYTE $6F
F1E2                    .BYTE $8F
F1E3                    .BYTE $AF
F1E4                    .BYTE $CF
F1E5                    .BYTE $EF                                  ;6
F1E6                    .BYTE $0F
F1E7                    .BYTE $2F
F1E8                    .BYTE $4F
F1E9                    .BYTE $6F
F1EA                    .BYTE $8F
F1EB                    .BYTE $AF                                  ;12
*
ZONOBJLH:
F1EC                    .BYTE $19                                  ;ZONE OBJECT LIST ADDRESSES
```

```
(HIGH)
F1ED                    .BYTE $19
F1EE                    .BYTE $19
F1EF                    .BYTE $19
F1F0                    .BYTE $19
F1F1                    .BYTE $19                                   ;6
F1F2                    .BYTE $1A
F1F3                    .BYTE $1A
F1F4                    .BYTE $1A
F1F5                    .BYTE $1A
F1F6                    .BYTE $1A
F1F7                    .BYTE $1A                                   ;12

**************************************************************************
*                                                                        *
*    FREEMSK  --  MASKS FOR SETTING/UNSETTING DLIST FREE LIST ENTRIES    *
*                                                                        *
**************************************************************************
*
;Disassembly of $F1F8-$F1FF compliments of Dan Boris & "Scotty"
FREEMSK:
;Display List entries

;$2253 Score Line
F1F8                    .BYTE $00,$60,$21,$F9,$0C
         ;Data=$2100,Indirect,Pal7,Width8,HPOS12
F1FD                    .BYTE $0E,$60,$21,$B9,$30
         ;Data=$210E,Indirect,Pal5,Width8,HPOS48
F202                    .BYTE $07,$60,$21,$F9,$58
         ;Data=$2107,Indirect,Pal7,Width8,HPOS88
F207                    .BYTE $15,$60,$21,$B9,$78
         ;Data=$2115,Indirect,Pal5,Width8,HPOS120
F20C                    .BYTE $00,$00                                     ;End

;$226C Level line
F20E                    .BYTE $1C,$60,$21,$FE,$3C
         ;Data=$211C,Indirect,Pal7,Width3,HPOS60
F213                    .BYTE $1E,$60,$21,$FC,$48
         ;Data=$211E,Indirect,Pal7,Width5,HPOS72
F218                    .BYTE $23,$60,$21,$FE,$5D
         ;Data=$2123,Indirect,Pal7,Width3,HPOS93
F21D                    .BYTE $00,$00                                     ;End

;Top 4 DLL entries
F21F                    .BYTE $0F,$18,$00                           ;16 lines of $1800
F222                    .BYTE $07,$18,$00                           ;8 lines of $1800
F225                    .BYTE $07,$22,$53                           ;8 lines of $2253
F228                    .BYTE $07,$22,$6C                           ;8 lines of $226C

;Bottom 3 DLL entries
F22B                    .BYTE $8F,$18,$00                           ;16 Lines of $1800 with DLI
F22E                    .BYTE $07,$18,$00                           ;8 Lines of $1800
F231                    .BYTE $82,$18,$00                           ;3 Lines of $1800 with DLI

;UNKNOWN USAGE ($F234-$F28F)
F234                    .BYTE $00,$AF,$80
F237                    .BYTE $08,$11,$AD

F23A                    .BYTE $80,$4C,$00,$00,$24,$AF,$80,$08,$35
F243                    .BYTE $AD,$80,$4C,$00,$00,$48,$CE,$80,$2D

F24C                    .BYTE $00,$00,$5A,$CE,$80,$2D,$00,$00,$6C,$DD,$80,$48,$00,$00,$CF,$18

F25C                    .BYTE $00,$4F,$22,$6C
F260                    .BYTE $4F,$22,$76,$4F,$18,$00,$4F,$22,$80,$4F,$22,$86,$4F,$22,$8C,$47
F270                    .BYTE $18,$00,$86,$8D,$9C,$A6,$AE,$D9,$D1,$D6,$D8,$D5,$41,$33,$3B,$40
F280                    .BYTE $3A,$20,$84,$00,$01,$00,$00,$00,$00,$00,$1C,$04,$18,$19,$D6,$04

;Pointer table
F290                    .BYTE $56,$00,$AA,$54,$FE,$A8,$52,$FC,$A6,$50,$AC,$02,$58,$AE,$04,$5A
F2A0                    .BYTE $B0,$06,$5C,$B2,$00

F2A5                    .BYTE $1C,$1C,$1B,$1B,$1A,$1A,$1A,$19,$19,$19,$1C,$1D,$1D,$1D,$1E,$1E
```

```
F2B5                     .BYTE $1E,$1F,$1F,$1F,$18

;Pointers into address table at $F290
F2BA                     .BYTE $14,$09,$08,$07,$06,$05,$04,$03,$02,$01,$00,$0A,$0B,$0C,$0D
F2C9                     .BYTE $0E,$0F,$10,$11,$12,$13,$14

;UNKNOWN USAGE ($F2D0-$F31C)
F2D0                     .BYTE $D4,$E4,$F2,$F2,$88,$D6,$88,$D8,$88,$DA,$88,$DC,$80,$D8,$80,$D6
F2E0                     .BYTE $80,$DA,$80,$DC,$80,$DE,$80,$DE,$80,$DE,$80,$DE,$80,$DE,$80,$DE
F2F0                     .BYTE $80,$DE,$80,$DE,$50,$58,$60,$68,$70,$78,$80,$88,$90,$98,$48,$40

F300                     .BYTE $38,$30,$28,$20,$18,$10,$08,$00,$00,$02,$04,$06,$08,$0A,$0C,$0E
F310                     .BYTE $10,$12,$14,$00,$08,$10,$18,$20,$28,$30,$38,$40,$48

******************************************
******************************************
*                                        *
*         ROBOTRON CODE:                 *
*                                        *
*            MAIN ROUTINES               *
*                                        *
******************************************


******************************************
*
*         VECTORS
*
          ORG     $FFFC
          DB      L(INIT),H(INIT)
          DB      L(KERNEL),H(KERNEL)
*
          ORG     $F31D                  ;START OF PROGRAM ROM

***********************************************************************
*                                                                     *
*       INIT    --  FIRST ROUTINE IN CARTRIDGE - SETS UP SYSTEM       *
*                                                                     *
***********************************************************************
*
;Disassembly of $F31D-$F97D compliments of Dan Boris & "Scotty"
INIT:
F31D     78             SEI
F31E     D8             CLD
F31F     A9 17          LDA #$17
F321     85 01          STA $01                                     ;Lock into 7800 mode
F323     A9 7F          LDA #$7F
F325     85 3C          STA $3C                                     ;Turn off DMA
F327     A2 FF          LDX $FF                                     ;Set stack pointer
F329     9A             TXS

F32A     A9 18          LDA #$18                                    ;Set pointer to start of RAM
F32C     85 B8          STA TEMPX
F32E     A9 00          LDA #$00
F330     85 B7          STA FRMCNT
F332     A9 00          LDA #$00
F334     A2 08          LDX $08                                     ;Clear 8 pages
F336     A0 00          LDY $00
F338     91 B7          STA (FRMCNT),Y                              ;Clear RAM
F33A     88             DEY
F33B     D0 FB          BNE $F338                                   ;End of page?
F33D     E6 B8          INC TEMPX                                         ;Next page
F33F     CA             DEX                                               ;Count pages
F340     D0 F6          BNE $F338                                   ;Last page?

F342     A9 00          LDA #$00                                    ;Clear Zero page
F344     A2 C0          LDX $C0
F346     95 3F          STA $3F,X
F348     CA             DEX
F349     D0 FB          BNE $F346

F34B     A9 00          LDA #$00                                    ;Clear $2000-$203F
F34D     A2 40          LDX $40
```

```
F34F      9D FF 1F        STA $1FFF,X
F352      CA              DEX
F353      D0 FA           BNE $F34F

F355      A9 00           LDA #$00                              ;Clear $3000-$303F
F357      A2 40           LDX $40
F359      9D FF 20        STA $20FF,X
F35C      CA              DEX
F35D      D0 FA           BNE $F359
;
;Clear $2200 - $27FF
;
F35F      A9 22           LDA #$22                              ;Set pointer to $2200
F361      85 B8           STA TEMPX
F363      A9 00           LDA #$00
F365      85 B7           STA FRMCNT
F367      A9 00           LDA #$00
F369      A2 06           LDX $06                               ;Clear 6 pages
F36B      A0 00           LDY $00
F36D      91 B7           STA (FRMCNT),Y                        ;Clear RAM
F36F      88              DEY
F370      D0 FB           BNE $F36D                             ;End of page?
F372      E6 B8           INC TEMPX                                   ;Next page
F374      CA              DEX
F375      D0 F6           BNE $F36D                             ;Last page?
;
;Setup zero page variable
;
F377      A9 00           LDA #$00                              ;
F379      85 6C           STA NUMPLAYERS                        ;Set number of players to 1
F37B      85 69           STA $69
F37D      85 74           STA $74
F37F      85 01           STA $01
F381      A9 01           LDA #$01                              ;
F383      85 64           STA SKILL                             ;Set skill level to 1
F385      A9 01           LDA #$01
F387      85 67           STA $67
;
F389      20 BA D6        JSR MCSEND_$D6BA
F38C      20 86 DC        JSR $DC86                             ;Setup DLL and DLs
F38F      20 9A DD        JSR $DD9A
F392      20 86 DC        JSR $DC86

F395      A2 FF           JSR #$FF
F397      9A              TXS
F398      A9 01           LDA #$01
F39A      85 67           STA $67
F39C      A9 00           LDA #$00
F39E      85 E3           STA $E3
F3A0      A9 36           LDA #$36
F3A2      85 68           STA $68
F3A4      20 56 E3        JSR $E356
F3A7      20 30 D5        JSR $D530
F3AA      20 86 DC        JSR $DC86
F3AD      20 91 D6        JSR $D691
F3B0      A9 0F           LDA #$0F
F3B2      85 63           STA $63
F3B4      C6 63           DEC $63
F3B6      10 1D           BPL $F3D5
F3B8      A2 30           LDX $30
F3BA      A9 18           LDA #$18
F3BC      9D 05 18        STA $1805,X
F3BF      A9 00           LDA #$00
F3C1      9D 06 18        STA $1806,X
F3C4      CA              DEX
F3C5      CA              DEX
F3C6      CA              DEX
F3C7      10 F1           BPL $F3BA
F3C9      A9 00           LDA #$00
F3CB      85 A6           STA TEMP6
F3CD      20 19 F5        JSR $F519
F3D0      F0 FB           BEQ $F3CD
F3D2      4C 95 F3        JMP $F395
```

```
F3D5     20 52 F5       JSR $F552
F3D8     20 BE F5       JSR $F5BE
F3DB     20 08 F6       JSR $F608
F3DE     A9 1F          LDA #$1F
F3E0     85 A1          STA TEMP1
F3E2     A9 00          LDA #$00
F3E4     85 A2          STA TEMP2
F3E6     A9 02          LDA #$02
F3E8     85 A3          STA TEMP3
F3EA     20 BB F6       JSR $F6BB
F3ED     20 19 F5       JSR $F519
F3F0     D0 03          BNE $F3F5
F3F2     4C 7E F4       JMP $F47E
F3F5     A5 A1          LDA TEMP1
F3F7     F0 03          BEQ $F3FC
F3F9     4C 7E F4       JMP $F47E
F3FC     A9 00          LDA #$00
F3FE     85 A2          STA TEMP2
F400     A9 04          LDA #$04
F402     85 A3          STA TEMP3
F404     A9 0F          LDA #$0F
F406     85 A1          STA TEMP1
F408     A5 0C          LDA $0C
F40A     10 15          BPL $F421
F40C     A5 0D          LDA $0D
F40E     10 11          BPL $F421
F410     AD 82 02       LDA $0282                              ;Read console switches
F413     29 01          AND #$01                               ;Mask off reset
F415     D0 11          BNE $F428                              ;Branch if no reset
F417     20 38 F5       JSR $F538
F41A     AD 82 02       LDA $0282                              ;Read console switched
F41D     29 01          AND #$01                               ;mask off reset
F41F     F0 F6          BEQ $F417                              ;Wait for it to be released
F421     A9 01          LDA #$01
F423     85 74          STA $74
F425     4C 03 90       JMP $9003

F428     AD 82 02       LDA $0282                              ;Read console switches
F42B     29 02          AND #$02                               ;mask off select switch
F42D     D0 12          BNE $F441                              ;Branch if not pushed
F42F     E6 64          INC SKILL                              ;Next skill level
F431     A5 64          LDA SKILL                              ;Read it
F433     C9 05          CMP #$05                               ; > 5
F435     30 0A          BMI $F441                              ;Branch if not
F437     A9 00          LDA #$00                               ;Cycle to first skill level
F439     85 64          STA SKILL                              ;Store it
F43B     A5 6C          LDA NUMPLAYERS                         ;Read number of players
F43D     49 01          EOR #$01                               ;Invert it
F43F     85 6C          STA NUMPLAYERS                         ;Store it

F441     AD 80 02       LDA $0280                              ;Read Joystick
F444     6A             ROR A                                  ;Move left stick to bottom 4
bits
F445     6A             ROR A                                  ;
F446     6A             ROR A                                  ;
F447     6A             ROR A                                  ;
F448     29 0F          AND #$0F                               ;Mask it off
F44A     AA             TAX                                    ;
F44B     BD 4D EC       LDA YDIRTBL4_$EC4D,X                   ;Read from stick table
F44E     F0 0D          BEQ $F45D                              ;Branch if not pushed up or
down
F450     A9 00          LDA #$00                               ;
F452     85 6C          STA NUMPLAYERS                         ;Set number of players to 1
F454     BD 4D EC       LDA YDIRTBL4_$EC4D,X                   ;Read stick table
F457     10 04          BPL $F45D                              ;Stick Pushed up?
F459     A9 01          LDA #$01                               ;Set number of players to 2
F45B     85 6C          STA NUMPLAYERS                         ;

F45D     A5 64          LDA SKILL                              ;Read skill level
F45F     18             CLC                                    ;
F460     7D 3D EC       ADC XDIRTBL4_$EC3D,X                   ;Change skill level
F463     85 64          STA SKILL                              ;
F465     A5 64          LDA SKILL                              ;
```

```
F467     10 02          BPL $F46B                                     ;
F469     E6 64          INC SKILL                                     ;
F46B     C9 05          CMP #$05                                      ;
F46D     30 02          BMI $F471                                     ;
F46F     C6 64          DEC SKILL                                     ;
F471     20 BE F5       JSR $F5BE
F474     20 20 D4       JSR RESETSC_$D420
F477     20 BA D6       JSR MCSEND_$D6BA
F47A     A9 00          LDA #$00
F47C     85 74          STA $74
F47E     C6 A1          DEC TEMP1
F480     10 04          BPL $F486
F482     A9 00          LDA #$00
F484     85 A1          STA TEMP1
F486     A5 A2          LDA TEMP2
F488     D0 04          BNE $F48E
F48A     C6 A3          DEC TEMP3
F48C     30 08          BMI $F496
F48E     C6 A2          DEC TEMP2
F490     20 38 F5       JSR $F538
F493     4C EA F3       JMP $F3EA
F496     AD 00 39       LDA $3900
F499     C9 C6          CMP #$C6
F49B     D0 07          BNE $F4A4
F49D     AD 04 39       LDA $3904
F4A0     C9 FE          CMP #$FE
F4A2     F0 03          BEQ $F4A7
F4A4     4C 03 F5       JMP $F503
F4A7     A9 01          LDA #$01
F4A9     85 69          STA $69
F4AB     A5 64          LDA SKILL
F4AD     85 A5          STA TEMP5
F4AF     A9 00          LDA #$00
F4B1     85 64          STA SKILL
F4B3     20 41 F6       JSR $F641
F4B6     A5 64          LDA SKILL
F4B8     0A             ASL A
F4B9     0A             ASL A
F4BA     8D 02 1A       STA $1A02
F4BD     A0 1A          LDY $1A
F4BF     A9 00          LDA #$00
F4C1     8D 0E 1A       STA $1A0E
F4C4     20 F7 3F       JSR $3FF7
F4C7     30 25          BMI $F4EE
F4C9     A9 7F          LDA #$7F                                      ;Turn off DMA
F4CB     85 3C          STA $3C                                       ;
F4CD     20 AE F6       JSR $F6AE
F4D0     20 41 F6       JSR $F641
F4D3     A5 64          LDA SKILL
F4D5     0A             ASL A
F4D6     0A             ASL A
F4D7     8D 02 1A       STA $1A02
F4DA     A0 1A          LDY $1A
F4DC     20 FA 3F       JSR $3FFA
F4DF     A5 0C          LDA $0C
F4E1     10 08          BPL $F4EB
F4E3     A5 0D          LDA $0D
F4E5     30 07          BMI $F4EE
F4E7     A5 A5          LDA TEMP5
F4E9     85 64          STA SKILL
F4EB     4C 03 90       JMP $9003
F4EE     E6 64          INC SKILL
F4F0     A5 64          LDA SKILL
F4F2     C9 05          CMP #$05
F4F4     90 BD          BCC $F4B3
F4F6     20 45 F5       JSR $F545
F4F9     A9 00          LDA #$00
F4FB     85 A6          STA TEMP6
F4FD     85 69          STA $69
F4FF     A5 A5          LDA TEMP5
F501     85 64          STA SKILL
F503     A9 01          LDA #$01
F505     85 E3          STA $E3
```

```
F507      85 6A          STA CURRENTOBJ_6A
F509      85 6B          STA $6B
F50B      A9 00          LDA #$00
F50D      85 62          STA CURPLAYERS
F50F      4C 07 90       JMP $9007
F512      A9 00          LDA #$00
F514      85 E3          STA $E3
F516      4C 95 F3       JMP $F395
F519      AD 82 02       LDA $0282
F51C      49 FF          EOR #$FF
F51E      29 03          AND #$03
F520      D0 15          BNE $F537
F522      AD 80 02       LDA $0280
F525      49 FF          EOR #$FF
F527      D0 0E          BNE $F537
F529      A5 0C          LDA $0C
F52B      49 FF          EOR #$FF
F52D      29 80          AND #$80
F52F      D0 06          BNE $F537
F531      A5 0D          LDA $0D
F533      49 FF          EOR #$FF
F535      29 80          AND #$80
F537      60             RTS

