

```
*****
;
; Note: Markers like [1] indicate bytes that can be
;       saved by obvious (or nearly so) optimizations,
;       and [0] indicates that cycles can be saved.
;
;       Some of the optimizations can't be made because
;       they will change locked address, and some are
;       so non-obvious that I can't figure out what I
;       was thinking when I marked them!
;
*****

; LOCKED ADDRESSES -- THESE *MUST* BE CORRECT OR SOME CARTRIDGES WILL NOT RUN!
;
; In order to allow easy verification, these addresses have labels
; starting with an 'A' and containing the locked address.
;
; These are the important addresses (like vectors), plus a few used by the
; small-time offenders.
;
;      0000      0066      080B      18A3
;      0008      0069      08C0      18D4.
;      0010      006A      143B      18F7*
;      0018      006C      144D      1968
;      0020      01B1*     1463      196B
;      0028      01D5      14B4      1CCA.
;      0030      023B.     14C1      1D43
;      0038      02EE      14C3      1D47
;      003B.     07E8*     15A3      1F61
;                               158B*
;
; ...and these are the addresses used by three of the four Interphase
; games, most of which have vectors available.
;
;      0213.     114A*.     1BAA*     1D3E*.
;      025E.     18E9.      1C27.     1D57.
;      027F.     1979*.     1C4F*     1D5A.
;      0300.     1B1D*.     1C82*.     1D60*.
;      1105*     1BA3.      1D01*.     1D66*.
;
; These two were called from Boulder Dash, but the call instructions
; themselves may be unreachable from the rest of the game.
;
;      01C1*     026A*
;
; Any address with a '*' is only referenced by ONE game that I can find.
; Note that Interphase games account for 11 out of 15 of these. Any address
; with a '.' has a vector available. Note also that all but two Interphase-
; only addresses (18 of 20) have vectors! Oh the humanity!
;
; These are in the "OS7SYM" file which was provided to Adam programmers:
;
;FREQ_SWEEP      EQU      000FCH !
```

```
;ATN_SWEEP      EQU    0012FH !
;DECLSN         EQU    00190H !
;DECMSN         EQU    0019BH !
;MSNTOLSN       EQU    001A6H !
```

```
0              EQU    0FFH          ; filler for unused bytes
```

```
;*****
; RAM usage
;*****
```

```
                ORG     7000H

                DS      12          ; default CtlState storage?
                DS      20
SndArray        DS      2          ; music array pointer?
NoiseP          DS      2          ; Pointer to program for noise generator
Tone1P          DS      2          ; Pointer to program for tone1 generator
Tone2P          DS      2          ; Pointer to program for tone2 generator
Tone3P          DS      2          ; Pointer to program for tone3 generator
NoiseCtlShad    DS      1          ; Noise control register shadow

                ORG     73B9H

Stack           DS      1          ; Default initial stack pointer
ParmArea        DS      9          ; Parameter storage for PCOPY parameters
VDP0Shad        DS      1          ; VDP register 0 shadow
VDP1Shad        DS      1          ; VDP register 1 shadow
                DS      1          ; unused?
D73C6           DS      1          ; flag byte?
WrtRAMSprt      DS      1          ; BlkWrtVRAM to RAM sprite attr table if =1
RandSeed        DS      2          ; Random number seed
D73CA           DS      1          ; unknown
D73CB           DS      1          ; unknown
D73CC           DS      1          ; unknown
D73CD           DS      2          ; unknown
D73CF           DS      2          ; unknown
D73D1           DS      2          ; unknown
TimerList       DS      2          ; Pointer to timer list
TimerAux        DS      2          ; Pointer to end of timer aux storage
RawCtlState     DS      20         ; Raw controller state table (2 x 10 bytes)
PulseCnt1       DS      1          ; Pulse counter #1
PulseCnt2       DS      1          ; Pulse counter #2
                DS      1          ; unused?
Joy1Shad        DS      1          ; shadow for joystick #1
Joy2Shad        DS      1          ; shadow for joystick #2
Key1Shad        DS      1          ; shadow for keypad #1
Key2Shad        DS      1          ; shadow for keypad #2
VDPBaseShad     EQU     $          ; shadow for VDP table base addresses
SprtTabShad     DS      2          ; shadow for sprite table VRAM base addr
SprtPatTabShad  DS      2          ; shadow for sprite pattern generator VRAM base addr
NameTabShad     DS      2          ; shadow for name table VRAM base address
PatGenTabShad   DS      2          ; shadow for pattern generator VRAM base addr
```

```
ClrTabShad    DS      2          ; shadow for color table VRAM base address
              DS      2          ; unused?
D73FE         DS      2          ; temp3

;*****
; Cartridge header addresses
;*****

                ORG      8000H

Cart_Sig       DS      2          ; AA55 = title screen, 55AA = no screen
RamSprtTab     DS      2          ; RAM sprite attribute table pointer
RAMSprtIdx     DS      2          ; sprite index table pointer
VDP_Temp       DS      2          ; pointer to temp image storage (up to 40 bytes used)
CtlState       DS      2          ; pointer to controller state table (2 + 2x5 bytes)
Cart_Start     DS      2          ; start of cart code
V_RST_08H      DS      3          ; RST 08H vector
V_RST_10H      DS      3          ; RST 10H vector
V_RST_18H      DS      3          ; RST 18H vector
V_RST_20H      DS      3          ; RST 20H vector
V_RST_28H      DS      3          ; RST 28H vector
V_RST_30H      DS      3          ; RST 30H vector
V_RST_38H      DS      3          ; RST 38H vector
V_NMI          DS      3          ; NMI vector (vertical blank interrupt)
Cart_Title     DS      0          ; Title string "LINE 3/LINE 2/yyyy"

;*****
; Offsets into data blocks
;*****

;                Offsets into RawCtlState

RawCtlLeft     EQU      00H        ; raw left controller state
RawCtlRight    EQU      0AH        ; raw right controller state

;                Offsets into RawCtlLeft and RawCtlRight

RawCtlLFBits   EQU      00H        ; previous left fire bit
RawCtlLFState  EQU      01H        ; left fire button state
RawCtlDBits    EQU      02H        ; previous directional bits
RawCtlDState   EQU      03H        ; directionals state
;              EQU      04H        ; unused?
;              EQU      05H        ; unused?
RawCtlRFBits   EQU      06H        ; previous right fire bit
RawCtlRFState  EQU      07H        ; right fire button state
RawCtlKPBits   EQU      08H        ; previous keypad bits
RawCtlKPState  EQU      09H        ; keypad state

;                Offsets into CtlState table

CtlStateLFlag  EQU      00H        ; left controller flags
CtlStateRFlag  EQU      01H        ; right controller flags
CtlStateLeft   EQU      02H        ; left controller state
CtlStateRight  EQU      07H        ; right controller state
```

```

;          CtlStateLF/CtlStateRF bits

CtlCheckMe    EQU    80H    ; 7    ; if =0, do not check this ctrl at all
;            EQU    40H    ; 6    ; unused?
;            EQU    20H    ; 5    ; unused?
CtlCheckKP    EQU    10H    ; 4    ; check keypad
CtlCheckRFire EQU    08H    ; 3    ; check right fire button
CtlCheckSpinner EQU    04H    ; 2    ; check spinner
CtlCheckDir   EQU    02H    ; 1    ; check directionals
CtlCheckLFire EQU    01H    ; 0    ; check left fire button

```

```

;          Offsets into CtlStateLeft and CtlStateRight

```

```

CtlStateLFire EQU    00H    ; left fire button
CtlStateDir   EQU    01H    ; directionals
CtlStateSpin  EQU    02H    ; spinner value
CtlStateRFire EQU    03H    ; right fire button
CtlStateKey   EQU    04H    ; key code

```

```

;*****
; I/O port addresses
;*****

```

```

IO_KP_Select   EQU    080H    ; Keypad select output port
IO_Joy_Select  EQU    0C0H    ; Joystick select output port
IO_Joy1        EQU    0FCH    ; Joystick 1 input port
IO_Joy2        EQU    0FFH    ; Joystick 2 input port
IO_Sound       EQU    0FFH    ; Sound chip output port
IO_VDP_Data    EQU    0BEH    ; VDP data port
IO_VDP_Addr    EQU    0BFH    ; VDP VRAM address output port
IO_VDP_Status  EQU    0BFH    ; VDP status input port

```

```

ORG    0000H

```

```

;*****
; Everything starts here
;*****

```

```

A0000          LD      SP,Stack    ; Initialize stack pointer
               JR      L006E      ; Go to rest of cold-start code

```

```

;*****
; These are the RST vectors, mixed with some (formerly) wasted bytes
;*****

```

```

P_AddSound
P_StopTimer
P_TestTimer
P_InitRAMSprt
P_CopyRAMSprt  DB      1
               DW      1

A0008          JP      V_RST_08H

```

```

P_InitSound
P_StartTimer

```

```

P_BaseLoad      DB      2
                 DW      1,2

A0010           JP      V_RST_10H

D0482           DB      2
                 DW      -2,1

A0018           JP      V_RST_18H

D064F           DB      2
                 DW      1,-2

A0020           JP      V_RST_20H

D06C1           DB      2
                 DW      2,1

A0028           JP      V_RST_28H

P_InitTimers    DB      2
                 DW      2,2

A0030           JP      V_RST_30H

P_WriteReg      DB      2
                 DW      1,1

A0038           JP      V_RST_38H

```

```

;*****
;      1FFD      Random
;
; Returns a random number in A.  This routine apparently
; uses a pseudo-random shift register algorithm based on
; the XOR of bits 8 and 15.
;*****
A003B
_Random         LD      HL,(RandSeed)
                BIT      7,H          ; Set the carry flag to
                JR      Z,L0048      ; bit 8 XOR bit 15
                BIT      0,H
                JR      Z,L004C
L004F           OR      A            ; Clear carry flag
L0050           RL      L            ; Rotate the carry into HL
                RL      H
                LD      (RandSeed),HL ; Update the random number seed
                LD      A,L          ; Return the LSB
                RET

L0048           BIT      0,H
                JR      Z,L004F

D1AC4 ; DB '7'
L004C           SCF                  ; Set carry flag
                JR      L0050

```

```

;      DE = DE + IX

L0478      PUSH    IX
           EX      (SP),HL
L047C      ADD     HL,DE
           EX      DE,HL
           POP     HL
           RET

P_WrtVRAM
P_ReadVRAM DB      3
           DW      -2,2,2

DB 0

;*****
;      NMI vector
;*****
A0066      JP      V_NMI

;*****
; I'm not really sure what these are for, but aside from the
; jump vectors at the end of the ROM, they are probably the
; only ROM addresses that you should reference directly.
;*****
A0069      DB      60          ; this might mean a 60hz display (NTSC)
A006A      DW      D16AB       ; this points to the font bitmap for 'A'
A006C      DW      D1623       ; this points to the font bitmap for '0'

;*****
;
; First part of cold start code
;
;*****
L006E      LD      HL,(Cart_Sig) ; Check first word of cart for 55AAH
           LD      A,L          ; 8000=55H and 8001=AAH
           CP      55H
           JR      NZ,L0081
           LD      A,H
           CP      0AAH
           JR      NZ,L0081
           LD      HL,(Cart_Start) ; If 55H/AAH, jump into cartridge
           JP      (HL)

L0081      CALL    NoSound      ; Turn off sound
D1AC0      EQU     $+1          ; DB '3'
           LD      HL,0033H     ; Initialize random seed
           LD      (RandSeed),HL
           CALL    InitCtlState ; Clear all controller state flags
;
           XOR     A            ; (A-reg is already = 00H)
D1ABF ; DB '2'
           LD      (D73C6),A     ; Clear some unknown flags
           LD      (WrtRAMSprt),A ; Clear BlkWrtVRAM RAM sprite attr flag
           JP      L1319         ; Go display copyright screen

```

```

;*****
;      PCopy
;
; This routine copies in-line paramters for the subroutines.
;
; The descriptor table entries are 2 bytes each. The first word
; of the descriptor list is the number of descriptors in the list.
;
; (Note: for optimization purposes, the number of descriptors is now a byte)
;
; If an entry is positive, it contains the number of bytes to copy
; to the storage area.
;
; The parameters are 2 bytes each and normally contain the
; address of the data to be copied into the storage area.
;
; If a descriptor table entry is negative, the next parameter
; is taken as a 2-byte literal and stored directly in the data area.
;
; If a parameter word is zero, the next parameter word contains the
; address of the real parameter word to use. This is done before
; checking for a negative descriptor table entry.
;
; ENTRY BC points to parameter descriptor table
;      DE points to parameter storage area
;      (SP+2) = return address of calling routine
; EXIT: DE points to first unused byte of storage area
;*****
PCopy      POP      HL          ; Swap caller's return address
           EX       (SP),HL     ; (the parameter pointer) to
           PUSH     HL          ; the top of the stack
           LD       A,(BC)      ; HL = first descriptor table word
           LD       L,A
           INC      BC
           LD       H,00H
           EX       (SP),HL     ; Swap parameter pointer
           PUSH     DE          ; with storage pointer

;      Get next parameter word

L00A3      LD       E,(HL)      ; DE = next parameter word
           INC      HL
           LD       D,(HL)
           INC      HL
           PUSH     HL          ; Save parameter pointer
           LD       A,E         ; Check if parameter = 0
           OR       D
           JR       NZ,L00B7    ; Branch if = 0

;      Handle zero parameter word

           POP      HL          ; Get back parameter pointer
           LD       E,(HL)      ; DE = next parameter word
           INC      HL

```

```

        LD      D,(HL)
        INC     HL
        PUSH    HL          ; Save parameter pointer
        EX      DE,HL       ; DE = (DE)
        LD      E,(HL)
        INC     HL
        LD      D,(HL)

;      Check sign of descriptor word

L00B7      INC     BC          ; Get MSB of descriptor
          LD      A,(BC)
          RLCA
          JR      NC,L00DA     ; Branch if positive

;      Handle negative descriptor word

          INC     BC          ; Point to next descriptor word
          POP     HL          ; HL = parameter pointer
          EX      (SP),HL     ; Swap with storage pointer
          LD      (HL),E      ; Store DE in storage area
          INC     HL
          LD      (HL),D
          INC     HL
L00C4      POP     DE          ; DE = storage pointer
          EX      (SP),HL     ; Swap it with parameter counter
          DEC     HL          ; Decrement count
          LD      A,H         ; Test for zero count
          OR      L
          JR      Z,L00D6
L00D0      EX      (SP),HL     ; Swap counter back on stack
          PUSH    HL          ; Put storage pointer back on stack
          EX      DE,HL       ; HL = parameter pointer
          JR      L00A3       ; Go back for next parameter

L00D6      POP     HL          ; Get storage pointer
          EX      DE,HL       ; DE = storage pointer, HL = parm pointer
          EX      (SP),HL     ; Put parm pointer back on stack
          JP      (HL)        ; Return to caller

;      Handle positive descriptor word

L00DA      POP     HL          ; Swap parameter pointer
          EX      (SP),HL     ; and storage pointer
          PUSH    HL
          RRCA                ; Restore value of MSB
          LD      H,A         ; H = MSB
          DEC     BC          ; L = LSB
          LD      A,(BC)
          LD      L,A
          EX      (SP),HL     ; HL = storage pointer, TOS = count
          INC     BC          ; Point to next descriptor
          INC     BC
L00E5      LD      A,(DE)     ; Copy next parameter byte
          LD      (HL),A