F538      85 24          STA $24
F53A      A5 28          LDA $28
F53C      10 FA          BPL $F538
F53E      85 24          STA $24
F540      A5 28          LDA $28
F542      30 FA          BMI $F53E
F544      60             RTS
;
;Wait for start of next VBLANK
;
F545      85 24          STA $24                                       ;Wait for Sync
F547      A5 28          LDA #$28                                      ;
F549      30 FA          BMI $F545                                     ;Wait for end of VBLANK
F54B      85 24          STA $24                                       ;Wait for Sync
F54D      A5 28          LDA #$28                                      ;
F54F      10 FA          BPL $F54B                                     ;Wait for start of VBLANK
F551      60             RTS                                           ;

F552      20 5A DD       JSR $DD5A
F555      A0 00          LDY $00
F557      B9 5A F2       LDA $F25A,Y
F55A      99 0D 18       STA $180D,Y
F55D      C8             INY
F55E      C0 18          CPY $18
F560      D0 F5          BNE $F557
F562      A0 00          LDY $00
F564      B9 34 F2       LDA $F234,Y
F567      99 6C 22       STA $226C,Y
F56A      C8             INY
F56B      C0 29          CPY $29
F56D      D0 F5          BNE $F564
F56F      60             RTS

F570      A9 6F          LDA #$6F
F572      8D 80 24       STA $2480
F575      A9 D4          LDA #$D4
F577      8D 81 24       STA $2481
F57A      A9 80          LDA #$80
F57C      8D 82 24       STA $2482
F57F      A9 3A          LDA #$3A
F581      8D 83 24       STA $2483
F584      A9 00          LDA #$00
F586      8D 85 24       STA $2485
F589      60             RTS

F58A      A9 7B          LDA #$7B
F58C      8D 00 25       STA $2500
F58F      A9 B7          LDA #FRMCNT
F591      8D 01 25       STA $2501
```

```
F594     A9 80          LDA #$80
F596     8D 02 25       STA $2502
F599     A9 3C          LDA #$3C
F59B     8D 03 25       STA $2503
F59E     A9 84          LDA #$84
F5A0     18             CLC
F5A1     65 61          ADC $61
F5A3     8D 04 25       STA $2504
F5A6     A9 BF          LDA #$BF
F5A8     8D 05 25       STA $2505
F5AB     A9 80          LDA #$80
F5AD     8D 06 25       STA $2506
F5B0     A9 64          LDA #SKILL
F5B2     8D 07 25       STA $2507
F5B5     A9 00          LDA #$00
F5B7     8D 09 25       STA $2509
F5BA     20 91 D6       JSR $D691
F5BD     60             RTS

F5BE     A9 84          LDA #$84
F5C0     18             CLC
F5C1     65 6C          ADC NUMPLAYERS
F5C3     8D 04 26       STA $2604
F5C6     A9 DF          LDA #$DF
F5C8     8D 05 26       STA $2605
F5CB     A9 80          LDA #$80
F5CD     8D 06 26       STA $2606
F5D0     A9 3A          LDA #$3A
F5D2     8D 07 26       STA $2607
F5D5     A9 00          LDA #$00
F5D7     8D 09 26       STA $2609
F5DA     A9 7B          LDA #$7B
F5DC     8D 00 26       STA $2600
F5DF     A9 D7          LDA #$D7
F5E1     8D 01 26       STA $2601
F5E4     A9 80          LDA #$80
F5E6     8D 02 26       STA $2602
F5E9     A9 43          LDA #$43
F5EB     8D 03 26       STA $2603
F5EE     A6 64          LDX SKILL
F5F0     BD 72 F2       LDA $F272,X
F5F3     8D 80 26       STA $2680
F5F6     BD 77 F2       LDA $F277,X
F5F9     8D 81 26       STA $2681
F5FC     A9 80          LDA #$80
F5FE     8D 82 26       STA $2682
F601     BD 7C F2       LDA $F27C,X
F604     8D 83 26       STA $2683
F607     60             RTS

F608     A9 C6          LDA #$C6
F60A     8D 80 27       STA $2780
F60D     A9 F0          LDA #$F0
F60F     8D 81 27       STA $2781
F612     A9 80          LDA #$80
F614     8D 82 27       STA $2782
F617     A9 0A          LDA #$0A
F619     8D 83 27       STA $2783
F61C     A9 B9          LDA #$B9
F61E     8D 84 27       STA $2784
F621     A9 F3          LDA #$F3
F623     8D 85 27       STA $2785
F626     A9 80          LDA #$80
F628     8D 86 27       STA $2786
F62B     A9 5A          LDA #$5A
F62D     8D 87 27       STA $2787
F630     60             RTS

F631     A9 00          LDA #$00
F633     8D 0D 18       STA $180D
F636     A9 26          LDA #$26
F638     8D 17 18       STA $1817
F63B     A9 80          LDA #$80
```

```
F63D      8D 18 18        STA $1818
F640      60              RTS

F641      A2 0E           LDX $0E
F643      BD 81 F2        LDA $F281,X
F646      9D 00 1A        STA $1A00,X
F649      CA              DEX
F64A      10 F7           BPL $F643
F64C      A9 CF           LDA #$CF
F64E      8D 04 18        STA $1804
F651      60              RTS

F652      AD 00 39        LDA $3900
F655      C9 C6           CMP #$C6
F657      D0 50           BNE $F6A9
F659      AD 04 39        LDA $3904
F65C      C9 FE           CMP #$FE
F65E      D0 49           BNE $F6A9
F660      20 45 F5        JSR $F545
F663      A9 01           LDA #$01
F665      85 69           STA $69
F667      A9 7F           LDA #$7F                                    ;Turn off DMA
F669      85 3C           STA $3C                                            ;
F66B      20 AE F6        JSR $F6AE
F66E      20 41 F6        JSR $F641
F671      A5 64           LDA SKILL
F673      0A              ASL A
F674      0A              ASL A
F675      8D 02 1A        STA $1A02
F678      20 55 D2        JSR $D255
F67B      A2 03           LDX $03
F67D      A0 00           LDY $00
F67F      B5 40           LDA $40,X
F681      99 00 1C        STA $1C00,Y
F684      C8              INY
F685      CA              DEX
F686      10 F7           BPL $F67F
F688      A0 1A           LDY $1A
F68A      20 FD 3F        JSR $3FFD
F68D      A5 6C           LDA NUMPLAYERS
F68F      F0 18           BEQ $F6A9
F691      EE 02 1A        INC $1A02
F694      A2 03           LDX $03
F696      A0 00           LDY $00
F698      B5 44           LDA $44,X
F69A      99 00 1C        STA $1C00,Y
F69D      C8              INY
F69E      CA              DEX
F69F      10 F7           BPL $F698
F6A1      A0 1A           LDY $1A
F6A3      20 FD 3F        JSR $3FFD
F6A6      20 45 F5        JSR $F545
F6A9      A9 00           LDA #$00
F6AB      85 69           STA $69
F6AD      60              RTS

F6AE      20 45 F5        JSR $F545
F6B1      20 52 F5        JSR $F552
F6B4      20 31 F6        JSR $F631
F6B7      20 BE F5        JSR $F5BE
F6BA      60              RTS

F6BB      A5 68           LDA $68
F6BD      29 0F           AND #$0F
F6BF      85 5F           STA $5F
F6C1      A5 E5           LDA $E5
F6C3      0A              ASL A
F6C4      0A              ASL A
F6C5      29 F0           AND #$F0
F6C7      05 5F           ORA $5F
F6C9      85 68           STA $68
F6CB      20 91 D6        JSR $D691
F6CE      60              RTS
```

```
F6CF      A5 E5           LDA $E5
F6D1      C5 79           CMP $79
F6D3      D0 01           BNE $F6D6
F6D5      60              RTS

F6D6      85 79           STA $79
F6D8      A5 68           LDA $68
F6DA      29 0F           AND #$0F
F6DC      85 5F           STA $5F
F6DE      A5 E5           LDA $E5
F6E0      0A              ASL A
F6E1      0A              ASL A
F6E2      0A              ASL A
F6E3      29 F0           AND #$F0
F6E5      05 5F           ORA $5F
F6E7      85 68           STA $68
F6E9      85 21           STA $21
F6EB      49 F0           EOR #$F0
F6ED      85 22           STA $22
F6EF      38              SEC
F6F0      E5 68           SBC $68
F6F2      29 F0           AND #$F0
F6F4      05 5F           ORA $5F
F6F6      85 23           STA $23
F6F8      60              RTS

DISPINIT:
F6F9      20 9A DD        JSR $DD9A
F6FC      A9 00           LDA #$00
F6FE      85 7A           STA $7A
F700      85 7B           STA $7B
F702      A9 37           LDA #$37
F704      85 68           STA $68
F706      A5 E5           LDA $E5
F708      85 79           STA $79
F70A      A6 7B           LDX $7B
F70C      BD D0 F2        LDA $F2D0,X
F70F      85 BA           STA TADDRL
F711      BD D2 F2        LDA $F2D2,X
F714      85 BB           STA TADDRH
F716      A0 0F           LDY $0F
F718      B1 BA           LDA (TADDRL),Y
F71A      99 A0 00        STA $00A0,Y
F71D      88              DEY
F71E      10 F8           BPL $F718
F720      A6 7A           LDX $7A
F722      BD 90 F2        LDA $F290,X
F725      85 BA           STA TADDRL
F727      BD A5 F2        LDA $F2A5,X
F72A      85 BB           STA TADDRH
F72C      C8              INY
F72D      A5 A1           LDA TEMP1
F72F      91 BA           STA (TADDRL),Y
F731      C8              INY
F732      A9 1E           LDA #$1E
F734      91 BA           STA (TADDRL),Y
F736      A5 A0           LDA TEMP0
F738      C8              INY
F739      91 BA           STA (TADDRL),Y
F73B      C8              INY
F73C      BD FE F2        LDA $F2FE,X
F73F      91 BA           STA (TADDRL),Y
F741      18              CLC
F742      69 08           ADC #$08
F744      85 B1           STA TEMP17
F746      C8              INY
F747      BD 08 F3        LDA $F308,X
F74A      AA              TAX
F74B      F0 1C           BEQ $F769
F74D      A5 AD           LDA TEMP13
F74F      91 BA           STA (TADDRL),Y
F751      C8              INY
```

```
F752      A9 1E           LDA #$1E
F754      91 BA           STA (TADDRL),Y
F756      C8              INY
F757      A5 AC           LDA TEMP12
F759      91 BA           STA (TADDRL),Y
F75B      C8              INY
F75C      A5 B1           LDA TEMP17
F75E      91 BA           STA (TADDRL),Y
F760      18              CLC
F761      69 08           ADC #$08
F763      85 B1           STA TEMP17
F765      C8              INY
F766      CA              DEX
F767      D0 E4           BNE $F74D
F769      A6 7A           LDX $7A
F76B      A5 A3           LDA TEMP3
F76D      91 BA           STA (TADDRL),Y
F76F      C8              INY
F770      A9 1E           LDA #$1E
F772      91 BA           STA (TADDRL),Y
F774      A5 A2           LDA TEMP2
F776      C8              INY
F777      91 BA           STA (TADDRL),Y
F779      C8              INY
F77A      BD F4 F2        LDA $F2F4,X
F77D      91 BA           STA (TADDRL),Y
F77F      A0 00           LDY $00
F781      BD 9A F2        LDA $F29A,X
F784      85 BA           STA TADDRL
F786      BD AF F2        LDA $F2AF,X
F789      85 BB           STA TADDRH
F78B      A5 A5           LDA TEMP5
F78D      91 BA           STA (TADDRL),Y
F78F      C8              INY
F790      A9 1E           LDA #$1E
F792      91 BA           STA (TADDRL),Y
F794      A5 A4           LDA TEMP4
F796      C8              INY
F797      91 BA           STA (TADDRL),Y
F799      C8              INY
F79A      BD FE F2        LDA $F2FE,X
F79D      91 BA           STA (TADDRL),Y
F79F      18              CLC
F7A0      69 08           ADC #$08
F7A2      85 B1           STA TEMP17
F7A4      C8              INY
F7A5      BD 08 F3        LDA $F308,X
F7A8      AA              TAX
F7A9      F0 1C           BEQ $F7C7
F7AB      A5 AF           LDA TEMP15
F7AD      91 BA           STA (TADDRL),Y
F7AF      C8              INY
F7B0      A9 1E           LDA #$1E
F7B2      91 BA           STA (TADDRL),Y
F7B4      C8              INY
F7B5      A5 AE           LDA TEMP14
F7B7      91 BA           STA (TADDRL),Y
F7B9      C8              INY
F7BA      A5 B1           LDA TEMP17
F7BC      91 BA           STA (TADDRL),Y
F7BE      18              CLC
F7BF      69 08           ADC #$08
F7C1      85 B1           STA TEMP17
F7C3      C8              INY
F7C4      CA              DEX
F7C5      D0 E4           BNE $F7AB
F7C7      A6 7A           LDX $7A
F7C9      A5 A7           LDA TEMP7
F7CB      91 BA           STA (TADDRL),Y
F7CD      C8              INY
F7CE      A9 1E           LDA #$1E
F7D0      91 BA           STA (TADDRL),Y
F7D2      A5 A6           LDA TEMP6
```

```
F7D4      C8              INY
F7D5      91 BA           STA (TADDRL),Y
F7D7      C8              INY
F7D8      BD F4 F2        LDA $F2F4,X
F7DB      91 BA           STA (TADDRL),Y
F7DD      BD 90 F2        LDA $F290,X
F7E0      18              CLC
F7E1      69 56           ADC #$56
F7E3      85 BA           STA TADDRL
F7E5      BD A5 F2        LDA $F2A5,X
F7E8      69 00           ADC #$00
F7EA      85 BB           STA TADDRH
F7EC      A5 BA           LDA TADDRL
F7EE      18              CLC
F7EF      7D 13 F3        ADC $F313,X
F7F2      85 BA           STA TADDRL
F7F4      A5 BB           LDA TADDRH
F7F6      69 00           ADC #$00
F7F8      85 BB           STA TADDRH
F7FA      A0 00           LDY $00
F7FC      BD FE F2        LDA $F2FE,X
F7FF      85 B1           STA TEMP17
F801      BD F4 F2        LDA $F2F4,X
F804      85 B2           STA TEMP18
F806      BD 08 F3        LDA $F308,X
F809      AA              TAX
F80A      F0 39           BEQ $F845
F80C      A0 00           LDY $00
F80E      A5 A9           LDA TEMP9
F810      91 BA           STA (TADDRL),Y
F812      A9 1E           LDA #$1E
F814      C8              INY
F815      91 BA           STA (TADDRL),Y
F817      C8              INY
F818      A5 A8           LDA TEMP8
F81A      91 BA           STA (TADDRL),Y
F81C      C8              INY
F81D      A5 B1           LDA TEMP17
F81F      91 BA           STA (TADDRL),Y
F821      C8              INY
F822      A5 AB           LDA TEMP11
F824      91 BA           STA (TADDRL),Y
F826      A9 1E           LDA #$1E
F828      C8              INY
F829      91 BA           STA (TADDRL),Y
F82B      C8              INY
F82C      A5 AA           LDA TEMP10
F82E      91 BA           STA (TADDRL),Y
F830      C8              INY
F831      A5 B2           LDA TEMP18
F833      91 BA           STA (TADDRL),Y
F835      18              CLC
F836      A5 BA           LDA TADDRL
F838      69 56           ADC #$56
F83A      85 BA           STA TADDRL
F83C      A5 BB           LDA TADDRH
F83E      69 00           ADC #$00
F840      85 BB           STA TADDRH
F842      CA              DEX
F843      D0 C7           BNE $F80C
F845      A4 7A           LDY $7A
F847      C8              INY
F848      C0 0A           CPY #$0A
F84A      D0 09           BNE $F855
F84C      A5 7B           LDA $7B
F84E      D0 0D           BNE $F85D
F850      A8              TAY
F851      A9 01           LDA #$01
F853      85 7B           STA $7B
F855      84 7A           STY $7A
F857      20 5E F8        JSR $F85E
F85A      4C 0A F7        JMP $F70A
F85D      60              RTS
```

```
F85E     A0 03          LDY $03
F860     20 38 F5       JSR $F538
F863     20 CF F6       JSR $F6CF
F866     20 34 DB       JSR $DB34
F869     88             DEY
F86A     10 F4          BPL $F860
F86C     60             RTS

;Look for collisions between a sprite and player
;Expects
; y = index of object to perform collision detection against player
;
; Returns
; If the player collides with a family member
; $6B will hold the index of the family member and y will be 0.
;
; if there was no collision with anything y will be set to 0 anyway,
; otherwise y will be index of enemy collided with.
;
F86D     A0 01          LDY $01                                   ;First Sprite
F86F     B9 91 1F       LDA SPRITE_STATE_$1F91,Y                  ;Get sprite enabled flag
F872     C9 01          CMP #$01                                  ;Is sprite enabled?
F874     D0 2A          BNE $F8A0                                 ;Branch if it isn't
F876     18             CLC                                       ;
F877     AD 26 1B       LDA SPRITE_Y                              ;Get player Y position
F87A     69 20          ADC #$20                                  ;
F87C     D9 26 1B       CMP SPRITE_Y,Y                            ;Compare against sprite y
position
F87F     90 1F          BCC $F8A0                                 ;Branch if (player_y + 20) <
sprite_y
F881     AD 26 1B       LDA SPRITE_Y                              ;Get player Y position
F884     E9 20          SBC #$20                                  ;
F886     D9 26 1B       CMP SPRITE_Y,Y                            ;Compare against sprite y
position
F889     B0 15          BCS $F8A0                                 ;Branch if (player_y - 20) >=
sprite_y
F88B     18             CLC                                       ;
F88C     AD CF 1A       LDA SPRITE_X                              ;Get player X position
F88F     69 13          ADC #$13                                  ;
F891     D9 CF 1A       CMP SPRITE_X,Y                            ;Compare against sprite x
position
F894     90 0A          BCC $F8A0                                 ;Branch if (player_x + 13) <
sprite_x
F896     AD CF 1A       LDA SPRITE_X                              ;Get player X position
F899     E9 13          SBC #$13                                  ;
F89B     D9 CF 1A       CMP SPRITE_X,Y                            ;Compare against sprite x
position
F89E     90 08          BCC $F8A8                                 ;Branch if (player_x - 13) <
sprite_x
F8A0     C8             INY                                       ;Next sprite
F8A1     C0 54          CPY #$54                                  ;End of list?
F8A3     90 CA          BCC $F86F                                 ;Branch if not
F8A5     A0 00          LDY $00                                   ;Reset Y to 0 meaning "no
collision with
                                                                  ;   player" (IMPORTANT -
calling routine
                                                                  ;   checks this)
F8A7     60             RTS