```

```
        INC     HL
        INC     DE
        EX      (SP),HL      ; HL = count, TOS = storage pointer
        DEC     HL          ; Decrement counter
        LD      A,H          ; Loop until count is zero
        OR      L
        JR      Z,L00F8      ; Branch if count = 0
L00F4    EX      (SP),HL      ; HL = storage pointer, TOS = count
        JR      L00E5        ; Go back to copy next byte

L00F8    POP     HL          ; HL = storage pointer
        JR      L00C4        ; Go check for end of parameter list
```

```
;*****
;
; SOUND ROUTINES BEGIN HERE
;
;*****
```

```
;      Process the duration and twang values?
```

```
L00FC    LD      A,(IX+07H)
        OR      A
        JR      NZ,L010C
        LD      A,(IX+05H)
        DEC     A
        RET     Z
        LD      (IX+05H),A
        RET
```

```
L010C    PUSH    IX
        POP     HL
        LD      DE,0006H
        ADD     HL,DE
        CALL    L0190
        RET     NZ
        CALL    L01A6
        DEC     HL
        LD      A,(HL)
        DEC     A
        RET     Z
        LD      (HL),A
        DEC     HL
        DEC     HL
        LD      A,(IX+07H)
        CALL    A01B1
        INC     HL
        RES     2,(HL)
        JR      L015D
```

```
;      Process the decay?
```

```
L012F    LD      A,(IX+08H)
        OR      A
        RET     Z
```

```

        PUSH    IX
        POP     HL
        LD      DE,0009H
        ADD     HL,DE
        CALL    L0190
        RET     NZ
        CALL    L01A6
        DEC     HL
        CALL    L0190
        JR      Z,L0161
        LD      A,(HL)
        AND     0F0H
        LD      E,A
        DEC     HL
        DEC     HL
        DEC     HL
        LD      A,(HL)
        AND     0F0H
        ADD     A,E
        LD      E,A
        LD      A,(HL)
        AND     0FH
        OR      E
        LD      (HL),A
L015D   OR      0FFH
        RET

D1AC3  ; DB '6'
L0161   LD      (HL),00H
        RET

;*****
;
; Put up the introduction screen
;
;*****
L1352   POP     HL                ; End of list, clean up

        LD      HL,Title_Msgs    ; Put up "COLECOVISION" title screen
        CALL    ScrnMsgs

        LD      HL,A143B         ; Point to color table
;
        LD      DE,0000H         ; Table index
        LD      D,B              ; (BC=0 from ScrnMsgs)
        LD      E,C
        LD      IY,0012H         ; Byte count
        CALL    _BlkWrtVRAM4     ; Color table
        LD      BC,01C0H         ; Unblank the screen
        CALL    _WriteReg
        LD      HL,Cart_Sig      ; Check first 2 bytes of cartridge
        LD      A,(HL)           ; If 8000<=AA or 8001<=55,
        CP      0AAH
        JR      NZ,L13B7         ; put up "please turn machine off
        INC     HL               ; before inserting cartridge" msg

```

```

L13B7      LD      A,(HL)
           CP      55H
           JP      NZ,L13FF

           LD      HL,Cart_Title    ; Find name string
           LD      D,H              ; Display address = first string
           LD      E,L
           CALL    L1946            ; Scan for first '/'
           PUSH    HL              ; Save address of first '/'
           LD      HL,0201H         ; Screen address for first string
           CALL    L1951            ; Display first string centered

           POP     HL              ; Recover address of first '/'
           INC     HL              ; Adjust for second string
           LD      D,H              ; Display address = second string
           LD      E,L
           CALL    L1946            ; Scan for second '/'
           PUSH    HL              ; Save address of second '/'
           LD      HL,01C1H         ; Screen address for second string
           CALL    L1951            ; Display second string centered

           POP     HL              ; Recover address of second '/'
           INC     HL              ; Adjust for third string
           LD      DE,02ACH         ; Screen address for copyright date
           LD      IY,0004H         ; 4 bytes
           CALL    _BlkWrtVRAM2

           CALL    A1968            ; "Respect the seven second delay we
                                   ; use" - Donald Fagen, "The Nightfly"

           LD      BC,0180H         ; Blank screen
           CALL    _WriteReg

           LD      HL,(Cart_Start) ; Get start address
           JP      (HL)             ; Jump into cartridge

DB 0,0,0,0,0,0,0,0,0,0
DB 0

```

```

;      Add A (signed byte) to (HL)

```

```

A01B1      LD      B,00H    ; sign-extend A -> BA
           BIT     7,A
           JR      Z,L01B9
           DEC     B
L01B9      ADD     A,(HL)
           LD      (HL),A
           INC     HL
           LD      A,(HL)
           ADC     A,B
           LD      (HL),A
           DEC     HL
           RET

```

DB 0

; Index into the 7020H array using B as the index
; B = 1 returns HL = (7020H)+2 and IX = (HL)

```
A01C1      LD      HL,(SndArray)    ; HL = (7020H)-2 + B*4
           DEC     HL
           DEC     HL
           LD      C,B
           LD      B,00H
           ADD     HL,BC
           ADD     HL,BC
           ADD     HL,BC
           ADD     HL,BC
           LD      E,(HL)          ; IX = (HL)
           INC     HL
           LD      D,(HL)
           PUSH    DE
           POP     IX
           RET
```

DB 0

; Update function code? (Used by Carnival!)

```
A01D5      LD      (IX+01H),L
           LD      (IX+02H),H
           LD      A,(DE)
           AND     3FH
           LD      B,A
           LD      A,(IX+00H)
           AND     0C0H
           OR      B
           LD      (IX+00H),A
           RET
```

; Get command list pointer?

```
L01E9      LD      A,(IX+00H)
           CP      0FFH
           RET     Z
           AND     3FH
           CP      3EH
           RET     NZ
           PUSH    IX
           POP     HL
           INC     HL
           LD      E,(HL)
           INC     HL
           LD      D,(HL)
           EX      DE,HL
           RET
```

DB 0,0,0,0,0,0

```

;*****
;      1FB2    PInitSound
;
; Parameter block version of 1FEE
;
; Parm 1 (byte) = number of array elements
; Parm 2 (word) = pointer to array of pointers
;
;*****
_PInitSound    LD      BC,P_InitSound
               LD      DE,ParmArea
               CALL    PCopy
               LD      A,(ParmArea)
               LD      B,A
               LD      HL,(ParmArea+1)

;*****
;      1FEE    InitSound
;
; Initializes 7020 with HL, clears out the array pointed to by
; (HL+2) to contain an FF every ten bytes for B times, then stores
; a zero at the end. Then the sound variables are initialized and
; the sound chip is silenced.
;
; Array at 7020H contains a list of four-byte entries. The
; second word of the list contains a pointer to a list of
; 10-byte blocks, and the second word of each other entry
; probably contains a pointer into this list as well. The
; first word of each four-byte entry is apparently a pointer
; to a list of command bytes.
;
; Each 10 byte block has the following format:
;
; +0    bits 0-5 = function code (3EH is special)
;        bits 6-7 = channel code (0=noise, else tone channel number)
; +1,+2 pointer to list of command bytes
; +3    low 8 bits of frequency
; +4    bits 0-3 = high 2 bits of frequency or noise control
;        bits 4-7 = amplitude (00H=max, 0FH=off)
; +5    duration counter?
; +6    bits 0-3 = twang time counter?
;        bits 4-7 = twang time?
; +7    twang offset? (signed byte)
; +8    bits 0-3 = decay high counter?
;        bits 4-7 = decay high nibble?
; +9    bits 0-3 = decay low counter?
;        bits 4-7 = decay low nibble?
;
; ENTRY B = number of array elements
;        HL = pointer to array of pointers
;*****
A0213
_InitSound     LD      (SndArray),HL
               INC     HL
               INC     HL

```

```

        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        EX      DE,HL
        LD      DE,000AH
NoSoundP EQU    $+1      ; null sound program = single FFH byte
L0220    LD      (HL),0FFH
        ADD     HL,DE
        DJNZ    L0220
        LD      (HL),D
        CALL    L0297
        LD      A,(HL)  ; = 0FFH
        LD      (NoiseCtlShad),A
        JR      A023B

```

```

DB 0,0,0,0,0,0,0,0,0,0
DB 0,0,0

```

```

;*****
;          1FD6    NoSound
;
; This routine silences the sound chip.
;*****

```

A023B

```

_NoSound    LD      A,9FH
            OUT      (IO_Sound),A
            LD      A,0BFH
            OUT      (IO_Sound),A
            LD      A,0DFH
            OUT      (IO_Sound),A
            LD      A,0FFH
            OUT      (IO_Sound),A
            RET

```

```

DB 0,0,0,0,0

```

```

;*****
;          1FB5    PAddSound
;
; Parameter block version of 1FF1
;
; Parm 1 (byte) = index into array at 7020H
;*****

```

```

_PAddSound  LD      BC,P_AddSound
            LD      DE,ParamArea
            CALL    PCopy
            LD      A,(ParamArea)
            LD      B,A

```

```

;*****
;          1FF1    AddSound
;
; This routine sets up a new entry in the sound list. (?)
;
; ENTRY B = index into array at 7020H, 1 = first entry

```

```
;*****
A025E
_AddSound      PUSH    BC
               CALL    A01C1
               LD      A,(IX+00H)
               AND     3FH
               POP     BC
               CP      B
               RET     Z
A026A          LD      (IX+00H),B
               DEC     HL
               DEC     HL
               LD      D,(HL)
               DEC     HL
               LD      E,(HL)
               LD      (IX+01H),E
               LD      (IX+02H),D
               CALL    L035F
               JR      L0295

DB 0,0

;*****
;      1FF4      UpdateSound
;
; This routine updates the current sound pointers
; by searching through the sound list.
;*****
A027F
_UpdateSound   LD      B,01H
               CALL    A01C1
L0284          XOR     A
               CP      (IX+00H)
               RET     Z
               CALL    L02D6
               LD      DE,000AH
               ADD     IX,DE
               JR      L0284

; Update stream pointers

L0295          CALL    L0297
               PUSH    IX

               LD      B,01H
               CALL    A01C1
L02AB          LD      A,(IX+00H)
               OR      A
               JR      Z,L02D3
               INC     A
               JR      Z,L02CB
               LD      A,(IX+00H)
               AND     0C0H
               RLCA
               RLCA
```

```

                                RLCA
                                LD     E,A
                                LD     D,00H
                                LD     HL,NoiseP
                                ADD    HL,DE
                                PUSH   IX
                                POP    DE
                                LD     (HL),E
                                INC    HL
                                LD     (HL),D
L02CB                          LD     DE,000AH
                                ADD    IX,DE
                                JR     L02AB

L02D3                          POP    IX
                                RET

L02D6                          CALL    L01E9
                                CP      0FFH
                                RET     Z
                                CP      3EH
                                JR     NZ,L02E6
                                LD     DE,0007H
                                ADD    HL,DE
                                JP     (HL)

L0297                          LD     HL,NoSoundP
                                LD     (NoiseP),HL
                                LD     (Tone1P),HL
                                LD     (Tone2P),HL
                                LD     (Tone3P),HL
                                RET

                                DB 0,0,0

L02E6                          CALL    L012F
                                CALL    L00FC
                                RET     NZ
A02EE                          LD     A,(IX+00H)      ; (called by Carnival)
                                PUSH   AF
                                CALL    L035F
                                POP    BC
                                LD     A,(IX+00H)
                                CP      B
                                RET     Z
                                JR     L0295

D1AC6                          DB      'TWO'

;*****
;      1F61    DoSound
;
; This updates the sound chip registers based on the
; current sound array pointers for each channel.
;*****
```

```
A0300
_DoSound    LD      A,9FH
            LD      C,90H
D0587      EQU      $+1
            LD      D,80H
            LD      IX,(Tone1P)
            CALL    L034E
            LD      A,0BFH
            LD      C,0B0H
            LD      D,0A0H
            LD      IX,(Tone2P)
            CALL    L034E
            LD      A,0DFH
            LD      C,0D0H
            LD      D,0C0H
            LD      IX,(Tone3P)
            CALL    L034E
            LD      A,0FFH
            LD      C,0F0H
            LD      IX,(NoiseP)
            LD      E,(IX+00H)
            INC     E
            JR      Z,L0335
            CALL    L0164
            LD      A,(IX+04H)
            AND     0FH
            LD      HL,NoiseCtlShad
            CP      (HL)
            RET     Z
            LD      (HL),A
            LD      C,0E0H
```

; Send amplitude or noise control to sound chip

```
L0164      LD      A,(IX+04H)
            BIT     4,C
            JR      Z,L016F
            RRCA
            RRCA
            RRCA
            RRCA
            RRCA
L016F      AND     0FH
            OR      C
L0335      OUT     (IO_Sound),A
            RET
L034E      LD      E,(IX+00H)
            INC     E
            JR      Z,L0335
            CALL    L0164
```

; Send frequency to sound chip

```
L0175      LD      A,(IX+03H)
            AND     0FH
```

```

    OR    D
    OUT   ( IO_Sound ),A
    LD    A,(IX+03H)
    AND   0F0H
    LD    D,A
    LD    A,(IX+04H)
    AND   0FH
    OR    D
    RRCA
    RRCA
    RRCA
    RRCA
    JR    L0335

L035F    LD    A,(IX+00H)
    AND   3FH
    PUSH  AF
    LD    (IX+00H),0FFH
    LD    L,(IX+01H)
    LD    H,(IX+02H)
    LD    A,(HL)
    LD    B,A
    BIT   5,A
    JR    Z,L0391
    PUSH  BC
    AND   1FH
    INC   HL
    LD    (IX+01H),L
    LD    (IX+02H),H
    LD    (IX+04H),0F0H
    LD    (IX+05H),A
    JR    L03F0

L0391    BIT   4,A
    JR    Z,L03A4
    BIT   3,A
    JR    Z,L039E
    POP   BC
    JP    _AddSound

L039E    LD    A,0FFH
    PUSH  AF
    JR    L03F8

L03A4    AND   3CH
    CP    04H
    JR    NZ,L03D2
    POP   IY
    PUSH  IY
    PUSH  BC
    INC   HL
    LD    E,(HL)
    LD    (IX+01H),E
    INC   HL
    LD    D,(HL)
```