; If we get here, a collision has occurred with the player.  But what type of collision?
F8A8     B9 8C 1E       LDA SPRITE_TYPE_$1E8C,Y                   ;Get sprite type
F8AB     29 1F          AND #$1F                                  ;Mask type
F8AD     C9 02          CMP #$02                                  ;Is it a Mommy??
F8AF     90 09          BCC $F8BA                                 ;No, it must be a grunt
F8B1     C9 05          CMP #$05                                  ;A Hulk?
F8B3     B0 05          BCS $F8BA                                 ;Yes, whatever it is, its an
enemy and
                                                                  ; will kill the player, so exit
sub
F8B5     84 6B          STY $6B                                   ;If we get here it must be a
family
                                                                  ; member.  Save index of
```

```
family member.
F8B7      4C A0 F8       JMP $F8A0                               ;Back to collision scan
F8BA      60             RTS                                     ;End up here if we hit
something deadly
;
; This routine is called from one place and one place only ($D94F).
;
F8BB      86 AE          STX TEMP14                              ;Save X in a temp variable
F8BD      A5 C9          LDA CRELEFT                             ;How many enemies we got on
screen?
F8BF      F0 11          BEQ $F8D2
F8C1      20 6D F8       JSR CHK_PLYER_COLLISION_$F86D           ;Look for collisions
F8C4      C0 00          CPY #$00                                ;Did we hit something deadly?
F8C6      D0 23          BNE $F8EB                               ;Branch if we did
F8C8      A5 C9          LDA CRELEFT                             ;How many enemies on screen?
F8CA      C9 01          CMP #$01                                ;Just the one?
F8CC      D0 0D          BNE $F8DB                               ;No
F8CE      A5 5B          LDA $5B                                 ;How many family members are on
screen?
F8D0      F0 09          BEQ $F8DB                               ;Zero
F8D2      A6 AE          LDX TEMP14                              ;Restore X from temp variable
F8D4      A9 00          LDA #$00
F8D6      85 AE          STA TEMP14
F8D8      A9 0F          LDA #$0F                                ;Set return value of 15 -
exciting!!!
F8DA      60             RTS

F8DB      A4 6A          LDY CURRENTOBJ_6A
F8DD      C8             INY
F8DE      C0 54          CPY #$54
F8E0      90 02          BCC $F8E4
F8E2      A0 01          LDY $01
F8E4      84 6A          STY CURRENTOBJ_6A
F8E6      B9 91 1F       LDA SPRITE_STATE_$1F91,Y
F8E9      F0 F2          BEQ $F8DD
F8EB      A2 00          LDX $00
F8ED      86 B7          STX FRMCNT
F8EF      20 6E BD       JSR PICK_DIRECTION_$BD6E
F8F2      C9 0F          CMP #$0F
F8F4      F0 DC          BEQ $F8D2
F8F6      A8             TAY
F8F7      A6 AE          LDX TEMP14
F8F9      A9 01          LDA #$01
F8FB      85 AE          STA TEMP14
F8FD      B9 15 EC       LDA $EC15,Y
F900      4C 72 D9       JMP $D972

F903      CE 2B 1C       DEC SPRITE_DELTA_Y_$1C2B
F906      30 06          BMI $F90E
F908      AD D4 1B       LDA SPRITE_DELTA_X_$1BD4
F90B      4C BA D7       JMP $D7BA
F90E      A9 00          LDA #$00
F910      8D 2B 1C       STA SPRITE_DELTA_Y_$1C2B
F913      20 6D F8       JSR CHK_PLYER_COLLISION_$F86D
F916      C0 00          CPY #$00
F918      F0 1A          BEQ $F934
F91A      A2 00          LDX $00
F91C      86 B7          STX FRMCNT
F91E      20 6E BD       JSR PICK_DIRECTION_$BD6E
F921      C9 0F          CMP #$0F
F923      F0 41          BEQ $F966
F925      AA             TAX
F926      A9 03          LDA #$03
F928      8D 2B 1C       STA SPRITE_DELTA_Y_$1C2B
F92B      BC 80 F9       LDY $F980,X
F92E      B9 15 EC       LDA $EC15,Y
F931      4C BA D7       JMP $D7BA
F934      A5 5B          LDA $5B
F936      F0 2E          BEQ $F966
F938      A4 6B          LDY $6B                                 ;Get sprite number of Human we
hit
F93A      B9 91 1F       LDA SPRITE_STATE_$1F91,Y                ;Look up enable flag
F93D      C9 01          CMP #$01                                ;
```

```
F93F      D0 2A          BNE $F96B                                  ;Branch if it isn't enabled
F941      B9 8C 1E       LDA SPRITE_TYPE_$1E8C,Y                    ;Get Sprite type
F944      29 1F          AND #$1F                                   ;Mask off type
F946      C9 02          CMP #$02                                   ;
F948      90 21          BCC $F96B                                  ;Branch if less the 2
F94A      C9 05          CMP #$05                                   ;
F94C      B0 1D          BCS $F96B                                  ;Branch if >= 5
F94E      A2 00          LDX $00                                    ;
F950      86 B7          STX FRMCNT
F952      20 6E BD       JSR PICK_DIRECTION_$BD6E
F955      C9 0F          CMP #$0F
F957      F0 0D          BEQ $F966
F959      A8             TAY
F95A      B9 15 EC       LDA $EC15,Y
F95D      4C BA D7       JMP $D7BA
F960      AD D4 1B       LDA SPRITE_DELTA_X_$1BD4
F963      4C BA D7       JMP $D7BA
F966      A9 0F          LDA #$0F
F968      4C BA D7       JMP $D7BA
F96B      C8             INY
F96C      84 6B          STY $6B
F96E      38             SEC
F96F      AD 06 19       LDA OBJS_PER_LEVEL_$1906
F972      69 18          ADC #$18
F974      C5 6B          CMP $6B
F976      B0 BC          BCS $F934
F978      AC 06 19       LDY OBJS_PER_LEVEL_$1906
F97B      84 6B          STY $6B
F97D      4C 66 F9       JMP $F966

;UNKNOWN USAGE ($F980-$F987)
F980                     .BYTE $02,$03,$01,$00,$05,$06,$07,$04,

F988                                                         $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
F990                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
F9A0                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
F9B0                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
F9C0                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
F9D0                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
F9E0                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
F9F0                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

FA00                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FA10                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FA20                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FA30                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FA40                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FA50                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FA60                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FA70                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FA80                     .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FA90                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FAA0                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FAB0                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FAC0                                                                    .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
```

```
FAD0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FAE0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FAF0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

FB00                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FB10                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FB20                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FB30                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FB40                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FB50                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FB60                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FB70                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FB80                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FB90                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FBA0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FBB0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FBC0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FBD0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FBE0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FBF0                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

FC00                        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FC10                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FC20                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FC30                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FC40                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FC50                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FC60                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FC70                                                                 .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

;UNKNOWN USAGE ($FC80-$FD13)
FC80                        .BYTE $C0,$C0,$C0,$C0,$C0,$C0,$C0,$A0,$A0,$A0,$C0,$A0,$C0,$A0,$A0,$C0
FC90                        .BYTE $A0,$00,$10,$11,$1A,$22,$2A,$2E,$2F,$30,$31,$32,$53,$52,$58,$59
FCA0                        .BYTE $3A,$4A,$10,$11,$12,$00,$00,$08,$08,$08,$00,$0C,$0C,$0C,$00,$04
FCB0                        .BYTE $00,$00,$13,$1D,$21,$2B,$2F,$2B,$2B,$2F,$2F,$25,$33,$37,$3B,$3F
FCC0                        .BYTE $3B,$3B,$3F,$3F,$43,$47,$4B,$4F,$4B,$4B,$4F,$4F,$53,$57,$5B,$5F
FCD0                        .BYTE $63,$6A,$77,$7F,$8B,$95,$9F,$AA,$9F,$9F,$AA,$AA,$B5,$B5,$B5,$B5
FCE0                        .BYTE $B5,$B6,$B7,$B8,$B5,$B7,$B6,$B8,$B5,$B9,$B9,$B5,$BA,$BD,$C0,$C3
FCF0                        .BYTE $C6,$C9,$CC,$CF,$D2,$D6,$D7,$D8,$D9,$DA,$E0,$FC,$00,$02,$00,$04
FD00                        .BYTE $06,$08,$06,$0A,$0C,$0E,$0C,$10,$12,$14,$12,$16,$18,$1C,$20,$22
FD10                        .BYTE $24,$22,$26,$C9

******************************************************************
*   RDIRS.S                     DIRECTION DATA
*
*

************
*
```

```
*         TABLES FOR ANIMATION
*
************




*******************************************************************
*                                                                 *
*         CRETODST STANDS FOR CREATURE TO DIRECTION START         *
*         IT TELLS WHICH ENTRY IN THE DIRECTION TABLE IS THE      *
*         FIRST POINTER TO THE STEPS                              *
*                                                                 *
*******************************************************************
*
CRETODST:;.BYTE 0        ;MC
FD14                     .BYTE $CB
                         ;.BYTE (GD-DIRTOSTE)                         ;GD
FD15                     .BYTE $CD
                         ;.BYTE (MOD-DIRTOSTE)                        ;MOD
FD16                     .BYTE $CF
                         ;.BYTE (DD-DIRTOSTE)                         ;DD
FD17                     .BYTE $D1
                         ;.BYTE (MID-DIRTOSTE)                        ;MI
FD18                     .BYTE $D3
                         ;.BYTE (HD-DIRTOSTE)                         ;H
FD19                     .BYTE $28
                         ;.BYTE (SD-DIRTOSTE)                         ;S
FD1A                     .BYTE $2A
                         ;.BYTE (QD-DIRTOSTE)                         ;Q
FD1B                     .BYTE $28
                         ;.BYTE (ED-DIRTOSTE)                         ;E
FD1C                     .BYTE $2C
                         ;.BYTE (TD-DIRTOSTE)                         ;T
FD1D                     .BYTE $2E
                         ;.BYTE (BD-DIRTOSTE)                         ;B
FD1E                     .BYTE $30
                         ;.BYTE 0
FD1F                     .BYTE $2E
                         ;.BYTE 0
FD20                     .BYTE $32
                         ;.BYTE 0
FD21                     .BYTE $70
                         ;.BYTE 0
FD22                     .BYTE $73
                         ;.BYTE (MCSD-DIRTOSTE)                       ;MCS
FD23                     .BYTE $76

********************************************************
*         DIRTOSTE STANDS FOR DIRECTION TO STEP IT IS
*         ACCESSED BY ADDING THE CONTENTS OF CRETODST
*         TO THE DIRECTION AND USING THAT TO INDEX IN.
*         TAKING ITS CONTENTS AND ADDING THE STEP GIVES
*         A POINTER INTO ALL THE STAMP TABLES
*
DIRTOSTE:;.BYTE 0        ;MC DIRECTIONS
FD24                     .BYTE $79
                         ;.BYTE 0                                     ;1
FD25                     .BYTE $7C
                         ;.BYTE 0                                     ;2
FD26                     .BYTE $7F
                         ;.BYTE 0                                     ;3
FD27                     .BYTE $34
                         ;.BYTE 0                                     ;4
FD28                     .BYTE $36
                         ;.BYTE (MCD2-STAMPL)                         ;5
FD29                     .BYTE $34
                         ;.BYTE (MCD2-STAMPL)                         ;6
FD2A                     .BYTE $38
                         ;.BYTE (MCD2-STAMPL)                         ;7
FD2B                     .BYTE $3A
                         ;.BYTE 0                                     ;8
FD2C                     .BYTE $3C
```

```
                         ;.BYTE (MCD3-STAMPL)                      ;9
FD2D                     .BYTE $3A
                         ;.BYTE (MCD3-STAMPL)                      ;A
FD2E                     .BYTE $3E
                         ;.BYTE (MCD3-STAMPL)                      ;B
FD2F                     .BYTE $40
                         ;.BYTE 0                                  ;C
FD30                     .BYTE $42
                         ;.BYTE (MCD1-STAMPL)                      ;D
FD31                     .BYTE $40
                         ;.BYTE 0                                  ;E
FD32                     .BYTE $44
                         ;.BYTE 0                                  ;F
FD33                     .BYTE $46

GD:                      ;.BYTE (GD0-STAMPL)                       ;G
FD34                     .BYTE $48

MOD:                     ;.BYTE (MOD0-STAMPL)                      ;MOMMY DIRECTIONS
FD35                     .BYTE $46
                         ;.BYTE (MOD1-STAMPL)
FD36                     .BYTE $4A
                         ;.BYTE (MOD2-STAMPL)
FD37                     .BYTE $4C
                         ;.BYTE (MOD3-STAMPL)
FD38                     .BYTE $4E
                         ;.BYTE (MOD2-STAMPL)                      ;D4
FD39                     .BYTE $4C
                         ;.BYTE (MOD2-STAMPL)                      ;D5
FD3A                     .BYTE $50
                         ;.BYTE (MOD3-STAMPL)                      ;D6
FD3B                     .BYTE $52
                         ;.BYTE (MOD3-STAMPL)                      ;D7
FD3C                     .BYTE $54
                         ;.BYTE (FDIE-STAMPL)
FD3D                     .BYTE $52

DD:                      ;.BYTE (DD0-STAMPL)                       ;DADDY DIRS
FD3E                     .BYTE $56
                         ;.BYTE (DD1-STAMPL)
FD3F                     .BYTE $58
                         ;.BYTE (DD2-STAMPL)
FD40                     .BYTE $5A
                         ;.BYTE (DD3-STAMPL)
FD41                     .BYTE $58
                         ;.BYTE (DD2-STAMPL)                       ;D4
FD42                     .BYTE $5C
                         ;.BYTE (DD2-STAMPL)                       ;D5
FD43                     .BYTE $5E
                         ;.BYTE (DD3-STAMPL)                       ;D6
FD44                     .BYTE $60
                         ;.BYTE (DD3-STAMPL)                       ;D7
FD45                     .BYTE $5E

MID:                     ;.BYTE (MID0-STAMPL)                      ;MIKEY DIRS
FD46                     .BYTE $62
                         ;.BYTE (MID1-STAMPL)
FD47                     .BYTE $64
                         ;.BYTE (MID2-STAMPL)
FD48                     .BYTE $66
                         ;.BYTE (MID3-STAMPL)
FD49                     .BYTE $64
                         ;.BYTE (MID2-STAMPL)                      ;D4
FD4A                     .BYTE $68
                         ;.BYTE (MID2-STAMPL)                      ;D5
FD4B                     .BYTE $6A
                         ;.BYTE (MID3-STAMPL)                      ;D6
FD4C                     .BYTE $6C
                         ;.BYTE (MID3-STAMPL)                      ;D7
FD4D                     .BYTE $6A

HD:                      ;.BYTE (HD0-STAMPL)                       ;HULK DIRS
FD4E                     .BYTE $6E
```

```
                        ;.BYTE (HD1-STAMPL)
FD4F                    .BYTE $82
                        ;.BYTE (HD2-STAMPL)
FD50                    .BYTE $84
                        ;.BYTE (HD3-STAMPL)
FD51                    .BYTE $82

SD:                     ;.BYTE (SD0-STAMPL)                         ;S
FD52                    .BYTE $86

QD:                     ;.BYTE (QD0-STAMPL)                         ;Q
FD53                    .BYTE $88

ED:                     ;.BYTE (ED0-STAMPL)                         ;E
FD54                    .BYTE $8A

TD:                     ;.BYTE (TD0-STAMPL)                         ;T
FD55                    .BYTE $88

BD:                     ;.BYTE (BD0-STAMPL)                         ;B
FD56                    .BYTE $8C
                        ;.BYTE (BD1-STAMPL)
FD57                    .BYTE $8E
                        ;.BYTE (BD2-STAMPL)
FD58                    .BYTE $90
                        ;.BYTE (BD3-STAMPL)
FD59                    .BYTE $8E
                        ;.BYTE (BD2-STAMPL)                         ;D4
FD5A                    .BYTE $92
                        ;.BYTE (BD2-STAMPL)                         ;D5
FD5B                    .BYTE $94
                        ;.BYTE (BD3-STAMPL)                         ;D6
FD5C                    .BYTE $96
                        ;.BYTE (BD3-STAMPL)                         ;D7
FD5D                    .BYTE $94

MCSD:                   ;.BYTE (MCSD0-STAMPL)                       ;MCS
FD5E                    .BYTE $98
                        ;.BYTE (MCSD0-STAMPL)                       ;1
FD5F                    .BYTE $9A
                        ;.BYTE (MCSD0-STAMPL)                       ;2
FD60                    .BYTE $9D
                        ;.BYTE (MCSD0-STAMPL)                       ;3
FD61                    .BYTE $A0
                        ;.BYTE (MCSD0-STAMPL)                       ;4
FD62                    .BYTE $A3
                        ;.BYTE (MCSD5-STAMPL)                       ;5
FD63                    .BYTE $A6
                        ;.BYTE (MCSD6-STAMPL)                       ;6
FD64                    .BYTE $A9
                        ;.BYTE (MCSD7-STAMPL)                       ;7
FD65                    .BYTE $73
                        ;.BYTE (MCSD0-STAMPL)                       ;8
FD66                    .BYTE $57
                        ;.BYTE (MCSD6-STAMPL)                       ;9
FD67                    .BYTE $57
                        ;.BYTE (MCSD5-STAMPL)                       ;A
FD68                    .BYTE $5A
                        ;.BYTE (MCSD7-STAMPL)                       ;B
FD69                    .BYTE $5A
                        ;.BYTE (MCSD0-STAMPL)                       ;C
FD6A                    .BYTE $5D
                        ;.BYTE (MCSDD-STAMPL)                       ;D
FD6B                    .BYTE $5D
                        ;.BYTE (MCSDD-STAMPL)                       ;E
FD6C                    .BYTE $60
                        ;.BYTE (MCSD0-STAMPL)                       ;F
FD6D                    .BYTE $60

*****************************************************
*         DATA DESCRIBING THE STAMPS
*
*
```

```
STAMPL:  ;.BYTE L(MCD0S0)
FD6E                    .BYTE $63
                        ;.BYTE L(MCD0S1)
FD6F                    .BYTE $63
                        ;.BYTE L(MCD0S0)
FD70                    .BYTE $66
                        ;.BYTE L(MCD0S2)
FD71                    .BYTE $66

MCD1:                   ;.BYTE L(MCD1S0)
FD72                    .BYTE $69
                        ;.BYTE L(MCD1S1)
FD73                    .BYTE $47
                        ;.BYTE L(MCD1S0)
FD74                    .BYTE $49
                        ;.BYTE L(MCD1S2)
FD75                    .BYTE $4B

MCD2:                   ;.BYTE L(MCD2S0)
FD76                    .BYTE $4D
                        ;.BYTE L(MCD2S1)
FD77                    .BYTE $4F
                        ;.BYTE L(MCD2S0)
FD78                    .BYTE $51
                        ;.BYTE L(MCD2S2)
FD79                    .BYTE $53

MCD3:                   ;.BYTE L(MCD3S0)
FD7A                    .BYTE $55
                        ;.BYTE L(MCD3S1)
FD7B                    .BYTE $6C
                        ;.BYTE L(MCD3S0)
FD7C                    .BYTE $6F
                        ;.BYTE L(MCD3S2)
FD7D                    .BYTE $72

GD0:                    ;.BYTE L(GD0S0)
FD7E                    .BYTE $75
                        ;.BYTE L(GD0S1)
FD7F                    .BYTE $78
                        ;.BYTE L(GD0S0)
FD80                    .BYTE $7B
                        ;.BYTE L(GD0S2)
FD81                    .BYTE $7E
                        ;.BYTE L(GD0S3)               ;THESE ARE EXPLOSIONS
FD82                    .BYTE $81
                        ;.BYTE L(GD0S4)
FD83                    .BYTE $84
                        ;.BYTE L(GD0S5)
FD84                    .BYTE $87
                        ;.BYTE L(GD0S6)
FD85                    .BYTE $8A
                        ;.BYTE L(GD0S7)
FD86                    .BYTE $8D
                        ;.BYTE L(GD0S8)
FD87                    .BYTE $AC

MOD0:                   ;.BYTE L(MOD0S0)
FD88                    .BYTE $AE
                        ;.BYTE L(MOD0S1)
FD89                    .BYTE $AC
                        ;.BYTE L(MOD0S0)
FD8A                    .BYTE $B0
                        ;.BYTE L(MOD0S2)
FD8B                    .BYTE $D5

MOD1:                   ;.BYTE L(MOD1S0)
FD8C                    .BYTE $D7
                        ;.BYTE L(MOD1S1)
FD8D                    .BYTE $D9
                        ;.BYTE L(MOD1S0)
FD8E                    .BYTE $DB
                        ;.BYTE L(MOD1S2)
```

```
FD8F                    .BYTE $DD

FDIE:                   ;.BYTE L(SKULL)
FD90                    .BYTE $DF
                        ;.BYTE L(SCORE1K)
FD91                    .BYTE $B2
                        ;.BYTE L(SCORE2K)
FD92                    .BYTE $B4
                        ;.BYTE L(SCORE3K)
FD93                    .BYTE $B2
                        ;.BYTE L(SCORE4K)
FD94                    .BYTE $B6
                        ;.BYTE L(SCORE5K)
FD95                    .BYTE $D5

MOD2:                   ;.BYTE L(MOD2S0)
FD96                    .BYTE $D7
                        ;.BYTE L(MOD2S1)
FD97                    .BYTE $D9
                        ;.BYTE L(MOD2S0)
FD98                    .BYTE $DB
                        ;.BYTE L(MOD2S2)
FD99                    .BYTE $DD

MOD3:                   ;.BYTE L(MOD3S0)
FD9A                    .BYTE $DF
                        ;.BYTE L(MOD3S1)
FD9B                    .BYTE $B8
                        ;.BYTE L(MOD3S0)
FD9C                    .BYTE $BA
                        ;.BYTE L(MOD3S2)
FD9D                    .BYTE $B8

DD0:                    ;.BYTE L(DD0S0)
FD9E                    .BYTE $BC
                        ;.BYTE L(DD0S1)
FD9F                    .BYTE $D5
                        ;.BYTE L(DD0S0)
FDA0                    .BYTE $D7
                        ;.BYTE L(DD0S2)
FDA1                    .BYTE $D9

DD1:                    ;.BYTE L(DD1S0)
FDA2                    .BYTE $DB
                        ;.BYTE L(DD1S1)
FDA3                    .BYTE $DD
                        ;.BYTE L(DD1S0)
FDA4                    .BYTE $DF
                        ;.BYTE L(DD1S2)
FDA5                    .BYTE $E3

DD2:                    ;.BYTE L(DD2S0)
FDA6                    .BYTE $BE
                        ;.BYTE L(DD2S1)
FDA7                    .BYTE $C0
                        ;.BYTE L(DD2S0)
FDA8                    .BYTE $BE
                        ;.BYTE L(DD2S2)
FDA9                    .BYTE $C2

DD3:                    ;.BYTE L(DD3S0)
FDAA                    .BYTE $D5
                        ;.BYTE L(DD3S1)
FDAB                    .BYTE $D7
                        ;.BYTE L(DD3S0)
FDAC                    .BYTE $D9
                        ;.BYTE L(DD3S2)
FDAD                    .BYTE $DB

MID0:                   ;.BYTE L(MID0S0)
FDAE                    .BYTE $DD
                        ;.BYTE L(MID0S1)
FDAF                    .BYTE $DF
```

```
                         ;.BYTE L(MID0S0)
FDB0                     .BYTE $E1
                         ;.BYTE L(MID0S2)
FDB1                     .BYTE $C4

MID1:                    ;.BYTE L(MID1S0)
FDB2                     .BYTE $C5
                         ;.BYTE L(MID1S1)
FDB3                     .BYTE $C6
                         ;.BYTE L(MID1S0)
FDB4                     .BYTE $C7
                         ;.BYTE L(MID1S2)
FDB5                     .BYTE $C8

MID2:                    ;.BYTE L(MID2S0)
FDB6                     .BYTE $1B
                         ;.BYTE L(MID2S1)
FDB7                     .BYTE $2C
                         ;.BYTE L(MID2S0)
FDB8                     .BYTE $2E
                         ;.BYTE L(MID2S2)
FDB9                     .BYTE $1D

MID3:                    ;.BYTE L(MID3S0)
FDBA                     .BYTE $30
                         ;.BYTE L(MID3S1)
FDBB                     .BYTE $32
                         ;.BYTE L(MID3S0)
FDBC                     .BYTE $1F
                         ;.BYTE L(MID3S2)
FDBD                     .BYTE $2C

HD0:                     ;.BYTE L(HD0S0)
FDBE                     .BYTE $2E
                         ;.BYTE L(HD0S1)
FDBF                     .BYTE $21
                         ;.BYTE L(HD0S0)
FDC0                     .BYTE $34
                         ;.BYTE L(HD0S2)
FDC1                     .BYTE $2E

HD1:                     ;.BYTE L(HD1S0)
FDC2                     .BYTE $23
                         ;.BYTE L(HD1S1)
FDC3                     .BYTE $34
                         ;.BYTE L(HD1S0)
FDC4                     .BYTE $36
                         ;.BYTE L(HD1S2)
FDC5                     .BYTE $24

HD2:                     ;.BYTE L(HD2S0)
FDC6                     .BYTE $37
                         ;.BYTE L(HD2S1)
FDC7                     .BYTE $39
                         ;.BYTE L(HD2S0)
FDC8                     .BYTE $26
                         ;.BYTE L(HD2S2)
FDC9                     .BYTE $3B

HD3:                     ;.BYTE L(HD3S0)
FDCA                     .BYTE $3D
                         ;.BYTE L(HD3S1)
FDCB                     .BYTE $28
                         ;.BYTE L(HD3S0)
FDCC                     .BYTE $3F
                         ;.BYTE L(HD3S2)
FDCD                     .BYTE $43

SD0:                     ;.BYTE L(SD0S0)
FDCE                     .BYTE $E5
                         ;.BYTE L(SD0S1)
FDCF                     .BYTE $E7
                         ;.BYTE L(SD0S2)
```

```
FDD0                    .BYTE $E9
                        ;.BYTE L(SD0S3)
FDD1                    .BYTE $EB
                        ;.BYTE L(SD0S4)
FDD2                    .BYTE $90
                        ;.BYTE L(SD0S5)
FDD3                    .BYTE $92
                        ;.BYTE L(SD0S6)
FDD4                    .BYTE $94
                        ;.BYTE L(SD0S7)
FDD5                    .BYTE $96
                        ;.BYTE L(QD0S0)
FDD6                    .BYTE $98

QD0:                    ;.BYTE L(QD0S1)
FDD7                    .BYTE $9A
                        ;.BYTE L(QD0S2)
FDD8                    .BYTE $9C
                        ;.BYTE L(QD0S3)
FDD9                    .BYTE $9E
                        ;.BYTE L(QD0S4)
FDDA                    .BYTE $A0
                        ;.BYTE L(QD0S5)
FDDB                    .BYTE $A2
                        ;.BYTE L(QD0S6)
FDDC                    .BYTE $A4
                        ;.BYTE L(QD0S7)
FDDD                    .BYTE $A6

ED0:                    ;.BYTE L(ED0S0)
FDDE                    .BYTE $A8
                        ;.BYTE L(ED0S1)
FDDF                    .BYTE $AA

TD0:                    ;.BYTE L(TD0S0)
FDE0                    .BYTE $AC
                        ;.BYTE L(TD0S1)
FDE1                    .BYTE $AE
                        ;.BYTE L(TD0S2)
FDE2                    .BYTE $B0
                        ;.BYTE L(TD0S3)
FDE3                    .BYTE $B2

BD0:                    ;.BYTE L(BD0S0)
FDE4                    .BYTE $B4
                        ;.BYTE L(BD0S1)
FDE5                    .BYTE $B6
                        ;.BYTE L(BD0S0)
FDE6                    .BYTE $B8
                        ;.BYTE L(BD0S2)
FDE7                    .BYTE $BA

BD1:                    ;.BYTE L(BD1S0)
FDE8                    .BYTE $BC
                        ;.BYTE L(BD1S1)
FDE9                    .BYTE $BE
                        ;.BYTE L(BD1S0)
FDEA                    .BYTE $C0
                        ;.BYTE L(BD1S2)
FDEB                    .BYTE $C2

BD2:                    ;.BYTE L(BD2S0)
FDEC                    .BYTE $C4
                        ;.BYTE L(BD2S1)
FDED                    .BYTE $C6
                        ;.BYTE L(BD2S0)
FDEE                    .BYTE $C8
                        ;.BYTE L(BD2S2)
FDEF                    .BYTE $CA

BD3:                    ;.BYTE L(BD3S0)
FDF0                    .BYTE $CC
                        ;.BYTE L(BD3S1)
```

```
FDF1                    .BYTE $CE
                        ;.BYTE L(BD3S0)
FDF2                    .BYTE $D0
                        ;.BYTE L(BD3S2)
FDF3                    .BYTE $D2

MCSD0:                  ;.BYTE L(MCSD0S0)
FDF4                    .BYTE $D4
MCSD5:                  ;.BYTE L(MCSD5S0)
FDF5                    .BYTE $D6
MCSD6:                  ;.BYTE L(MCSD6S0)
FDF6                    .BYTE $D8
MCSD7:                  ;.BYTE L(MCSD7S0)
FDF7                    .BYTE $DA
MCSDD:                  ;.BYTE L(MCSDDS0)
FDF8                    .BYTE $DC

STAMPHGH:
FDF9                    .BYTE $0B                                    ;MC D0 S0
FDFA                    .BYTE $0B                                    ;MC D1
FDFB                    .BYTE $0B                                    ;MC D2
FDFC                    .BYTE $0B                                    ;MC D3
FDFD                    .BYTE $0B                                    ;MC D4
FDFE                    .BYTE $0B                                    ;MC D5
FDFF                    .BYTE $0B                                    ;MC D6
FE00                    .BYTE $0B                                    ;MC D7
FE01                    .BYTE $0B                                    ;MC D8
FE02                    .BYTE $0B                                    ;MC D9
FE03                    .BYTE $0B                                    ;MC DA
FE04                    .BYTE $0B                                    ;MC DB
FE05                    .BYTE $0B                                    ;MC DC
FE06                    .BYTE $0B                                    ;MC DD
FE07                    .BYTE $0B                                    ;MC DE
FE08                    .BYTE $0B                                    ;MC DF
FE09                    .BYTE $0B                                    ;G D0
FE0A                    .BYTE $0B                                    ;MO D0
FE0B                    .BYTE $0B                                    ;MO D1
FE0C                    .BYTE $0B                                    ;MO D2
FE0D                    .BYTE $0B                                    ;MO D3
FE0E                    .BYTE $0B                                    ;MO D2
FE0F                    .BYTE $0B                                    ;MO D2
FE10                    .BYTE $0B                                    ;MO D3
FE11                    .BYTE $0B                                    ;MO D3

;(A DYING FAMILY MEMBER EITHER POINTS OR SKULL)
FE12                    .BYTE $0B                                    ;FDIE
FE13                    .BYTE $0B                                    ;D D0
FE14                    .BYTE $0B                                    ;D D1
FE15                    .BYTE $0B                                    ;D D2
FE16                    .BYTE $0B                                    ;D D3
FE17                    .BYTE $0B                                    ;D D2
FE18                    .BYTE $0B                                    ;D D2
FE19                    .BYTE $0B                                    ;D D3
FE1A                    .BYTE $0B                                    ;D D3
FE1B                    .BYTE $0A                                    ;MI D0
FE1C                    .BYTE $0A                                    ;MI D1
FE1D                    .BYTE $0A                                    ;MI D2
FE1E                    .BYTE $0A                                    ;MI D3
FE1F                    .BYTE $0A                                    ;MI D2
FE20                    .BYTE $0A                                    ;MI D2
FE21                    .BYTE $0A                                    ;MI D3
FE22                    .BYTE $0A                                    ;MI D3
FE23                    .BYTE $0D                                    ;H D0
FE24                    .BYTE $0D                                    ;H D1
FE25                    .BYTE $0D                                    ;H D2
FE26                    .BYTE $0D                                    ;H D3
FE27                    .BYTE $0E                                    ;S D0 S0
FE28                    .BYTE $09                                    ;Q D0 S0
FE29                    .BYTE $0A                                    ;E D0 S0
FE2A                    .BYTE $10                                    ;T D0 S0
FE2B                    .BYTE $0D                                    ;B D0
FE2C                    .BYTE $0D                                    ;B D1
FE2D                    .BYTE $0D                                    ;B D2
```

```
FE2E                     .BYTE $0D                                  ;B D3
FE2F                     .BYTE $0D                                  ;B D2
FE30                     .BYTE $0D                                  ;B D2
FE31                     .BYTE $0D                                  ;B D3
FE32                     .BYTE $0D                                  ;B D3
FE33                     .BYTE $00                                  ;MCSD0
FE34                     .BYTE $00                                  ;MCSD1
FE35                     .BYTE $00                                  ;MCSD2
FE36                     .BYTE $00                                  ;MCSD3
FE37                     .BYTE $00                                  ;MCSD4
FE38                     .BYTE $07                                  ;MCSD5
FE39                     .BYTE $07                                  ;MCSD6
FE3A                     .BYTE $01                                  ;MCSD7
FE3B                     .BYTE $00                                  ;MCSD8
FE3C                     .BYTE $07                                  ;MCSD6
FE3D                     .BYTE $07                                  ;MCSD5
FE3E                     .BYTE $01                                  ;MCSD7
FE3F                     .BYTE $00                                  ;MCSDC
FE40                     .BYTE $07                                  ;MCSDD
FE41                     .BYTE $07                                  ;MCSDD
FE42                     .BYTE $00                                  ;MCSDF

;UNKNOWN USAGE
FE43                     .BYTE $00,$00,$00,$00,$00,$00,$00,$00
FE4B                     .BYTE $07,$0F,$0F,$0B,$0B,$0B,$0B,$05

PALNWID:
FE53                     .BYTE $1C                                  ;MC D0 S0
FE54                     .BYTE $1C                                  ;MC D1
FE55                     .BYTE $1E                                  ;MC D2
FE56                     .BYTE $1E                                  ;MC D3
FE57                     .BYTE $1E                                  ;MC D4
FE58                     .BYTE $1E                                  ;MC D5
FE59                     .BYTE $1E                                  ;MC D6
FE5A                     .BYTE $1E                                  ;MC D7
FE5B                     .BYTE $1E                                  ;MC D8
FE5C                     .BYTE $1E                                  ;MC D9
FE5D                     .BYTE $1E                                  ;MC DA
FE5E                     .BYTE $1E                                  ;MC DB
FE5F                     .BYTE $1E                                  ;MC DC
FE60                     .BYTE $1E                                  ;MC DD
FE61                     .BYTE $1E                                  ;MC DE
FE62                     .BYTE $1E                                  ;MC DF
FE63                     .BYTE $3E                                  ;G D0
FE64                     .BYTE $5E                                  ;MO D0
FE65                     .BYTE $5E                                  ;MO D1
FE66                     .BYTE $5F                                  ;MO D2
FE67                     .BYTE $5F                                  ;MO D3
FE68                     .BYTE $5E                                  ;MO D2
FE69                     .BYTE $5E                                  ;MO D2
FE6A                     .BYTE $5F                                  ;MO D3
FE6B                     .BYTE $5F                                  ;MO D3
FE6C                     .BYTE $BD                                  ;FDIE
FE6D                     .BYTE $7E                                  ;D D0
FE6E                     .BYTE $7E                                  ;D D1
FE6F                     .BYTE $7E                                  ;D D2
FE70                     .BYTE $7E                                  ;D D3
FE71                     .BYTE $7E                                  ;D D2
FE72                     .BYTE $7E                                  ;D D2
FE73                     .BYTE $7E                                  ;D D3
FE74                     .BYTE $7E                                  ;D D3
FE75                     .BYTE $3E                                  ;MI D0
FE76                     .BYTE $3E                                  ;MI D1
FE77                     .BYTE $3E                                  ;MI D2
FE78                     .BYTE $3F                                  ;MI D3
FE79                     .BYTE $3E                                  ;MI D2
FE7A                     .BYTE $3E                                  ;MI D2
FE7B                     .BYTE $3F                                  ;MI D3
FE7C                     .BYTE $3F                                  ;MI D3
FE7D                     .BYTE $9E                                  ;H D0
FE7E                     .BYTE $9E                                  ;H D1
FE7F                     .BYTE $9E                                  ;H D2
FE80                     .BYTE $9E                                  ;H D3
```