	LD	(IX+02H),D	
	INC	HL	
	PUSH	IY	
	POP	AF	
	PUSH	DE	
	POP	IY	
	LD	DE,L03C6	
	PUSH	DE	
	JP	(IY)	
L03C6	LD	DE,0007H	
	ADD	IY,DE	
	LD	DE,L0461	
	PUSH	DE	
	JP	(IY)	
L03D2	PUSH	BC	
	LD	A,B	
	AND	03H	
	JR	NZ,L03FA	
	INC	HL	
	INC	HL	
	INC	HL	
	INC	HL	
	LD	(IX+01H),L	
	LD	(IX+02H),H	
	DEC	HL	
	LD	DE,0005H	
	CALL	L0478	; DE = DE + IX
	LD	BC,0003H	
	LDDR		
L03F0	LD	(IX+07H),00H	
L03F4	LD	(IX+08H),00H	
L03F8	JR	L0461	
L03FA	DEC	A	; CP 01H
	JR	NZ,L0419	
	LD	DE,0006H	
	ADD	HL,DE	
	LD	(IX+01H),L	
	LD	(IX+02H),H	
	DEC	HL	
	INC	E	
	CALL	L0478	; DE = DE + IX
	LD	BC,0005H	
	LDDR		
	JR	L03F4	
L0419	DEC	A	; CP 02H
	JR	NZ,L0445	
	LD	DE,0006H	
	ADD	HL,DE	
	POP	AF	
	PUSH	AF	
	AND	0C0H	

```

    JR      NZ,L0429
    DEC     HL
L0429    LD      (IX+01H),L
    LD      (IX+02H),H
    DEC     HL
    LD      E,09H
    CALL    L0478      ; DE = DE + IX
    LD      BC,0002H
    LDDR
    XOR     A
    LD      (DE),A
    DEC     DE
    DEC     DE
    LD      C,03H
    LDDR
    JR      L0461

L0445    LD      DE,0008H
    ADD     HL,DE
    LD      (IX+01H),L
    LD      (IX+02H),H
    DEC     HL
    INC     E
    CALL    L0478      ; DE = DE + IX
    LD      BC,0007H
    LDDR

L0461    PUSH    IX
    POP     HL
    POP     AF
    POP     BC
    CP      0FFH
    RET     Z
    LD      D,A
    AND     3FH
    CP      04H
    JR      NZ,L0472
    LD      B,3EH
L0472    LD      A,D
    AND     0C0H
    OR      B
    LD      (HL),A
    RET
```

```

;*****
;
; VIDEO ROUTINES BEGIN HERE
;
;*****
```

```

;*****
;      1F64
;
; Parameter block version of 1FF7
;
; Parm 1 (lit) = ?
```

```
; Parm 2 (byte) = ?
;*****
L0488      LD      BC,D0482
           LD      DE,ParmArea
           CALL   PCopy
           LD      HL,(ParmArea)
           LD      E,(HL)
           INC     HL
           LD      D,(HL)
           EX      DE,HL
           LD      A,(ParmArea+2)
           OR      A          ; (clears carry)
           JR      Z,L04A3
           SCF

;*****
;      1FF7
;
; ENTRY HL = ?
;      C-flag = ?
;*****
L04A3      LD      E,(HL)
           INC     HL
           LD      D,(HL)
           INC     HL
           LD      C,(HL)
           INC     HL
           LD      B,(HL)
           INC     HL
           LD      A,00H      ; note: must preserve carry here!
           LD      (BC),A
           LD      A,(DE)
           PUSH    AF
           AND     0FH
           JR      Z,L04E7
           DEC     A
           JR      Z,L05F1
           DEC     A
           JR      Z,L0600
           DEC     A
           JR      Z,L0600
           DEC     A
           JR      NZ,L04E5

; 04A3/1FF7 Function 4

L04C6      LD      A,(DE)
           RRA
           RRA
           RRA
           RRA
           AND     0FH
           LD      B,A
           LD      E,(HL)
           INC     HL
```

```
LD      D,(HL)
INC     HL
OR      A
JR      Z,L04E5
L04D5   POP     AF
        PUSH    AF
        PUSH    HL
        PUSH    BC
        EX      DE,HL
        CALL    L04A3
        POP     BC
        POP     HL
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        INC     HL
        DJNZ    L04D5
L04E5   POP     AF
        RET
```

; 04A3/1FF7 Function 1

```
L05F1   CALL    L0572
        INC     DE
        LD      A,(DE)
        LD      (IY+05H),A
        INC     DE
        LD      A,(DE)
        LD      (IY+06H),A
        POP     AF
        RET
```

; 04A3/1FF7 Functions 2 & 3

```
L0600   INC     BC
        INC     BC
        INC     BC
        INC     BC
        INC     BC
        EX      DE,HL
        INC     HL
        LD      A,(HL)
        LD      E,A
        LD      D,00H
        PUSH    DE
        INC     HL
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        INC     HL
        ADD     A,(HL)
        LD      (BC),A
        LD      C,(HL)
        LD      B,00H
        PUSH    BC
```

```

POP      IY
EX       DE,HL
POP      DE
POP      AF
RET      NC
LD       A,01H      ; Sprite pattern gen table
JP       _BlkWrtVRAM

```

```
; 04A3/1FF7 Function 0
```

```

L04E7    CALL    L0572
LD       A,(DE)
LD       L,A
INC      DE
LD       A,(DE)
ADD      A,L
LD       (IY+05H),A
LD       H,00H
POP      AF
RET      NC          ; this is the carry flag from L04A3
PUSH     AF
LD       A,(VDP0Shad) ; Check for hi-res graphics
BIT      1,A
JR       Z,L0530     ; Branch if not
EX       DE,HL
LD       B,H
LD       C,L
LD       L,(HL)
LD       H,00H
PUSH     HL
ADD      HL,HL
ADD      HL,HL
ADD      HL,HL
PUSH     HL
INC      BC
LD       A,(BC)
LD       L,A
INC      BC
LD       A,(BC)
LD       H,A
POP      BC
POP      IY
POP      AF
BIT      7,A
CALL     NZ,L0594
INC      D           ; DE = DE + 0100H
BIT      6,A
CALL     NZ,L0594
INC      D           ; DE = DE + 0100H
BIT      5,A
RET      Z
L0594    PUSH     AF
PUSH     BC
PUSH     IY
PUSH     DE

```

```

PUSH    HL
CALL    _BlkWrtVRAM3    ; Pattern generator table
POP     HL
POP     DE
POP     IY
POP     BC
POP     AF
PUSH    AF
PUSH    BC
PUSH    IY
PUSH    DE
PUSH    HL
BIT     4,A
JR      NZ,L05BC
ADD     HL,BC
CALL    _BlkWrtVRAM4    ; Color table
L05B5   POP     HL
        POP     DE
        POP     IY
        POP     BC
        POP     AF
        RET

L0530   EX      DE,HL
        LD      C,(HL)
        LD      B,00H
        PUSH    BC
        POP     IY
        INC     HL
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        PUSH    HL
        PUSH    BC
        PUSH    DE
        PUSH    IY
        CALL    _BlkWrtVRAM3    ; Pattern generator table
        POP     BC
        POP     HL
        LD      E,L
        LD      D,H
        ADD     HL,BC
        DEC     HL
        SRL     H
        RR      L
        SRL     H
        RR      L
        SRL     H
        RR      L
        SRA     E
        SRA     E
        SRA     E
        OR      A
        SBC     HL,DE
```

```
INC     HL
PUSH    HL
POP     IY
POP     HL
ADD     HL,HL
ADD     HL,HL
ADD     HL,HL
POP     BC
ADD     HL,BC
CALL    _BlkWrtVRAM4    ; Color table
POP     AF
RET

L05BC   ADD     HL,BC
        LD      C,L
        LD      B,H
        PUSH    IY
        POP     HL
L05C2   PUSH    HL
        LD      A,(BC)
        PUSH    BC
        LD      BC,0008H
        LD      HL,(VDP_Temp)
        ADD     HL,BC
        LD      B,C
L05CE   DEC     HL
        LD      (HL),A
        DJNZ    L05CE
        PUSH    DE
        CALL    L1EA5    ; LD A,04H LD IY,0001H JP _BlkWrtVRAM
        POP     DE
        POP     BC
        INC     DE
        INC     BC
        POP     HL
        DEC     HL
        LD      A,H
        OR      L
        JR      NZ,L05C2
        JR      L05B5

L0572   PUSH    BC
        POP     IY
        PUSH    DE
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        BIT     7,D
        JR      NZ,L0591
        LD      A,D
        CP      70H
        JR      C,L0588
        LD      A,80H
        LD      (DE),A
        JR      L0591
```

```
L0588      LD      HL,D0587
           LD      BC,0001H
           CALL    _WrtVRAM
L0591      POP     DE
           INC     DE
           RET

;*****
;      1F67
;
; Parameter block version of 1FFA
;
; Parm 1 (word) = ???
; Parm 2 (byte) = ???
;*****
L06C7      LD      BC,D06C1
           LD      DE,ParamArea
           CALL    PCopy
           LD      IX,(ParamArea)
           LD      A,(ParamArea+2)
           LD      B,A

;*****
;      1FFA
;
; ENTRY IX = ???
;      A = ???
;*****
L06D8      LD      A,(D73C6)
           DEC     A
           JR      NZ,L06E3

L0623      PUSH    IX
           LD      HL,(73CDH)
           POP     DE
           LD      (HL),E
           INC     HL
           LD      (HL),D
           INC     HL
           LD      (HL),B
           INC     HL
           EX      DE,HL
           LD      A,(73CBH)
           INC     A
           LD      HL,73CAH
           CP      (HL)
           EX      DE,HL
           JR      NZ,L0669
           XOR     A
           LD      HL,(73D1H)
L0669      LD      (73CDH),HL
           LD      (73CBH),A
           RET
```