```
FE81                    .BYTE $3D                                   ;S D0 S0
FE82                    .BYTE $7D                                   ;Q D0 S0
FE83                    .BYTE $7E                                   ;E D0 S0
FE84                    .BYTE $FD                                   ;T D0 S0
FE85                    .BYTE $DE                                   ;B D0
FE86                    .BYTE $DE                                   ;B D1
FE87                    .BYTE $DE                                   ;B D2
FE88                    .BYTE $DE                                   ;B D3
FE89                    .BYTE $DE                                   ;B D2
FE8A                    .BYTE $DE                                   ;B D2
FE8B                    .BYTE $DE                                   ;B D3
FE8C                    .BYTE $DE                                   ;B D3
FE8D                    .BYTE $BF                                   ;MCSD0
FE8E                    .BYTE $BF                                   ;MCSD1
FE8F                    .BYTE $BF                                   ;MCSD2
FE90                    .BYTE $BF                                   ;MCSD3
FE91                    .BYTE $BF                                   ;MCSD4
FE92                    .BYTE $BF                                   ;MCSD5
FE93                    .BYTE $BF                                   ;MCSD6
FE94                    .BYTE $BF                                   ;MCSD7
FE95                    .BYTE $BF                                   ;MCSD8
FE96                    .BYTE $BF                                   ;MCSD6
FE97                    .BYTE $BF                                   ;MCSD5
FE98                    .BYTE $BF                                   ;MCSD7
FE99                    .BYTE $BF                                   ;MCSDC
FE9A                    .BYTE $BF                                   ;MCSDD
FE9B                    .BYTE $BF                                   ;MCSDD
FE9C                    .BYTE $BF                                   ;MCSDF

;UNKNOWN USAGE
FE9D                    .BYTE $BE,$BE,$BE,$BE,$BF,$BE,$BE,$BC
FEA5                    .BYTE $BE,$BE,$BE,$BE,$BE,$BE,$BE,$BF

STAMPPWD:
FEAD                    .BYTE $04                                   ;MC D0 S0
FEAE                    .BYTE $04                                   ;MC D1
FEAF                    .BYTE $04                                   ;MC D2
FEB0                    .BYTE $04                                   ;MC D3
FEB1                    .BYTE $04                                   ;MC D4
FEB2                    .BYTE $04                                   ;MC D5
FEB3                    .BYTE $04                                   ;MC D6
FEB4                    .BYTE $04                                   ;MC D7
FEB5                    .BYTE $04                                   ;MC D8
FEB6                    .BYTE $04                                   ;MC D9
FEB7                    .BYTE $04                                   ;MC DA
FEB8                    .BYTE $04                                   ;MC DB
FEB9                    .BYTE $04                                   ;MC DC
FEBA                    .BYTE $04                                   ;MC DD
FEBB                    .BYTE $04                                   ;MC DE
FEBC                    .BYTE $04                                   ;MC DF
FEBD                    .BYTE $06                                   ;G D0
FEBE                    .BYTE $04                                   ;MO D0
FEBF                    .BYTE $04                                   ;MO D1
FEC0                    .BYTE $04                                   ;MO D2
FEC1                    .BYTE $04                                   ;MO D3
FEC2                    .BYTE $04                                   ;MO D2
FEC3                    .BYTE $04                                   ;MO D2
FEC4                    .BYTE $04                                   ;MO D3
FEC5                    .BYTE $04                                   ;MO D3
FEC6                    .BYTE $01                                   ;FDIE
FEC7                    .BYTE $04                                   ;D D0
FEC8                    .BYTE $04                                   ;D D1
FEC9                    .BYTE $04                                   ;D D2
FECA                    .BYTE $04                                   ;D D3
FECB                    .BYTE $04                                   ;D D2
FECC                    .BYTE $04                                   ;D D2
FECD                    .BYTE $04                                   ;D D3
FECE                    .BYTE $04                                   ;D D3
FECF                    .BYTE $04                                   ;MI D0
FED0                    .BYTE $04                                   ;MI D1
FED1                    .BYTE $04                                   ;MI D2
FED2                    .BYTE $04                                   ;MI D3
FED3                    .BYTE $04                                   ;MI D2
```

```
FED4                    .BYTE $04                                 ;MI D2
FED5                    .BYTE $04                                 ;MI D3
FED6                    .BYTE $04                                 ;MI D3
FED7                    .BYTE $06                                 ;H D0
FED8                    .BYTE $06                                 ;H D1
FED9                    .BYTE $06                                 ;H D2
FEDA                    .BYTE $06                                 ;H D3
FEDB                    .BYTE $0B                                 ;S D0 S0
FEDC                    .BYTE $06                                 ;Q D0 S0
FEDD                    .BYTE $07                                 ;E D0 S0
FEDE                    .BYTE $09                                 ;T D0 S0
FEDF                    .BYTE $06                                 ;B D0
FEE0                    .BYTE $06                                 ;B D1
FEE1                    .BYTE $06                                 ;B D2
FEE2                    .BYTE $06                                 ;B D3
FEE3                    .BYTE $06                                 ;B D2
FEE4                    .BYTE $06                                 ;B D2
FEE5                    .BYTE $06                                 ;B D3
FEE6                    .BYTE $06                                 ;B D3
FEE7                    .BYTE $00                                 ;MCSD0
FEE8                    .BYTE $00                                 ;MCSD0
FEE9                    .BYTE $00                                 ;MCSD0
FEEA                    .BYTE $00                                 ;MCSD0
FEEB                    .BYTE $00                                 ;MCSD0
FEEC                    .BYTE $03                                 ;MCSD5
FEED                    .BYTE $03                                 ;MCSD6
FEEE                    .BYTE $03                                 ;MCSD7
FEEF                    .BYTE $00                                 ;MCSD0
FEF0                    .BYTE $03                                 ;MCSD6
FEF1                    .BYTE $03                                 ;MCSD5
FEF2                    .BYTE $03                                 ;MCSD7
FEF3                    .BYTE $00                                 ;MCSD0
FEF4                    .BYTE $01                                 ;MCSDD
FEF5                    .BYTE $01                                 ;MCSDD
FEF6                    .BYTE $00                                 ;MCSD0

;UNKNOWN USAGE ($FEF7-$FF06)
FEF7                    .BYTE $00,$00,$00,$00,$00,$00,$00,$00,$05
FF00                    .BYTE $05,$05,$08,$08,$05,$07,$04,

FF07                                                   $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FF10                                                              .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FF20                                                              .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FF30                                                              .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FF40                                                              .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FF50                                                              .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FF60                                                              .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FF70                                                              .BYTE
$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF

;960-BIT (120 BYTE) ENCRYPTION ($FF80-FFF7)
FF80                    .BYTE $07,$F0,$8F,$F5,$9B,$31,$5E,$FF,$FC,$C3,$15,$B5,$A8,$69,$E3,$D5
FF90                    .BYTE $9F,$4C,$DB,$56,$1B,$8B,$B0,$E0,$BB,$C9,$73,$CA,$5D,$DF,$72,$E3
FFA0                    .BYTE $24,$6B,$94,$C9,$E0,$42,$D1,$7B,$68,$AE,$81,$54,$2F,$52,$20,$28
FFB0                    .BYTE $8C,$3B,$50,$B3,$AC,$F9,$26,$04,$A8,$4B,$36,$F9,$21,$6C,$69,$CD
FFC0                    .BYTE $88,$9A,$77,$28,$5B,$A6,$23,$E6,$15,$B9,$00,$FB,$9D,$AF,$7A,$1C
FFD0                    .BYTE $07,$E3,$70,$F9,$2B,$FE,$4A,$96,$77,$06,$79,$41,$18,$29,$32,$53
FFE0                    .BYTE $CF,$03,$34,$48,$5F,$FF,$66,$FF,$AF,$31,$81,$BB,$88,$1D,$14,$B7
FFF0                    .BYTE $94,$34,$76,$CD,$39,$12,$76,$98,$FF,$87,$6B,$D5,$1D,$F3,$27,$D6
;VECTOR (CART START) IS $F31D.


#######################################################################################
#######################################################################################
;UNCLAIMED ORPHANS
;
;THESE SCRAPS OF CODE ORIGINATED IN THE ORIGINAL SOURCE LIST AND COULD NOT BE EASILY
```

```
; PAIRED WITH THE RELEASED BINARY CODE DECODING.  THESE ARE RETAINED IN CASE SOMEONE
; (1) HAS THE TIME TO LOOK AT ALL THIS, AND (2) HAS SUPREME COMMAND OF HOW THE ATARI
; 7800 IS PROGRAMMED.  THIS WAY, NO ONE HAS TO START COMPLETELY FROM SCRATCH TO TRY
; TO DETERMINE IF ANY MORE ORIGINAL SOURCE LISTING CAN BE DECODED AND SALVAGED.  ANY
; COMMENTED CODE ALLOWS FURTHER UNDERSTANDING OF HOW THE GAME OPERATES.  THE PARTS
; OF SOURCE CODE THAT WERE SUCCESSFULLY DECODED ARE ABOVE, THE FAILED SET IS BELOW.

********************************
*
*         STAMP DATA
*
********************************
*

* THE STAMPS START AT STAMPBAS BUT ARE FILLED WITH ZEROES UP TO STAMPBAS+$F00
*         THE DIFFERENT LINES OF EACH STAMP ARE 100H APART

STAMPBAS  EQU    $4300      ;BASE ADDRESS OF STAMPS
*** WE MUST MAKE SURE THIS IS NOT OVERWRITTEN BY THE END OF
*********   THE PRECEDING TABLES.  IF SO, INCREASE STAMPBAS

          ORG     STAMPBAS

*         WE NEED 15 PAGES OF ZEROES HERE, AND 15 AFTER THE END OF
*                 EACH STAMP.  TO DO THIS, FIRST USE A DO LOOP
*                 TO FILL MEMORY FROM STAMPBAS FOR 46 PAGES
*                 46 = 15 (ZEROES) + 16 (MAX STAMP) + 15 (ZEROES)
*         AFTER WE CREATE ALL THOSE ZEROES, DEFINE STAMPS ON TOP
*                 AS NEEDED

          PRINT   OFF

          DO      16         ;16 PAGES
          DO     $10        ;ALLOCATE 256 BYTES PER PAGE
          DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
          ENDDO
          ENDDO

          ORG     STAMPBAS+$1E00
          DO      16         ;16 PAGES
          DO     $10        ;ALLOCATE 256 BYTES PER PAGE
          DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
          ENDDO
          ENDDO

          PRINT ON


********** END OF RDIRS.S *****************************

######################################################################################

######################################################################################
PART OF RSOUNDS.S (SOMEHOW)

;DURING DISASSEMBLY, THE "DOTUNE" LOCATION $E395 WAS DETERMINED BY THE JSR (20 95 E3)
AT ADDRESS $D9A4.  THE CODE HAS COMPLETELY STRAYED FROM OUR SOURCE LIST ON FILE:

DOTUNE:
E395                    STA TUNNUM                                  ;SAVE IT
         8A             TXA                                         ;STACK REGISTERS
         48             PHA
         98             TYA
         48             PHA

                        LDX TUNNUM                                  ;FIND PRIORITY OF NEW TUNE
                        LDA TPRIOR,X
                        STA TUNTEMP0
                        JSR GETCHANL                                ;GET A CHANNEL
                        CPX #$00                                    ;SEE IF WE GOT ONE
         30 XX          BMI DTOUT
                        LDA TUNNUM
         20 XX XX       JSR BEGINTUN                                ;START THE TUNE
```

```
DTOUT:
         68             PLA                                         ;UNSTACK REGISTERS
         A8             TAY
         68             PLA
         AA             TAX
         60             RTS


*  THIS ROUTINE TRIES TO GET A CHANNEL TO PUT A TUNE INTO.  THIS IS DONE BY
*  GIVING AN OPEN CHANNEL IF AVAILABLE OR BUMPING SOMEONE IF NONE ARE OPEN.
*  IF THE PRIORITY IS TO LOW TO GET ANYTHING, FF IS RETURNED.
*  INPUT: PRIORITY OF REQUESTOR IN TUNTEMP0
*  OUTPUT: CHANNEL IN X, $FF IF NO CHANNEL AVAILABLE
*  USES:  X, Y

GETCHANL:
                        LDX #$01                                    ;FIRST - SEE IF OPEN ACTIVE
CHANNEL

GCLOOP0:
                        LDA TUNON,X                                 ;SEE IF CHANNEL OPEN
                        BNE GCNEXT0
         60             RTS                                         ;GOT IT - EASY

GCNEXT0:
                        DEX
                        BPL GCLOOP0

                        LDX #$00                                    ;NOW, TRY TO BUMP ACTIVE
CHANNEL
                        LDA TUNPRIOR                                ;GET INDEX OF LOWER PRIORITY
ACTIVE CNL
                        CMP TUNPRIOR+1
                        BMI GCJMP0
                        INX

GCJMP0:
                        LDA TUNPRIOR,X                              ;SEE IF OUR PRIORITY HIGHER
                        CMP TUNTEMP0
                        BPL GCNONE                                  ;NO CHANNELS AVAILABLE

                        JMP ENDTUNE

                  ;HERE I DELETED STUFF TO HANDLE BACKUP CHANNELS

GCNONE:
                        LDX #$FF                                    ;NO CHANNELS AVAILABLE
         60             RTS


*  ROUTINE TO KILL A PARTICULAR TUNE - IF IT IS RUNNING
*  INPUT: TUNE NUMBER IN A
*  X AND Y ARE PRESERVED

KILLTUNE:
                        STA TUNNUM                                  ;SAVE IT
                        TXA                                         ;STACK REGISTERS
                        PHA
                        TYA
                        PHA
                        LDX #$01                                    ;CHECK ALL CHANNELS

KTLOOP:
                        LDA TUNON,X                                 ;SEE IF CHANNEL ON
                        BEQ KTNEXT
                        LDA TUNINDEX,X                              ;SEE IF HAS TUNE TO BE KILLED
                        CMP TUNNUM
                        BNE KTNEXT
         20 XX XX       JSR ENDTUNE                                 ;ERASE IT

KTNEXT:
         CA             DEX
         10 XX          BPL KTLOOP
         68             PLA                                         ;UNSTACK REGISTERS
         A8             TAY
         68             PLA
         AA             TAX
         60             RTS


*  THIS ROUTINE ERASES ALL TUNES
*  X AND Y ARE PRESERVED
CLEARTUN:
                        TXA                            ;STACK REGISTERS
                        PHA
                        TYA
                        PHA
                        LDX #$01

CTLOOP:
                        JSR ENDTUNE                                 ;ERASE CURRENT TUNE
                        DEX
                        BPL CTLOOP
                        PLA                                         ;UNSTACK REGISTERS
                        TAY
                        PLA
                        TAX
                        RTS


*  THIS ROUTINE IS CALLED EVERY VBLANK TO TAKE CARE OF TUNES
*  REGISTERS ARE NOT SAVED
TUNER:
                        LDX #$01                                    ;TWO TUNES CHANNELS, START
WITH SECOND
TUNLOOP:
                        LDA TUNON,X
                        BNE TUNBODY
                        STA AUDV0,X                                 ;CHANNEL OFF - MAKE SURE
VOLUME OFF
                        JMP TUNNEXT

TUNBODY:
                        LDA TUNBASE,X                               ;GET ADDRESS OF TUNE
                        STA SOUNDZP
                        LDA TUNBASE1,X
                        STA SOUNDZP+1

                        DEC FREQTIME,X                              ;DO FREQUENCY
                        BNE TUNCTL
                        JSR TNXTFREQ                                ;TIME FOR NEXT FREQUENCY

TUNCTL:
                        LDA CTLTIME,X                               ;DO CONTROL
                        BEQ TUNVOL                                  ;IS CTL CONSTANT? (STARTS AT
0)
                        DEC CTLTIME,X
                        BNE TUNVOL
                        JSR TNXTCTL                                 ;TIME FOR NEXT CTL

TUNVOL:
                        LDA VOLTIME,X                               ;DO VOLUME
                        BEQ TUNNEXT                                 ;IS VOLUME CONSTANT? (STARTS
AT 0)
                        DEC VOLTIME,X
                        BNE TUNNEXT
                        JSR TNXTVOL                                 ;TIME FOR NEXT VOLUME

TUNNEXT:
         CA             DEX                                         ;DONE WITH THAT TUNE, IS THERE
ANOTHER?
         10 XX          BPL TUNLOOP
         60             RTS                                         ;ALL DONE

*  ROUTINES TO GET NEXT FREQUENCY, CTL, OR VOLUME
*  THIS ROUTINE GETS NEXT FREQUENCY
TNXTFREQ:
                        LDY FREQOFF,X                              ;GET INDEX INTO TABLE
                        LDA (SOUNDZP),Y                            ;GET FREQUENCY
         30 XX          BMI TNFENDT                                ;IS THIS THE END OF THE TUNE?
                        STA AUDF0,X
                        INY
                        LDA (SOUNDZP),Y                            ;GET DURATION
                        STA FREQTIME,X
                        INY
                        TYA
                        STA FREQOFF,X
         60             RTS

TNFENDT:
                        CMP #$FF                                   ;SEE IF TUNE OVER
         F0 XX          BEQ TNFEOVER
         C9 FE          CMP #$FE                                   ;SEE IF TUNE REPEATS
         F0 XX          BEQ TNFEREPT
         C8             INY                                        ;ANOTHER TUNE COMING
                        LDA (SOUNDZP),Y                            ;FIND OUT WHICH TUNE
         20 XX XX       JSR BEGINTUN                               ;START TUNE
         4C XX XX       JMP TNFEOUT

TNFEREPT:
                        LDA TUNINDEX,X                             ;TUNE REPEATS - RESTART IT
         20 XX XX       JSR BEGINTUN                               ;START TUNE
         4C XX XX       JMP TNFEOUT