L06E3 LD L,(IX+00H)
 LD H,(IX+01H)
 LD A,(HL)
 LD C,A
 AND 0FH
 JR Z,L06FF
 DEC A
 JP Z,L0A87
 DEC A
 JR Z,L08DF
 DEC A
 JP Z,L0955
 JP L0EA2

L08DF LD E,0F9H
 CALL L08E0
 LD HL,0008H
 JP L0999

L06FF LD E,(IX+02H)
 LD D,(IX+03H)
 PUSH DE
 POP IY
 LD E,(IY+01H)
 LD D,(IY+02H)
 CALL A07E8
 LD C,E
 LD E,(IY+03H)
 LD D,(IY+04H)
 CALL A07E8
 LD B,E
 LD E,(IY+00H)
 LD D,00H
 ADD HL,DE
 ADD HL,DE
 LD E,05H
 ADD HL,DE
 LD E,(HL)
 INC HL
 LD D,(HL)
 EX DE,HL
 PUSH BC
 POP DE
 LD C,(HL)
 INC HL
 LD B,(HL)
 INC HL
 LD A,(IX+05H)
 BIT 7,A
 JP NZ,A080B
 PUSH BC
 PUSH DE
 PUSH HL
 CP 70H
 JR Z,L0744

L0744	JR	C,L074B	
	LD	H,A	
	LD	L,(IX+04H)	
	JR	L0780	
L074B	LD	HL,(VDP_Temp)	
	LD	E,(IX+04H)	
	LD	D,(IX+05H)	
	PUSH	HL	
	PUSH	DE	
	PUSH	HL	
	LD	BC,0004H	
	CALL	_ReadVRAM	
	POP	HL	
	LD	A,(HL)	
	CP	80H	
	JR	NZ,L0766	
	POP	DE	
	JR	L077F	
L0766	INC	HL	
	INC	HL	
	LD	B,(HL)	
	INC	HL	
	LD	E,(HL)	
	LD	D,00H	
	INC	HL	
	EX	DE,HL	
	DB	0FEH	; CP n - skips the ADD HL,HL
L0771	ADD	HL,HL	
	DJNZ	L0771	
	PUSH	HL	
	POP	BC	
	EX	DE,HL	
	POP	DE	
	INC	DE	
	INC	DE	
	INC	DE	
	INC	DE	
	CALL	_ReadVRAM	
L077F	POP	HL	
L0780	LD	A,(HL)	
	CP	80H	
	JR	Z,L0794	
	LD	E,(HL)	
	INC	HL	
	LD	D,(HL)	
	INC	HL	
	LD	C,(HL)	
	INC	HL	
	LD	B,(HL)	
	INC	HL	
	PUSH	IX	
	CALL	A080B	
	POP	IX	

```
L0794      POP      HL
           POP      DE
           POP      BC
           PUSH     BC
           PUSH     DE
           PUSH     HL
           LD        L,(IX+04H)
           LD        H,(IX+05H)
           LD        A,70H
           CP        H
           JR        C,L07A8
           LD        HL,(VDP_Temp)
L07A8      LD        (HL),E
           INC       HL
           LD        (HL),D
           INC       HL
           LD        (HL),C
           INC       HL
           LD        (HL),B
           INC       HL
           PUSH     IX
           CALL      L0898
           POP      IX
           POP      HL
           POP      DE
           POP      BC
           PUSH     IX
           CALL      A080B
           POP      IX
           LD        D,(IX+05H)
           LD        A,70H
           CP        D
           RET       Z
           RET       C
           LD        E,(IX+04H)
           EXX
           LD        HL,(VDP_Temp)
           PUSH     HL
           INC       HL
           INC       HL
           LD        E,(HL)
           LD        D,00H
           INC       HL
           LD        B,(HL)
           EX        DE,HL
           DB        0FEH          ; CP n - skips the ADD HL,HL
L07DD      ADD       HL,HL
           DJNZ      L07DD
           PUSH     HL
           EXX
           POP      BC
           POP      HL
           JP        _WrtVRAM
L08E0      LD        IY,(VDP_Temp)
```

	LD	L, (IX+02H)	
	LD	H, (IX+03H)	
	INC	HL	
	LD	C, (HL)	
	INC	HL	
	LD	B, (HL)	
	LD	A, B	
	OR	A	
	JR	Z, L0900	
	INC	A	
	JR	NZ, L0A54	
	LD	A, C	
	CP	E	
	JP	M, L0A54	
L0900	INC	HL	
	LD	C, (HL)	
	INC	HL	
	LD	B, (HL)	
	LD	A, B	
	OR	A	
	JR	Z, L0914	
	INC	A	
	JR	NZ, L0A54	
	LD	A, C	
	CP	E	
	JP	M, L0A54	
L0914	DEC	HL	
	DEC	HL	
	LD	A, (HL)	
	OR	A	
	JR	Z, L09CA	
	DEC	HL	
	LD	C, (HL)	
	INC	HL	
	LD	B, (HL)	
	RET		
L09CA	POP	HL	; get rid of return address
	LD	L, (IX+02H)	
	LD	H, (IX+03H)	
	INC	HL	
	LD	A, (HL)	
	CALL	L09D5	
	LD	A, (HL)	
L09FD	LD	(IY+03H), A	
L0A00	LD	L, (IX+02H)	
	LD	H, (IX+03H)	
	INC	HL	
	INC	HL	
	INC	HL	
	LD	A, (HL)	
	LD	(IY+00H), A	
	CALL	L09D8	
	INC	HL	
	LD	A, (HL)	

```

LD      L,(IX+00H)
LD      H,(IX+01H)
INC     HL
ADD     A,(HL)
LD      (IY+02H),A
XOR     A                ; Sprite attribute table
LD      D,A
LD      E,(IX+04H)
PUSH    IY
JR      L0A7E

L0A54    POP     DE                ; get rid of return address
PUSH    IY
PUSH    IX
PUSH    IY
PUSH    IY
XOR     A                ; Sprite attribute table
LD      D,A
LD      E,(IX+04H)
POP     HL
CALL    L1E92            ; LD IY,0001H / JP _BlkReadVRAM
XOR     A
POP     IY
LD      (IY+01H),A
LD      A,80H
LD      (IY+03H),A
XOR     A                ; Sprite attribute table
LD      D,A
POP     IX
LD      E,(IX+04H)

L0A7E    POP     HL
JP      L1EA7            ; LD IY,0001H / JP _BlkWrtVRAM

;*****
;
; Initialize the video chip
;
;*****
L1319    CALL    InitScrn

;          Initialize the character cells for "COLECOVISION"

LD      HL,A18A3        ; Point to character cell usage list
LD      DE,0060H        ; Starting character code = 60H
L1330    PUSH    HL
LD      A,(HL)          ; Get block ID
RLCA                ; Multiply index by 8
JP      C,L1352        ; Check for end of list
RLCA
RLCA
LD      C,A            ; Load BC with block offset
LD      B,D            ; (D-reg = 00H here)
LD      HL,A14C3        ; Point to block data
L1356    ADD     HL,BC    ; Point HL to block image
PUSH    DE

```

```

CALL    _BlkWrtVRAM31    ; LD A,03H LD IY,0001H JP _BlkWrtVRAM
POP     DE
POP     HL
INC     DE
INC     HL
JR      L1330

InitScrn    XOR     A            ; Fill VRAM from address 0000H
            LD      H,A          ;   with 00H
            LD      L,A          ;
            LD      DE,4000H     ;   length 4000H
            CALL    _FillVRAM   ; Do the fill
            CALL    _InitVDP    ; Initialize the video chip
            JP      _InitFont   ; Initialize the text font

DB 0,0,0,0,0,0,0,0,0,0

A07E8      PUSH    HL
            SRA     D
            RR      E
            SRA     D
            RR      E
            SRA     D
            RR      E
            BIT     7,D
            LD      HL,0FF80H
            JR      NZ,L0802
            ADD     HL,DE
            POP     HL
            RET     NC
            LD      E,7FH
            RET

L0802      LD      H,00H        ; LD HL,0080H
            ADD     HL,DE
            POP     HL
            RET     C
            LD      E,80H
            RET

DB 0

A080B      PUSH    BC          ; (called by Antarctic Adventure and Destructor)
            PUSH    DE
            PUSH    HL
            EXX
            POP     HL
            POP     DE
            POP     BC
            CALL    A08C0
            EXX
            LD      A,E
            BIT     7,A
            JR      NZ,L081E
            CP      20H

```

	RET	NC
L081E	ADD	A,C
	BIT	7,A
	RET	NZ
	OR	A
	RET	Z
	BIT	7,E
	JR	Z,L0848
	LD	A,C
	ADD	A,E
	PUSH	DE
	CP	21H
	JR	C,L0831
	LD	A,20H
L0831	LD	E,A
	LD	D,00H
	PUSH	DE
	POP	IY
	POP	DE
	LD	A,E
	EXX	
	PUSH	BC
	NEG	
	LD	C,A
	LD	B,00H
	ADD	HL,BC
	EX	DE,HL
	ADD	HL,BC
	EX	DE,HL
	POP	BC
	EXX	
	JR	L0864
L0848	LD	A,E
	ADD	A,C
	CP	1FH
	JR	Z,L085D
	JR	C,L085D
	LD	A,20H
	SUB	E
	PUSH	DE
	LD	E,A
	LD	D,00H
	PUSH	DE
	POP	IY
	POP	DE
	JR	L0864
L085D	PUSH	BC
	LD	B,00H
	PUSH	BC
	POP	IY
	POP	BC
L0864	LD	E,00H
L0866	LD	A,D

```
ADD    A,E
BIT    7,A
JR     NZ,L0885
CP     18H
JR     NC,L0885
PUSH   BC
PUSH   DE
EXX
PUSH   BC
PUSH   DE
PUSH   HL
PUSH   IY
CALL   _BlkWrtVRAM2
POP    IY
POP    HL
POP    DE
POP    BC
EXX
POP    DE
POP    BC
EXX
PUSH   BC
LD     B,00H
ADD    HL,BC
EX     DE,HL
LD     C,20H
ADD    HL,BC
EX     DE,HL
POP    BC
EXX
INC     E
LD     A,E
CP     B
JR     NZ,L0866
RET

L0885
CALL   A08C0
PUSH   BC
LD     B,00H
PUSH   BC
POP    IY
POP    BC
PUSH   BC
PUSH   DE
PUSH   HL
PUSH   IY
LD     A,02H           ; Name table
CALL   _BlkReadVRAM
POP    IY
POP    HL
POP    DE
POP    BC
PUSH   BC
LD     B,00H
ADD    HL,BC

L0898
CALL   A08C0
PUSH   BC
LD     B,00H
PUSH   BC
POP    IY
POP    BC
PUSH   BC
PUSH   DE
PUSH   HL
PUSH   IY
LD     A,02H           ; Name table
CALL   _BlkReadVRAM
POP    IY
POP    HL
POP    DE
POP    BC
PUSH   BC
LD     B,00H
ADD    HL,BC

L08A2
```

```

        LD      C,20H
        EX      DE,HL
        ADD     HL,BC
        EX      DE,HL
        POP     BC
        DEC     B
        JR      NZ,L08A2
        RET

        DB 0,0,0,0

A08C0    PUSH    HL
        LD      H,00H    ; sign-extend D -> HL
        BIT     7,D
        JR      Z,L08CB
        DEC     H
L08CB    LD      L,D
        ADD     HL,HL    ; HL = D * 32 (signed multiply)
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL
        LD      D,00H    ; sign-extend E -> DE
        BIT     7,E
        JR      Z,L08DB
        DEC     D
L08DB    JP      L047C    ; DE = (D * 32) + E (signed arithmetic)

L0955    LD      E,0E1H
        CALL    L08E0
        LD      HL,0020H
L0999    ADD     HL,BC
        LD      A,L
        CALL    L09D5
        LD      A,(HL)
        OR      80H
        JP      L09FD

L0A87    LD      IY,(VDP_Temp)
        LD      A,(VDP0Shad)    ; Check for hi-res graphics
        BIT     1,A
        SET     7,B
        JR      NZ,L0A98    ; Branch if so
        RES     7,B
L0A98    LD      (IY+03H),B
        PUSH    HL
        LD      L,(IX+02H)
        LD      H,(IX+03H)
        LD      A,(HL)
        LD      (IY+04H),A
        XOR     80H
        LD      (HL),A
        INC     HL
        LD      E,(HL)
        LD      A,E

```

```

AND      07H
NEG
ADD      A,08H
LD       (IY+01H),A
INC      HL
LD       D,(HL)
CALL     A07E8
LD       (IY+11H),E
INC      HL
LD       E,(HL)
LD       A,E
AND      07H
LD       (IY+00H),A
INC      HL
LD       D,(HL)
CALL     A07E8
LD       (IY+12H),E
LD       HL,(VDP_Temp)
LD       DE,0013H
ADD      HL,DE
LD       E,(IY+11H)
LD       D,(IY+12H)
LD       BC,0303H
CALL     L0898
LD       E,(IX+04H)
LD       D,(IX+05H)
LD       A,(IX+06H)
POP      IX
LD       IY,(VDP_Temp)
LD       (IY+05H),A
PUSH     DE
LD       HL,(VDP_Temp)
LD       BC,0006H
ADD      HL,BC
LD       C,0BH
LD       A,D
CP       70H
JR       NC,L0B07
CALL     _ReadVRAM
JR       L0B0A

L0B07    EX      DE,HL
         LDIR
L0B0A    LD       HL,(VDP_Temp)
         PUSH    HL
         LD       DE,0013H
         ADD     HL,DE
         EXX
         POP     DE
         LD       HL,0008H
         ADD     HL,DE
         EX      DE,HL
         EXX
         LD       IY,(VDP_Temp)
         LD       C,(IY+05H)