TNFEOVER:
         20 XX XX       JSR ENDTUNE                                ;TUNE FINISHED

TNFEOUT:
         68             PLA                                        ;END OF TUNE
         68             PLA                                        ;GET RID OF WHERE WE ARE
RTS'ING TO
         4C XX XX       JMP TUNLOOP                                ;UPDATE THIS CHANNEL

*  THIS ROUTINE GETS NEXT CONTROL BYTE
TNXTCTL:
                        LDY CTLOFF,X                               ;GET INDEX INTO TABLE
                        LDA (SOUNDZP),Y                            ;GET FREQUENCY
                        STA AUDC0,X
         C8             INY
                        LDA (SOUNDZP),Y                            ;GET DURATION
                        STA CTLTIME,X
         C8             INY
         98             TYA
                        STA CTLOFF,X
         60             RTS

*  THIS ROUTINE GETS NEXT VOLUME BYTE
TNXTVOL:
                        LDY VOLOFF,X                               ;GET INDEX INTO TABLE
                        LDA (SOUNDZP),Y                            ;GET FREQUENCY
                        STA AUDV0,X
                        INY
                        LDA (SOUNDZP),Y                            ;GET DURATION
                        STA VOLTIME,X
         C8             INY
                        TYA
                        STA VOLOFF,X
         60             RTS

*  THIS ROUTINE CLEARS OUT A TUNE CHANNEL
*  INPUT: X IS CHANNEL

ENDTUNE:
                        LDA #$00
                        STA TUNON,X                                ;INDICATE CHANNEL CLEAR
                        STA TUNINDEX,X                             ;CLEAR TUNE INDEX
```

```
                  ;HERE WAS BACKUP CHANNEL STUFF
ETOUT:
         60             RTS

*  THIS ROUTINE STARTS A TUNE IN A CHANNEL
*  INPUT: X IS CHANNEL, A IS TUNE
*  USES:  Y
BEGINTUN:
                        A8 TAY                                      ;PUT TUNE IN Y
                        STA TUNINDEX,X                              ;SET THE TUNE INDEX
                        LDA TBASE,Y                                 ;SET THE BASE ADDRESS FOR TUNE
                        STA TUNBASE,X
                        LDA TBASE1,Y
                        STA TUNBASE1,X
                        LDA #$00                                    ;FREQUENCY IS AT START OF TUNE
                        STA FREQOFF,X
                        LDA TCTLOFF,Y                               ;SET CONTROL OFFSET
                        STA CTLOFF,X
                        LDA TVOLOFF,Y                               ;SET VOLUME OFFSET
                        STA VOLOFF,X
                        LDA TPRIOR,Y                                ;SET PRIORITY
                        STA TUNPRIOR,X
                        LDA #$01                                    ;SET FREQ, CTL, AND VOL TO BE
SET
                        STA FREQTIME,X                              ;  NEXT VBLANK (TICK DOWN TO 0
EACH)
                        STA CTLTIME,X
                        STA VOLTIME,X
                        STA TUNON,X                                 ;AND TURN THE TUNE ON!
                        RTS

#####################################################################################

*******************************************************************************
*  THIS ROUTINE MOVES A TUNE FROM ONE CHANNEL TO ANOTHER
*  INPUT: Y IS FROM CHANNEL, X IS TO CHANNEL
*         ; THIS ROUTINE IS NO LONGER NECESSARY  ;IS THIS TRUE IN 2013 ???
*
*  DATA FOR TUNES
*
*  TUNE TABLES, BASE ADDRESSES FOR TUNES AND THE OFFSETS WITHIN THE TUNES WHERE
*  THE CTL AND VOL INFORMATION START
*
TBASE:                  ;.BYTE L(TMCS),L(TFPICK),L(TCREDIE),L(TSKULL0),L(TSKULL1),L(TEXTRA)
                        .BYTE
                        ;.BYTE L(TRACKA),L(TRACKB),L(TMCDIE)
                        .BYTE

TBASE1:  ;.BYTE H(TMCS),H(TFPICK),H(TCREDIE),H(TSKULL0),H(TSKULL1),H(TEXTRA)
                        .BYTE
                        ;.BYTE H(TRACKA),H(TRACKB),H(TMCDIE)
                        .BYTE

TCTLOFF: ;.BYTE 021,023,015,017,017,009,078,111,009
                        .BYTE $15,$17,$0F,$11,$11,$09,$4D,$6F,$09

TVOLOFF: ;.BYTE 023,039,019,019,019,011,206,159,011
                        .BYTE $17,$27,$13,$13,$13,$11,$CE,$9F,$11

TPRIOR:  ;.BYTE 005,020,006,015,015,025,018,018,022
                        .BYTE $05,$14,$06,$0F,$0F,$19,$12,$12,$16
```

ADDRESS $E67A WAS AFTER THIS BLOCK

```
####################################################################################



#######################################################################################
```

ADDRESS $E6XX WAS BEFORE THIS BLOCK

```
***********
*         TEMPORARY EQUIVALENCES FOR ROBOTRON
***********
```

```
*SSKULL    EQU     SPACDTH
*SFPICK    EQU     SANIM10
*SMCS      EQU     SMUNCH
*SCREDIE   EQU     SFRTBNC
*SMCDIE    EQU     SENERGA


*  TUNE # -
;S          EQU     #
;T          DB      $00,$00,$00,$00,$00,$00,$00,$00,$00,$00
;           DB      $00,$00,$00,$00,$00,$00,$00,$00,$00,$00
;           DB      $00,$00,$00,$00,$00,$00,$00,$00,$00,$00

*********** END OF RSOUNDS.S *************

          EJE


#######################################################################################
```

ADDRESS $EC5C WAS BEFORE THIS BLOCK; THIS SHOULD BE IN $ED00 SOMEWHERE

```
*****************************************************************************
*
*         GSPTBL  -- STARTING GRUNT SPEEDS FOR VARIOUS WAVES
*                 DATA HERE IS FOR WAVES 1 TO 7, HIGHER WAVES USE #GSPMAX
*
;SHOULD BE  GSPTBL    DB      $A,$9,$8,$7,$6,$5,$4
GSPTBL:
                        .BYTE $7F,$9,$8,$7,$6,$5,$4
GSPMAX    EQU     $4                      ;MAXIMUM START-OF-RACK SPEED

*****************************************************************************
*
*         HSPTBL  -- STARTING HULK SPEEDS FOR VARIOUS WAVES
*                 DATA HERE IS FOR WAVES 1 TO 13, HIGHER WAVES USE #HSPMAX
*
HSPTBL:
                        .BYTE $7,$F,$E,$D,$0,$C,$B,$A,$9,$0,$8,$7,$6 ;ZEROES := NO HULKS
HSPMAX    EQU     $6                      ;MAXIMUM START-OF-RACK SPEED

*****************************************************************************
*
*         SQBTTBL -- BASE TIMES UNTIL FIRST BIRTH FOR VARIOUS WAVES
*                 DATA HERE IS FOR WAVES 1 TO 17, HIGHER WAVES USE #SQBTMAX
*
SQBTTBL:
                        .BYTE 0,70,60,50,0,45,60,40,35,35
                        .BYTE 35,50,32,32,32,32,39
SQBTMAX   EQU     30

*****************************************************************************
*
*         QSPTBL  -- STARTING QUARK SPEEDS FOR VARIOUS WAVES
*                 DATA HERE IS FOR WAVES 1 TO 13, HIGHER WAVES USE #QSPMAX
*
QSPTBL:
                        .BYTE 0,0,0,0,0,0,6,0,0,0,0,5,0,0,0,0,4
QSPMAX    EQU     3                       ;MAXIMUM START-OF-RACK SPEED

*****************************************************************************
*
*         TSPTBL  -- STARTING TANK SPEEDS FOR VARIOUS WAVES
*                 DATA HERE IS FOR WAVES 1 TO 17, HIGHER WAVES USE #TSPMAX
*
TSPTBL:
                        .BYTE $0,$0,$0,$0,$0,$0,$A,$0,$0,$0,$0,$9,$0,$0,$0,$0,$7
TSPMAX    EQU     $5                      ;MAXIMUM START-OF-RACK SPEED

*****************************************************************************
*
*         BSTPBL  -- STARTING BRAIN SPEEDS FOR VARIOUS WAVES
*                 DATA HERE IS FOR WAVES 1 TO 15, HIGHER WAVES USE #BSPMAX
```

```
*
BSPTBL:
                        .BYTE $C,$0,$0,$0,$A,$0,$0,$0,$0,$9,$0,$0,$0,$0,$7
BSPMAX    EQU     $6                     ;MAXIMUM START-OF-RACK SPEED

*****************************************************************************
*
*         BSTTBL  -- BASE BRAIN SHOT TIMER VALUES FOR VARIOUS WAVES
*                 DATA HERE IS FOR WAVES 1 TO 15, HIGHER WAVES USE #BSTMAX
*
BSTTBL:
                        .BYTE $0,$0,$0,$0,$C,$0,$0,$0,$0,$A,$0,$0,$0,$0,$8
BSTMAX    EQU     $7                     ;MAXIMUM START-OF-RACK VALUE
*****************************************************************************
ADDRESS $EF1A WAS AFTER THIS BLOCK

###########################################################################################


######################################################################
```
ADDRESS $B278 WAS BEFORE THIS BLOCK
```
WSHCONT           ;SET UP VARIABLES GLOBAL TO HULKS
*         SET HSPEED - NUMBER OF FRAMES BETWEEN HULK MOVES
                        LDA WAVENUM                ;CURRENT WAVE NUMBER
                        CMP #13                    ;ONLY HAVE 13 WAVES IN TABLE
                        BCC LOOKHSP                ;LOOK UP HSPEED FROM TABLE
*         WE ARE ABOVE WAVE 13, SET HSPEED TO #HSPMAX
                        LDA #HSPMAX
                        STA HSPEED
                        JMP WSH1




LOOKHSP:
                        TAY                            ;PUT WAVE NUMBER IN Y
                        LDA HSPTBL-1,Y   ;LOAD STARTING HSPEED - USE -1 BECAUSE NO WAVE 0
                        STA HSPEED
WSH1                                    ;DONE WITH HULK SETUP
######################################################################



######################################################################
******************************************
*
*         RDISP.S
*
******************************************
*
*  ROBOTRON DISPLAY DRIVERS              18-JUL-83     CARLOS
*                                        20-JUL-83
*                17-AUGUST-83            9:00
*
******************************************

*
************************************************************************
*                                                                      *
*     DISPINIT  --  INITIALIZE DISPLAY LIST AND ZONE OBJECT TABLES     *
*                                                                      *
************************************************************************
*
DISPINIT:
F6F9
                        LDA #H(DL)                                 ;SET DLIST ADDRESS
         85 BB          STA TADDRH
                        LDA #L(DL)
                        STA TADDRL

DISPINT0:
         A9 00          LDA #$00
         A8             TAY
```

```
DISPINT1:
                        STA (TADDRL),Y
         C8             INY
         C0 7F          CPY #$7F                                    ;TEST FOR END OF A ZONE LIST
         90 XX          BCC DISPINT1
         A9 1F          LDA #$1F
                        STA (TADDRL),Y                              ;SET LAST BYTE OF FREE LIST
         A5 BA          LDA TADDRL
         18             CLC
         69 80          ADC #$80                                    ;ADVANCE
                        STA TADDRL
         A5 BB          LDA TADDRH
         69 00          ADC #$00
                        STA TADDRH
         C9 20          CMP #$20                                    ;TEST IF END OF DISPLAY LISTS
         90 XX          BCC DISPINT0                                ;NOT DONE
*
                        LDA #H(ZONOBJL)                             ;SET ADDRESS OF ZONE OBJECT
LISTS
                        STA TADDRH
                        LDA #L(ZONOBJL)
                        STA TADDRL

DISPINT2:
         A9 00          LDA #$00
         A8             TAY

DISPINT3:
                        STA (TADDRL),Y
         C8             INY
                        CPY #$A8                                    ;TEST TO ADVANCE BASE ADDRESS
                        BCC DISPINT3
         A5 BA          LDA TADDRL
                        CLC
                        ADC #$A8                                    ;ADVANCE BASE ADDRESS
                        STA TADDRL
         A5 BB          LDA TADDRH
                        ADC #$00
                        STA TADDRH
                        CMP #H(DL)                                  ;TEST IF DONE
                        BNE DISPINT2
*
                        LDX #$0B                                    ;INIT OBJECT COUNT TABLES
                        LDA #$00

DISPIN4:
                        STA ZONOBJC,X
                        DEX
                        BPL DISPIN4

                        RTS
*
*************************************************************
*
*  DISPLOAD  --  LOAD DISPLAY LIST WITH INITIAL OBJECT DATA
*
*************************************************************
*
DISPLOAD

                        LDX #$4F
DISPL1:
                        LDA STTBL,X                                 ;SKIP OBJECTS WITH ZERO STATUS
         F0 XX          BEQ DISPL20
                        LDA YTBL,X                                  ;GET ZONE FROM Y POSITION
         85 B2          STA TEMP18
         4A             LSR A
         4A             LSR A
         4A             LSR A
         4A             LSR A
         85 AE          STA TEMP14                                  ;ZONE
         BD CF 1A       LDA XTBL,X
         85 B1          STA TEMP17                               ;HPOS

         20 EA D0       JSR GETSTAMP_$D0EA                       ;GET STAMP ADDRESS
         20 54 DE       JSR ZONELOAD_$DE54                       ;LOAD ZONE DISPLAY ENTRY

         A5 BA          LDA TADDRL                               ;SAVE OBJECTS DISPLAY LIST
ADDR
                        STA DLPLTBL,X
         A5 BB          LDA TADDRH
                        STA DLPHTBL,X
                        LDA YEXTBL,X                             ;GET LOWER ZONE
         4A             LSR A
         4A             LSR A
         4A             LSR A
         4A             LSR A
         C5 AE          CMP TEMP14                               ;COMPARE WITH TOP ZONE
         F0 XX          BEQ DISPL20                              ;SAME, GET NEXT OBJECT
         85 AE          STA TEMP14                               ;SECOND ZONE
         A9 00          LDA #$00
         20 EA D0       JSR GETSTAMP_$D0EA
         20 54 DE       JSR ZONELOAD_$DE54
         38             SEC                                      ;COMPUTE OFFSET TO SECOND
ENTRY
         A5 BA          LDA TADDRL
         FD XX XX       SBC DLPLTBL,X
         95 XX          STA DL2PTBL,X
*
DISPL20:
         CA             DEX
         10 XX          BPL DISPL1                               ;DONE?
         60             RTS

ADDRESS $DE54 WAS BEFORE THIS BLOCK
###############################################################################




ZONELOAD:
DE54     86 AD          STX TEMP13                               ;SAVE X
DE56     A6 AE          LDX TEMP14                               ;GET ZONE
                        INC ZONOBJC,X                            ;INC ZONE OBJECT COUNT
                        LDA ZONDLAL,X                            ;ZONE DISPLAY LIST ADDRESS
                        STA TADDRL
                        LDA ZONDLAH,X
                        STA TADDRH
                        LDA ZONLINE,X                            ;ZONE START LINE NUMBER
                        STA TEMP12
                        LDA ZONOBJLL,X                           ;ZONE OBJECT LIST ADDRESS
                        STA TADDR1L
                        LDA ZONOBJLH,X
                        STA TADDR1H
*
         A0 7C          LDY #$7C                                 ;FIND A FREE ENTRY
9696
ZONLD0:
         A2 00          LDX #$00
         B1 BA          LDA (TADDRL),Y
         C9 FF          CMP #$FF                                 ;TEST IF SECTION FULL
         D0 08          BNE ZONLD1                               ;HAS ROOM
         C8             INY
         C0 80          CPY #$80                                 ;TEST IF AT END
         90 F3          BCC ZONLD0
         4C XX XX       JMP ZONLD90                              ;NO ROOM IN ZONE

*
ZONLD1:
         0A             ASL A                                    ;SHIFT UNTIL EMPTY SPOT FOUND
         90 04          BCC ZONLD2
         E8             INX                                      ;COUNT BITS
         D0 FA          BNE ZONLD1
*
```

```
ZONLD2:
                        LDA FREEMSK,X                     ;UNFREE ENTRY
         11 BA          ORA (TADDRL),Y
         91 BA          STA (TADDRL),Y
         98             TYA                               ;COMPUTE OFFSET OF FREE ENTRY
         38             SEC
         E9 7C          SBC #$7C
         0A             ASL A
         0A             ASL A
         0A             ASL A                             ;8 X FREE LIST WORD NO.
         85 AB          STA TEMP11
         8A             TXA
         18             CLC
         65 AB          ADC TEMP11
         85 AA          STA TEMP10                        ;FREE ENTRY NUMBER
         0A             ASL A
         0A             ASL A                             ;X4 BYTES PER ENTRY
                        ADC TADDRL                        ;ADDRESS OF FREE ENTRY
                        STA TADDRL
*
                        LDY #$00
                        LDA TEMP19                        ;STAMP LOW ADDRESS
                        STA (TADDRL),Y
                        LDA TEMP15                        ;WIDTH AND PALETTE
                        INY
         91 BA          STA (TADDRL),Y
                        INY
                        LDA TEMP12                        ;COMPUTE OFFSET OF HIGH STAMP
ADDR
                        SEC
                        SBC TEMP18                        ;YPOS
         18             CLC
                        ADC TEMP20                        ;STAMP HIGH
         91 BA          STA (TADDRL),Y                    ;STAMP HIGH
         C8             INY
                        LDA TEMP17                        ;HPOS
         91 BA          STA (TADDRL),Y
*
                        LDY TEMP10
                        LDA TEMP13                        ;OBJECT NUMBER
                        STA (TADDR1L),Y                   ;INTO ZONE OBJECT LIST
                        JMP ZONLDX                        ;DONE
*
ZONLD90:
                        LDA #$00                          ;NO ROOM IN ZONE
                        STA TADDRL
                        STA TADDRH
*
ZONLDX:
                        LDX TEMP13
         60             RTS
*
*
******************************************
*
*  END OF RDISP.S
*
*
          EJE

******************************************************
############################################################################

###################################################################################
############################################################################
```
ADDRESS $B473 BEFORE THIS BLOCK
```
WSB1:                                                     ;DONE WITH BRAIN SETUP

***********PROGS
*         PROGS: ALLOCATE PNUM NULL PROGS

                        STX PPTR                          ;POINTER TO START OF PROGS
```

```
*          PROG LOOP

                         LDY PNUM                                      ;LOOP THRU ALL PROGS
                         BNE WSPGO                                     ;AT LEAST 1 PROG - DISTRIBUTE
THEM

*          NO PROGS

                         LDA #NULLCODE
         9D 8C 1E        STA CRTBL,X                                   ;STORE A NULL OBJECT
         E8              INX
         4C XX XX        JMP WSPCONT                                   ;SET UP GLOBAL PROG VARIABLES

WSPGO:
         88              DEY                                           ;Y RUNS UNTIL NEGATIVE

WSPLOOP:
         A9 00           LDA #NULLCODE
         9D 8C 1E        STA CRTBL,X                                   ;STORE NULL OBJECT
         E8              INX
         88              DEY
         10 F7           BPL WSPLOOP                                   ;MORE PROGS TO ALLOCATE

WSPCONT:
           ;SET UP VARIABLES GLOBAL TO PROGS
                  ;THERE ARE NONE SO FAR

**********MISSILES
33DA?    86 DC           STX MPTR                                      ;POINTER TO START OF MISSILES
3270?    86 D8
*          COMPUTE NUMBER OF MISSILES TO ALLOCATE: TOTAL OF THE 3 MISSILE TYPES
         18              CLC
                         LDA EMNUM
                         ADC CMNUM
                         ADC TMNUM
         A8              TAY                                           ;Y IS INDEX FOR LOOP THRU ALL
MISSILES
         D0 0A           BNE WSMGO                                     ;AT LEAST 1 MISSILE -
DISTRIBUTE THEM

*            NO MISSILES

         A9 00           LDA #NULLCODE
         9D 8C 1E        STA CRTBL,X                                   ;STORE A NULL OBJECT
         E8              INX
         4C XX XX        JMP WSMCONT                                   ;GO TO GLOBAL MISSILE VARIABLE
SETUP

WSMGO:
                         DEY                                           ;Y RUNS UNTIL NEGATIVE

WSMLOOP:
                         LDA #NULLCODE
         9D 8C 1E        STA CRTBL,X                                   ;STORE NULL OBJECT
                         INX
                         DEY
                         BPL WSMLOOP                                   ;MORE PROGS TO ALLOCATE

WSMCONT:                 ;SET UP VARIABLES GLOBAL TO MISSILES

*          CURRENTLY WE HAVE NONE TO SET UP
WSM1:                                                                  ;DONE WITH MISSILE SETUP

*@@@@@@@@@@@@@@@
*@@        THIS IS A HACK TO ELIMINATE LOADER PROBLEMS
*@@@@@
*@@        SET UP NULL THINGS IN EACH ZONE AT THE END OF DISPLAY LIST

                         JMP SHOOM                                     ;SKIP OVER THIS
;;;;;;;;;;;;;;;;;;

                         LDA #$01
```

```
               STA STTBL,X
               LDA #$05
               STA CRTBL,X
               LDA #$B0
               STA XTBL,X
               LDA #$B5
               STA XEXTBL,X
               LDA #$01
               STA YTBL,X
               LDA #$05
               STA YEXTBL,X
               INX

               LDA #$01
               STA STTBL,X
               LDA #$05
               STA CRTBL,X
               LDA #$B0
               STA XTBL,X
               LDA #$B5
               STA XEXTBL,X
               LDA #$11
               STA YTBL,X
               LDA #$15
               STA YEXTBL,X
               INX

               LDA #$01
               STA STTBL,X
               LDA #$05
               STA CRTBL,X
               LDA #$B0
               STA XTBL,X
               LDA #$B5
               STA XEXTBL,X
               LDA #$21
               STA YTBL,X
               LDA #$25
               STA YEXTBL,X
               INX

LDA     #$01
STA     STTBL,X
LDA     #$05
STA     CRTBL,X
LDA     #$B0
STA     XTBL,X
LDA     #$B5
STA     XEXTBL,X
LDA     #$31
STA     YTBL,X
LDA     #$35
STA     YEXTBL,X
INX

LDA     #$01
STA     STTBL,X
LDA     #$05
STA     CRTBL,X
LDA     #$B0
STA     XTBL,X
LDA     #$B5
STA     XEXTBL,X
LDA     #$41
STA     YTBL,X
LDA     #$45
STA     YEXTBL,X
INX

LDA     #$01
STA     STTBL,X
LDA     #$05
STA     CRTBL,X
```

```
LDA     #$B0
STA     XTBL,X
LDA     #$B5
STA     XEXTBL,X
LDA     #$51
STA     YTBL,X
LDA     #$55
STA     YEXTBL,X
INX

LDA     #$01
STA     STTBL,X
LDA     #$05
STA     CRTBL,X
LDA     #$B0
STA     XTBL,X
LDA     #$B5
STA     XEXTBL,X
LDA     #$61
STA     YTBL,X
LDA     #$65
STA     YEXTBL,X
INX

LDA     #$01
STA     STTBL,X
LDA     #$05
STA     CRTBL,X
LDA     #$71
LDA     #$B0
STA     XTBL,X
LDA     #$B5
STA     XEXTBL,X
STA     YTBL,X
LDA     #$75
STA     YEXTBL,X
INX

LDA     #$01
STA     STTBL,X
LDA     #$05
STA     CRTBL,X
LDA     #$81
LDA     #$B0
STA     XTBL,X
LDA     #$B5
STA     XEXTBL,X
STA     YTBL,X
LDA     #$85
STA     YEXTBL,X
INX

LDA     #$01
STA     STTBL,X
LDA     #$05
STA     CRTBL,X
LDA     #$91
LDA     #$B0
STA     XTBL,X
LDA     #$B5
STA     XEXTBL,X
STA     YTBL,X
LDA     #$95
STA     YEXTBL,X
INX

LDA     #$01
STA     STTBL,X
LDA     #$05
STA     CRTBL,X
LDA     #TEMP1
LDA     #$B0
STA     XTBL,X
```

```
          LDA     #$B5
          STA     XEXTBL,X
          STA     YTBL,X
          LDA     #TEMP5
          STA     YEXTBL,X
          INX
*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

SHOOM             ;;;;;;;;;;;;


*******************
*         WE HAVE FINISHED SETTING UP ALL THE OBJECTS IN THE OBJECT DATA TABLES.
*******************
*         SET LAST ENTRY IN CRTBL TO $FF
                        LDA #$FF
                        STA CRTBL,X
*


**************************
*         NOW DO A LOAD WHICH SETS UP THE DISPLAY LIST AND ZONE LIST************
*       FOR ALL THE OBJECTS                                         *********
*************************

*  FINALLY, READY TO START PLAYING:
                        RTS                            ;END OF WAVESTART SUBROUTINE
*
*********************************************************************************


******************************************
*
*         WAVESTMC  -  SET UP MC-RELATED STUFF AT WAVESTART
*
******************************************
*

WAVESTMC
*   INITIALIZE MC SHOT TABLES AND SHOT TIMER TO ZERO
          LDA     #$00
          STA     SDIRTBL                 ;SET DIRECTION CODE TO ZERO - NULL SHOT
          STA     SDIRTBL+1
          STA     SDIRTBL+2
          STA     SDIRTBL+3
          STA     SSATBL                  ;SET SHOT ANIMATION STEPS TO 0
          STA     SSATBL+1
          STA     SSATBL+2
          STA     SSATBL+3
          STA     MCSTMR                  ;LET MC SHOOT IMMEDIATELY
          STA     MCMTMR                  ;LET MC MOVE IMMEDIATELY

          LDA     #MCSCODE                ;CREATURE TYPE OF MC SHOTS
          STA     SCRTBL                  ;SET SHOT CREATURE TYPES
          STA     SCRTBL+1
          STA     SCRTBL+2
          STA     SCRTBL+3

*  INITIALIZE MC POSITION AND MAKE HIM APPEAR
          LDA     #MCXINIT
          STA     MCXPOS                  ;MC X POSITION
          LDA     #MCXINIT+MCWID
          STA     MCXEX                   ;MC X EXTENT
          LDA     #MCYINIT
          STA     MCYPOS                  ;MC Y POSITION
          LDA     #MCYINIT+MCHEIGHT
          STA     MCYEX                   ;MC Y EXTENT
          LDA     #$00
          STA     SATBL                   ;MC START ANIMATION STEP
          LDA     #$0D                    ;DOWN DIRECTION - D
          STA     MCDIR                   ;MC START DIRECTION - SOUTH
          LDA     #$01
          STA     MCCTMR                  ;INITIALIZE MC COLLISION TIMER
```

```
*
          LDA     #$01        ;INITIALIZE STTBL FOR MC @@@@@@@@@@@@@@@@@@@@@@@@@@
          STA     STTBL                  ; THROWAWAY @@@@@@@@@@@@@@@@@@@@@@@@@@@@
*
          RTS
*


********************************************************************************
********************************************************************************

*%        THIS IS FOR DEVELOPMENT OF WAVESTART ROUTINES ONLY
*%
*%        USE GLOBAL FIND/REPLACE IN THE EDITOR TO CUSTOMIZE THIS
*%        ALSO REMOVE THE *% AT THE BEGINNING OF EACH LINE
*%                WITH A F/*%//A
*%
*%        CREATURE NAME: CREE
*%        CREATURE LETTER: @
*%        NUMBER OF WAVES STORED IN TABLE FOR GLOBAL DATA: %%

**********CREES
*%          STX     @PTR                   ;POINTER TO START OF CREES
*%
*%          LDY     @NUM                   ;LOOP THRU ALL CREES
*%          BNE     WS@LOOP                ;AT LEAST 1 CREE - DISTRIBUTE CREES
*           NO CREES
*%          LDA     #NULLCODE
*%          STA     CRTBL,X                ;STORE A NULL OBJECT
*%          INX
*%          JMP     WS@CONT                ;GO TO GLOBAL CREE VARIABLE SETUP
*%
*%WS@LOOP   JSR     RANDXYBX               ;GET A VALID CREE POSITION
*%          LDA     RANDOMX
*%          STA     XTBL,X                 ;CREE XPOS
*%          CLC
*%          ADC     #@WID                  ;COMPUTE EXTENT
*%          STA     XEXTBL,X               ;CREE X EXTENT
*%          LDA     RANDOMY
*%          STA     YTBL,X                 ;CREE YPOS
*%          CLC
*%          ADC     #@HEIGHT               ;COMPUTE EXTENT
*%          STA     YEXTBL,X               ;CREE Y EXTENT
*%          TYA      ;USE CREE # AS SEED TO GET GOOD DISTRIBUTION OF MOVE TIMERS
*%          AND     #MASK3                 ;GET A NUMBER 0 - 7
*%          STA     MTTBL,X                ;NUMBER OF FRAMES UNTIL MOVE
*%          JSR     RAND2                  ;GET A NUMBER 0 - 2
*%          STA     SATBL,X                ;CREE ANIMATION STEP
*%          JSR     RANDOM
*%          AND     #MASK3
*%          STA     DTTBL,X                ;# MOVES UNTIL DIR CHANGE
*%          AND     #MASK2
*%          STA     DXTBL,X                ;DIRECTION MOVING
*%          LDA     #@CODE
*%          STA     CRTBL,X                ;CREE OBJECT CODE
*         DLPHTBL,DLPLTBL AND DL2PTBL WILL BE SET UP BY THE LOAD AT THE END OF
*                 THE WAVESTRT ROUTINE
*         DONE WITH THIS CREE, ON TO NEXT...
*%          INX                            ;INCREMENT RUNNING POINTER
*%          DEY
*%          BPL     WS@LOOP                ;MORE CREES TO SET UP
*%
*%WS@CONT           ;SET UP VARIABLES GLOBAL TO CREES
*         SET @SPEED - NUMBER OF FRAMES BETWEEN CREE MOVES
*%          LDA     WAVENUM                ;CURRENT WAVE NUMBER
*%          CMP     #%%                    ;ONLY HAVE %% WAVES IN TABLE
*%          BCC     LOOK@SP                ;LOOK UP @SPEED FROM TABLE
*         WE ARE ABOVE WAVE %%, SET @SPEED TO #@SPMAX
*%          LDA     #@SPMAX
*%          STA     @SPEED
*%          JMP     WS@1
*%LOOK@SP   TAY                            ;PUT WAVE NUMBER IN Y
*%          LDA     @SPTBL-1,Y   ;LOAD STARTING @SPEED - USE -1 BECAUSE NO WAVE
*%          STA     @SPEED
*%WS@1                                    ;DONE WITH CREE SETUP


*%        THIS IS FOR DEVELOPMENT OF WAVESTART ROUTINES ONLY
*%
*%        USE GLOBAL FIND/REPLACE IN THE EDITOR TO CUSTOMIZE THIS
*%        ALSO REMOVE THE *% AT THE BEGINNING OF EACH LINE
*%                WITH A F/*%//A
*%
*%        CREATURE NAME: CREE
*%        CREATURE LETTER: @
*%        NUMBER OF WAVES STORED IN TABLE FOR GLOBAL DATA: %%

*****************************************

**********  END OF RWAVE.S **************

          EJE
#######################################################################################

########################################################################################


****************************************************************************************
*                                                                                      *
*         GAMESTRT   -- VERY FIRST ROUTINE IN A GAME, INITIALIZES VARIOUS THINGS       *
*                                                                                      *
****************************************************************************************
*
GAMESTRT
                        JSR CLEARTUN                        ;TURN OFF SOUNDS

                        LDA #$00
                        STA WAVENUM

                        JSR RESETSC_D420                    ;RESET SCORES

* INITIALIZE RANDOM NUMBER GENERATOR  -  THIS SHOULD BE DONE EVERY SO OFTEN

                        LDA FRMCNT                          ;PSEUDO-RANDOM AT THIS POINT
                        EOR RNDM+1                          ;RANDOM NUMBER REGISTERS
                        STA RNDM

                        JSR WAVESTRT                        ;SET UP FOR START OF PLAY
                        JSR DISPINIT
                        JSR DISPLOAD

MAIN2:
                        BIT MSTAT                           ;INIT KERNEL
                        BVS MAIN2
                        BRK                                 ;ENTER KERNEL
                        NOP
                        CLI                                 ;TURN ON INTERRUPTS

*         MAKE CREATURES APPEAR, MAYBE MOVE, BUT DON'T START NORMAL ACTION
                        SEI                                 ;NO INTERRUPTS
                        LDX #WSWAIT                         ;LOOP FOR WSWAIT FRAMES
WSWLOOP:
   STX TEMP16                 ;SAVE X
*         NOW DO SOME COLOR CYCLING OF VARIOUS PALETTES

         20 XX XX       JSR CHKOBJ

                        LDX TEMP16                          ;IT WAS SAVED HERE
         E0 10          CPX #$10
         10 XX          BPL WSWLOOP1

*         WE ARE LESS THAN $10 FRAMES TILL ACTION

         20 XX XX       JSR WAVESTMC
         8A             TXA                                 ;A HAS A NUMBER F TO 0
         E9? 10         SBC #$10
         49 FF          EOR #$FF
         18             CLC
         69 01          ADC #$01
         85 21          STA P0C1
         85 22          STA P0C2
         85 23          STA P0C3

WAVEEND:
D000     20 56 E3       JSR $E356
D003     A9 06          LDA #SRACKA
D005     20 95 E3       JSR DOTUNE
D008     20 F9 F6       JSR DISPINIT
D00B     20 86 DC       JSR $DC86
D00E     4C 00 B0       JMP $B000



WSWLOOP1:
                        JSR DISPINIT
                        JSR DISPLOAD
                        BRK
                        NOP

         A6 B0          LDX TEMP16                                ;RESTORE X
                        DEX
                        BPL WSWLOOP

*         NOW START ACTION
                        JSR WAVESTMC                              ;SET UP MC
                        JSR MARINIT                               ;RESET PALETTES

                        JMP MAIN                                  ;GO!
*
ADDRESS $D364 WAS AFTER THIS






ADDRESS $D3A7 WAS BEFORE THIS
                        JSR MCSHOOT                               ;MOVE MC SHOTS, CHECK FOR HITS
                        JSR MCMOV                                 ;MOVE MAN, CHECK FOR
COLLISIONS
                        JSR CHKOBJ                                ;CHECK EACH OBJECT, POSSIBLY
ACT

*         CHECK FOR WAVE END
                        LDA CRELEFT
                        BNE KEEPGOIN
*         WAVE IS OVER
         20 00 D0       JSR WAVEEND                               ;DO SOMETHING FANCY

         4C 30 D6       JMP INIT                                  ;FOR NOW,
RESTART@@@@@@@@@@@@@

*

KEEPGOIN                ;WAVE CONTINUES INTO NEXT FRAME
*         UPDATE GLOBAL VARIABLES IF NECESSARY



*         FOR NOW:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
          JSR     DISPINIT               ;RELOAD
          JSR     DISPLOAD
          CLI                            ;INTERRUPTS OK

          BRK                            ;GO TO KERNEL
          NOP

          JMP     MAIN
*
```

ADDRESS $D73D WAS AFTER THIS

```
################################################################################
```
ADDRESS $DA5D WAS BEFORE THIS BLOCK

```
********************************************************************
*
*          CHKOBJ  -- CHECK OBJECTS LOOP
*                  LOOK AT EACH OBJECT, AND IF NECESSARY, MOVE IT
*
********************************************************************
*
CHKOBJ:
                         LDX #$01                                     ;OBJECT INDEX FOR FIRST OBJECT
(GRUNT)

*          X IS THE OBJECT INDEX: A COUNTER INTO THE OBJECT DATA TABLES
*
LOOP:
         BD 8C 1E       LDA CRTBL,X                                  ;GET CREATURE CODE
         F0 XX          BEQ OBJCONT                                  ;NULL OBJECT IF 0
         C9 FF          CMP #$FF
         F0 XX          BEQ DONE                                     ;END OF TABLE IF $FF
*
                         DEC MTTBL,X                                  ;DEC MOVEMENT TIMER
         30 XX          BMI MOVE                                     ;TIME HAS COME TO MOVE
```

CODE GOES ASTRAY

```
*  MOVE RETURNS TO HERE:

D799?    E8             INX                                          ;INCREMENT OBJECT INDEX: NEXT
OBJECT
D79A?    4C 3D D7       JMP $D73D                                    ;CHECK THE NEXT OBJECT

*
DONE:
D79D     60             RTS
```

ADDRESS $921D WAS AFTER THIS BLOCK
```
################################################################################
```

```
##############################################################################################
```
ADDRESS $9231 WAS BEFORE THIS BLOCK
```
********************************************
*          THROWAWAY GARBAGE:
*
*          MARINIT - INIT MARIA STUFF
MARINIT:
         A9 FF          LDA #$FF                                     ;SET UP GRAPHICS MODE
                         STA CTRL
M1:
                         EQU MARINIT                                  ;CODE WHICH MAY HAVE TO BE
CHANGED
                                                                      ;IN THE EMULATOR DUE TO MARIA
DIFFERENCES

         A9 00          LDA #$00
                         STA CTLSWA                                   ;INIT JOYSTICK PORT

         A9 00          LDA #$00                                     ;BACKGROUND COLOR
```

```
                         STA BACKGRND

         20 30 D6        JSR PALINIT                                  ;SET UP PALETTES

         60              RTS


##############################################################################
ADDRESS $D682 WAS BEFORE THIS BLOCK
         60              RTS
*

**********************************************************************
*                                                                    *
*          CONV- CONVERT DATA INTO Si MARIA FORM  (MUNG-O-RAMA)      *
*                                                                    *
**********************************************************************

;          CONVERTS GRAPHICS DATA FROM $4000 TO $5FFF
;          FROM THE FORMAT 76 54 32 10     TO        10 32 54 76

GRAPH      EQU $4300
END        EQU $6200     ;SHOULD BE ENLARGED LATER

CONV:
                         NOP                                          ;FOR SI MARIA, SHOULD BE A NOP
         A9 60           LDA #$60
                         STA CONV                                     ;RTS OUT THIS SUBROUTINE,
                                                                      ;   IT HAS BEEN USED ONCE
         A9 00           LDX #0
                         LDA #H(GRAPH)
                         STA LOAD+2
                         STA SAVE+2
LOAD:
                         LDA GRAPH,X
                         STA CONVTEMP
         29 C0           AND #$C0
         18              CLC
         2A              ROL A      ROL: 2A 26 36 2E 3E
         2A              ROL A
         2A              ROL A                                        ;BITS 7,6 IN PLACE
                         ASL CONVTEMP
                         ASL CONVTEMP
SK2:
                         ASL CONVTEMP
         90 XX           BCC SK3
         09 08           ORA #$8                                      ;ADD BIT 5

SK3:
                         ASL CONVTEMP
         90 XX           BCC SK4
         09 04           ORA #$4                                      ;ADD BIT 4

SK4:
                         ASL CONVTEMP
         90 XX           BCC SK5
         09 20           ORA #$20                                     ;ADD BIT 3

SK5:
                         ASL CONVTEMP
         90 XX           BCC SK6
         09 10           ORA #$10                                     ;ADD BIT 2

SK6:
                         ORA CONVTEMP                                 ;OR IN BITS 1 AND 0

SAVE:
                         STA GRAPH,X
                         INX
                         BNE LOAD
                         INC LOAD+2
                         INC SAVE+2
                         LDA LOAD+2
```

```
                        CMP #H(END)
                        BNE LOAD
         60             RTS

CONVTEMP:
                        .BYTE $00                                    ;TEMP, THROWAWAY VARIABLE



***********************************************
***********************************************
***********************************************


*         END OF RMAIN.S
          EJE


##################################################################################


**************************************************
*
*         CALL THE SOUND DRIVER ROUTINE
*
*                       JSR TUNER                                    ;ROUTINE IN RSOUNDS.S
*


*
*         END OF KERNEL
*

*         RESTORE THE REGISTERS THAT WERE SAVED AT THE BEGINING
*         AND DO AN RTI
          PLA
          TAY
          PLA
          TAX
          PLA
          RTI

******************************************

*         END OF RKERNEL.S
          EJE


ADDRESS $F1FF WAS BEFORE THIS BLOCK
******************************
*                            *
*         SOUND DATA         *
*                            *
******************************
*

*         NUMBERS (PRIORITY OF EACH SOUND)
NOTSOUND  EQU     0
GULPSND   EQU     1
MCDIESND  EQU     2

*
*         TABLES POINTING TO DATA:
*                 ( START WITH SOUND # 1 )

DUR               ;# OF FRAMES BETWEEN SOUND REGISTER CHANGES FOR EACH SOUND
          DB      1,1

SCNTRLS           ;AUDC0/1 TO USE FOR EACH SOUND
          DB      $44,$44

LENGTH            ;LENGTH OF SOUND TABLES - # BYTES OF V'S OR F'S FOR EACH SOUND
```

```
          DB      6,6

LVTABL            ;LOW BYTES OF VOLUMES FOR EACH SOUND
          DB      L(GULPVOL),L(MCDIEVOL)

LFTABL            ;LOW BYTES OF FREQUENCIES FOR EACH SOUND
          DB      L(GULPFRQ),L(MCDIEFRQ)

*
*         ACTUAL SOUND DATA TABLES: THIS SHOULD NOT CROSS A PAGE BOUNDARY
*
SOUNDS:
GULPFRQ   DB      1,2,3,4,5,6
GULPVOL   DB      9,8,7,6,5,4
MCDIEFRQ  DB      5,5,5,5,5,5
MCDIEVOL  DB      4,4,4,4,4,4

*         END OF RDATA.S

*
*         THIS ALSO THE END OF CONGLOMERATE FILE ROB.S
*         RSTAMPS.S MUST BE ASSEMBLED SEPARATELY AND LINKED
####################################################################################
```

```
####################################################################################
```
ADDRESS $9679 WAS BEFORE THIS BLOCK

```
9600
9696




9696

SQMOV:
                        DEC MISCTBL,X              ;SEE IF IT'S TIME TO GIVE BIRTH
                        BPL SQMOV1                 ;BRANCH BY IF NOT TIME

*         HERE CREATE AN ENFORCER OR TANK IN THE SMALL ANIMATION  **************

SQMOV1:
                        LDA CRTBL,X                ;GET CREATURE TYPE
                        CMP #7                     ;SEE IF IT'S A QUARK
                        BCS QMOV                   ;BRANCH IF QUARK
                        CLC
                        LDA XTBL,X                 ;GET X POSITION
                        ADC DXTBL,X                ;ADD dX TO CURRENT POSITION
                        STA XINTEND
                        LDA XEXTBL,X               ;GET X EXTENT
                        ADC DXTBL,X
                        STA XXINTEND
                        LDA YTBL,X                 ; GET YPOS
                        ADC DYTBL,X                ;ADD dY
                        STA YINTEND
                        LDA YEXTBL,X               ;GET Y EXTENT
                        ADC DYTBL,X
                        STA YXINTEND

*         HERE INC OR DEC DX AND DY RANDOMLY TO CAUSE A CURVE           ********

*         BE SURE THAT NEW X AND Y POSITIONS ARE SENT TO THE TBL'S      ********

          JMP OBJCONT


QMOV:
                        CLC
                        LDY DXTBL,X                ;GET THE DIRECTION
                        LDA XTBL,X
```

```
                         ADC XDIRTBL,Y              ;MOVE ACCORDING TO THE DIRECTION
                         STA XINTEND
                         LDA XEXTBL,X
                         ADC XDIRTBL,Y
                         STA XXINTEND

                         LDA YTBL,X
                         ADC YDIRTBL,Y              ;MOVE ACC TO DIR
                         STA YINTEND
                         LDA YEXTBL,X
                         ADC YDIRTBL,X
                         STA YXINTEND

*         CHANGE ANIMATION STEP

                         DEC SATBL,X                ;DECRIMENT THE ANIMATION
                         BPL QMOV5                  ;OK IF NONEGATIVE
                         LDA #$04                   ;HIGHEST QUARK ANIMATION STEP
                         STA SATBL,X                ;STORE NEW ANIMATION

QMOV5:
                         LDA #0                     ;PUT A ZERO IN THE DIRECTION

*      HERE CALL THE ANIMATOR AND THE  UNLOADER                    **************

                         DEC DYTBL,X                ;DECREMENT DIR CHANGE TIMER
                         BPL QMOV1                  ;BRANCH BY IF NOT DIR CHANGE TIME
                         JSR RANDOM                 ;GET A RANDOM NUMBER
                         AND #MASK3                 ;FILTER IT TO A 0-7 DIRECTION
                         STA DXTBL,X                ;STORE NEW DIRECTION

QMOV1:
                         NOP
                         JMP OBJCONT

*********************************************************
*
*         ETMOV   -- MOVE ENFORCERS AND TANKS
*
*********************************************************
*
*         USE OF OBJECT DATA TABLE ENTRIES:
*                      --ENFORCER--               --TANK--
*         DXTBL   -      DELTA X               DIRECTION (0-7)
*         DYTBL   -      DELTA Y             DIRECTION CHANGE TIMER
*         DTTBL   - DIRECTION CHANGE TIMER          NOT USED
*         SATBL   -        0 ALWAYS            ANIMATION (0-3)
*         MISCTBL -           # MOVES UNTIL NEXT SHOT
*         CRTBL   -          8                          9
*
**********
*

ETMOV     JMP     OBJCONT                             ;NOT READY YET
*         TIMER USED FOR FIRING TIMES

******************************************
*
*         BMOV    -- MOVE BRAINS
*
******************************************
*
*         USE OF OBJECT DATA TABLE ENTRIES:
*
*         DXTBL   -  DIRECTION (0-7)
*         DYTBL   -  TARGET NUMBER (0 IF MC)
*         DTTBL   -  FAMILY SEEK TIMER
*         SATBL   -  ANIMATION (0-3)
*         MISCTBL -  # OF MOVES UNTIL NEXT SHOT
*         CRTBL   -  A
*
**********
*
```

```
BMOV:
                        LDA STTBL,X                                 ;LOAD THIS BRAIN'S CURRENT
STATUS
         29 03          AND #$03                                    ;SEE IF BOTTOM 2 BITS ARE SET
                        BNE BMOV01                                  ;WILL BE 0 IF BRAIN IS DEAD AND
GONE
                        LDA #NULLCODE                               ;STTBL IS 0, SO NULL OUT CRTBL
                        STA CRTBL,X
         4C XX XX       JMP OBJCONT                                 ;NEVER WORRY ABOUT THIS BRAIN
ANYMORE

BMOV01:
                        AND #00000010B                              ;GET ONLY BIT 1 - 'DYING' FLAG
                        BEQ BOK
                        JMP BDYING                                  ;BRAIN IS DYING - DON'T MOVE

*         BRAIN IS ALIVE AND WELL

BOK:
                        DEC MISCTBL,X                               ;DECREMENT # OF MOVE TILL
SHOOT
                        BPL BMOV1                                   ;BRANCH IF NOT YET TIME FOR A
C M

*         HERE CREATE A CRUISE MISSILE AT THE PLACE WHERE THIS BRAIN IS *********
*         AND RESET MISCTBL TO THE TIME FOR THE NEXT CM                 *********

BMOV1:
                        DEC DTTBL,X                                 ;COUNT MOVES UNTIL TIME TO
LOOK AT FAM
                        BMI BMOV8                                   ;IS IT TIME YET
                        JMP BMOVST                                  ;JUST PLOW ON AHEAD

BMOV8:
                        LDA #$03                                    ;RESET TIMER
                        STA DTTBL,X

*         LOOK AT THE HUMAN POINTED TO IN THE BRAIN'S DYTBL.
*         IF THE HUMAN IS DEAD POINT TO THE NEXT HUMAN BUT CHASE MC THIS TIME.
*         THE NET RESULT IS THAT IF THERE IS A FAMILY MEMBER LEFT, THE
*         BRAIN WILL EVENTUALLY LATCH ON TO IT, OTHERWISE IT WILL CHASE MC.

                        LDY DYTBL,X                                 ;GET THE POINTER TO THE TARGET
                        LDA CRTBL,Y
                        BNE BCHASE                                  ;WE ARE ONTO A LIVE FAMILY
MEMBER
                        INC DYTBL,X                                 ;WE ARE ONLY CHASING MC
                        INY
                        CPY HPTR                                    ;END OF THE HUMANS
                        BMI BCHASENF                                ;RESET THE AIM IF WE GOT TO THE
END
                        LDA FPTR                                    ;START AT BEGINNING OF FAMILY
                        STA DYTBL,X

BCHASENF:
                        LDA #$00                                    ;CHASE MC WHEN FAMILY IS DEAD
BD6D                    TAY                                         ;WHEN Y IS 0 IT IS POINTING AT
MC

CODE GOES ASTRAY
```
ADDRESS $BD6E WAS AFTER THIS BLOCK
##

```
################################################################
```
ADDRESS $BDF1 WAS BEFORE THIS BLOCK
```
                        LDA STICKTBL,Y                              ;GET 0-7 FORM
                        LDY TEMP1                                   ;GET BACK POINTER TO HUMAN
                        STA DXTBL,X                                 ;STORE THE DIRECTION FOR THE
BRAIN
                        JMP BMOVST
```

```
BPROG:
                        LDA STTBL,Y                              ;GET THE HUMAN'S STATUS
         29 03          AND #$03                                 ;CHECK BOTTOM 2 BITS
         C9 01          CMP #$01                                 ;WE ONLY WANT HEALTHY HUMANS
         DO XX          BNE BMOVST

*         IT'S PROGGING TIME      THE PROGEE IS POINTED TO BY Y
*         THE FAMILY MEMBER HAS BEEN SEVERELY KILLED
*         ENTER THE FAMILY DYING SOUND INTO THE SOUND QUEUE

                        LDA #SSKULL0
         20 95 E3       JSR DOTUNE
                        LDA #SSKULL1
         20 95 E3       JSR DOTUNE

*         NOW SET FAMILY ANIMATION TO #0 (SKULL) WITH HIGH BIT SET.
*         SET THE FAMILY CODE TO BE A MOMMY - #MOCODE
*         ALSO SET THE 'DYING' BIT IN STTBL
*         ALSO SET THE DIRECTION TO 8
*         THE FAMILY WILL START DYING NEXT FRAME

                        LDA #$00                                 ;SKULL ANIMATION STEP
                        STA SATBL,Y
                        LDA #MOCODE
                        STA CRTBL,Y
                        LDA #$02                                 ;BIT 1 IS ON
                        ORA STTBL,Y
                        STA STTBL,Y                              ;SET BIT 1 IN STATUS ENTRY
                        LDA #$08
                        STA DXTBL,Y                              ;SET DIRECTION TO DYING
DIRECTION
                        LDA #$0A                                 ;WAIT A WHILE SO THAT IT
BOES'NT LOOK
                        STA DTTBL,X                              ;AT SKULL FOR DIRECTION
                        JMP OBJCONT

*         MOVE THE BRAIN A STEP

BMOVST:
         BC D4 1B       LDY DXTBL,X                              ;GET THE CURRENT DIRECTION
                        LDA XTBL,X                               ;GET X POS
         18             CLC
         69 XX          ADC XDIRTBL,Y                            ;ADD ONE STEP
                        STA XINTEND
                        LDA XEXTBL,X
         18             CLC
         69 XX          ADC XDIRTBL,Y
                        STA XXINTEND

                        LDA YTBL,X                               ;GET Y POS
         18             CLC
         69 XX          ADC YDIRTBL,Y                            ;MOVE ONE STEP
                        STA YINTEND
                        LDA YEXTBL,X
         18             CLC
         69 XX          ADC YDIRTBL,Y
                        STA YXINTEND

*         RESET MOVE TIMER

                        LDA BSPEED                               ;GET TIME TO MOVE
                        STA MTTBL,X                              ;STORE IT FOR NEXT MOVE

*         CHANGE ANIMATION STEP

                        DEC SATBL,X                              ;DECREMENT THE ANIMATION
                        BPL BMOV5                                ;OK IF NON-NEGATIVE
                        LDA #$03                                 ;HIGHEST BRAIN ANIMATION STEP
                        STA SATBL,X                              ;NEW ANIMATION STEP

                        JSR GETEXTEN                             ;@@@@@@@@@@ TEMPORARILY
RECHECK EXTENTS
                        LDA XINTEND
         18             CLC
         69 XX          ADC TEMP11
                        STA XXINTEND
                        LDA YINTEND
                        ADC TEMP12
                        STA YXINTEND

BMOV5:
                        JSR CHKINTBD                            ;KEEP IT ON THE SCREEN
*         HERE JUMP TO THE UNLOADER                      ******************

*         STORE THE NEW POSITION

                        LDA XINTEND
                        STA XTBL,X
                        LDA XXINTEND
                        STA XEXTBL,X
                        LDA YINTEND
                        STA YTBL,X
                        LDA YXINTEND
                        STA YEXTBL,X
                        JMP OBJCONT                             ;NOT READY YET
BDYING   :                                                      ;FOR NOW JUST MAKE IT GO AWAY
*         START BRAIN DEATH SOUND

                        LDA #SCREDIE
         20 95 E3       JSR DOTUNE

                        LDA #$00                                ;DEAD STATUS
                        STA STTBL,X
                        DEC CRELEFT                             ;ONE LESS CREATURE
                        JMP OBJCONT
*

******************************************
*
*         PMOV    -- MOVE PROGS
*
******************************************
*
*         USE OF OBJECT DATA TABLE ENTRIES:
*
*         DXTBL   -          NOT
*         DYTBL   -          YET
*         DTTBL   -          DEFINED
*         SATBL   -
*         MISCTBL -
*         CRTBL   -
*
**********
*

PMOV:
                        JMP OBJCONT                             ;NOT READY YET
*

******************************************
*
*         MMOV    -- MOVE OBJECT MISSILES
*
******************************************
*
*         USE OF OBJECT DATA TABLE ENTRIES:
*
*         DXTBL   -         NOT
*         DYTBL   -         YET
*         DTTBL   -         DEFINED
*         SATBL   -
*         MISCTBL -
*         CRTBL   -
*
**********
*
```

```
MMOV:
         4C XX XX       JMP OBJCONT                                 ;NOT READY YET
*
#################################################################



;THERE SHOULD BE A ROUTINE TO ADD TO THE PROPER PLAYER

*         START FAMILY PICKUP SOUND

                        LDA #SFPICK
                        JSR DOTUNE

*         FINALLY, RETURN WITH CURRENT LEVEL IN A

                        LDA FAMLEVEL
         60             RTS

******************************************
*         THROWAWAY ROUTINES
******************************************

*******************
*
*         SETSTAT - THROWAWAY ROUTINE TO SET STTBL,X TO 1
*
SETSTAT:
                        LDA #$01
                        STA STTBL,X
         60             RTS
*



********* END OF RSUBR.S ***************
          EJE
#########################################################################
```