```

```
L0B25      LD      B,09H
           LD      A,(HL)
           SUB     C
           CP      12H
           JR      NC,L0B39
           CP      09H
           JR      C,L0B31
           SUB     09H
L0B31      EXX
           LD      L,A
           LD      H,00H
           ADD     HL,DE
           LD      A,(HL)
           EXX
           LD      (HL),A
L0B39      INC     HL
           DJNZ    L0B25
           POP     DE
           LD      HL,(VDP_Temp)
           PUSH    HL
           LD      BC,0011H
           ADD     HL,BC
           LD      C,0BH
           LD      A,D
           CP      70H
           JR      NC,L0B51
           CALL    _WrtVRAM
           DB      11H                ; LD DE,n - skips LDIR instruction
L0B51      LDIR
           POP     DE
           PUSH    IX
           LD      HL,0013H
           ADD     HL,DE
           EX      DE,HL
           LD      BC,0014H
           ADD     HL,BC
           LD      B,09H
L0B64      LD      A,(DE)
           INC     DE
           PUSH    DE
           LD      DE,0008H
           ADD     HL,DE
           PUSH    HL
           LD      E,A
           LD      D,00H
           LD      C,A
           PUSH    BC
           LD      A,09H
           SUB     B
           LD      B,D
L0B76      SUB     03H
           JR      C,L0B7D
           INC     B
           JR      L0B76
```

```

L0B7D      LD      A,B
           LD      IY,(VDP_Temp)
           ADD     A,(IY+12H)
           BIT     7,(IY+03H)
           JR      NZ,L0BB8
           LD      A,03H          ; Pattern generator table
           CALL    L1E92          ; LD IY,0001H / JP _BlkReadVRAM
           POP     BC
           LD      HL,(VDP_Temp)
           PUSH    BC
           LD      DE,0084H
           ADD     HL,DE
           LD      E,C
           SRL     E
           SRL     E
           SRL     E
;           LD      D,00H
           LD      A,09H
           SUB     B
           LD      C,A
           LD      B,D
           ADD     HL,BC
           JR      L0BD6

L0BB8      SRA     A
           SRA     A
           SRA     A
           CP      03H
           JR      NC,L0BD9
           LD      D,A
           PUSH    DE
           PUSH    HL
           LD      A,03H          ; Pattern generator table
           CALL    L1E92          ; LD IY,0001H / JP _BlkReadVRAM
           POP     HL
           LD      DE,0068H
           ADD     HL,DE
           POP     DE
L0BD6      CALL    L1E90          ; LD A,04H LD IY,0001H JP _BlkReadVRAM
L0BD9      POP     BC
           POP     HL
           POP     DE
           DJNZ    L0B64
           POP     IX
           EXX
           LD      E,(IX+02H)
           LD      D,(IX+03H)
           LD      C,(IX+04H)
           LD      B,(IX+05H)
           EXX
           PUSH    IX
           POP     HL
           LD      IY,(VDP_Temp)
           LD      A,(IY+04H)
           ADD     A,A

```

	LD	C,A
	LD	DE,0006H
	LD	B,D
	ADD	HL,DE
	ADD	HL,BC
	LD	E,(HL)
	INC	HL
	LD	D,(HL)
	LD	HL,(VDP_Temp)
	LD	BC,007CH
	ADD	HL,BC
	PUSH	HL
	PUSH	BC
	LD	C,05H
	LD	A,D
	CP	70H
	JR	NC,L0C1A
	CALL	_ReadVRAM
	JR	L0C1D
L0C1A	EX	DE,HL
	LDIR	
L0C1D	LD	IY,(VDP_Temp)
	POP	BC
	ADD	IY,BC
	LD	A,(IY+04H)
	LD	IY,(VDP_Temp)
	LD	(IY+02H),A
	POP	DE
	LD	HL,(VDP_Temp)
	LD	BC,0064H
	ADD	HL,BC
	LD	B,04H
L0C38	LD	A,(DE)
	CP	(IX+01H)
	PUSH	DE
	EXX	
	LD	H,00H
	JR	NC,L0C55
	ADD	A,A
	ADD	A,A
	ADD	A,A
	LD	L,A
	ADD	HL,BC
	PUSH	HL
	EXX	
	POP	DE
L0C7C	EX	DE,HL
	PUSH	BC
	LD	BC,0008H
	LDIR	
	EX	DE,HL
	JR	L0C84
L0C55	SUB	(IX+01H)

```
ADD    A,A
ADD    A,A
ADD    A,A
LD     L,A
ADD    HL,DE
PUSH   HL
EXX
POP    DE
LD     A,D
CP     70H
JR     NC,L0C7C
PUSH   BC
PUSH   HL
PUSH   DE
LD     BC,0008H
PUSH   BC
CALL   _ReadVRAM
POP    BC
POP    HL
ADD    HL,BC
EX     DE,HL
POP    HL
ADD    HL,BC
POP    BC
POP    DE
INC    DE
DJNZ   L0C38
LD     IY,(VDP_Temp)
LD     DE,(VDP_Temp)
LD     HL,001CH
ADD    HL,DE
LD     C,(IY+00H)
LD     B,00H
ADD    HL,BC
PUSH   HL
POP    IX
LD     HL,0064H
ADD    HL,DE
PUSH   HL
LD     A,10H
EX     AF,AF'
POP    HL
LD     D,(HL)
INC    HL
PUSH   HL
LD     BC,000FH
ADD    HL,BC
LD     E,(HL)
EX     DE,HL
LD     B,(IY+01H)
XOR    A
DEC    B                ; cheap HL=HL<<B?
JP     M,L0CBC
ADD    HL,HL
RLA
```

```

JR      L0CB4

L0CBC   LD      E,A
        CALL    L0E2F
        LD      A,(IY+00H)
        INC     A
        LD      (IY+00H),A
        CP      08H
        JR      Z,L0CCF
        CP      10H
        JR      NZ,L0CD4
L0CCF   LD      BC,0010H
        ADD     IX,BC
L0CD4   INC     IX
        EX      AF,AF'
        DEC     A
        JR      NZ,L0CA5
L0CDD   POP     HL
        BIT     7,(IY+03H)
        JR      NZ,L0D03
        LD      HL,(VDP_Temp)
        LD      BC,0084H
        ADD     HL,BC
        LD      D,(IY+02H)
        BIT     1,(IY+03H)
        LD      BC,0900H
        JR      NZ,L0CFC
L0CFC   LD      C,0FH
        LD      A,(HL)
        AND     C
        OR      D
        LD      (HL),A
        INC     HL
        DJNZ    L0CFC
L0D03   LD      A,(IY+05H)
        BIT     7,(IY+04H)
        JR      Z,L0D0E
        ADD     A,09H
L0D0E   LD      C,A
        LD      HL,(VDP_Temp)
        LD      DE,0013H
        ADD     HL,DE
        LD      B,09H
L0D18   LD      (HL),A
        INC     A
        INC     HL
        DJNZ    L0D18
        BIT     7,(IY+03H)
        JR      NZ,L0D96
        LD      E,C
        LD      HL,(VDP_Temp)
        PUSH    HL
        LD      BC,001CH
        LD      D,B
        ADD     HL,BC

```

```

LD      IY,0009H
CALL    _BlkWrtVRAM3      ; Pattern generator table
LD      IY,(VDP_Temp)
POP     HL
LD      BC,0084H
ADD     HL,BC
LD      IX,(VDP_Temp)
LD      C,13H
ADD     IX,BC
LD      B,09H
L0D4C   LD      A,(IX+00H)
INC     IX
SRL     A
SRL     A
SRL     A
LD      E,A
LD      D,00H
PUSH    BC
LD      A,09H
SUB     B
LD      B,D
L0D60   CP      03H
JR      C,L0D69
SUB     03H
INC     B
JR      L0D60

L0D69   ADD     A,(IY+11H)
CP      20H
JR      NC,L0D87
LD      A,B
ADD     A,(IY+12H)
CP      18H
JR      NC,L0D87
PUSH    IX
PUSH    HL
CALL    L1EA5              ; LD A,04H LD IY,0001H JP _BlkWrtVRAM
POP     HL
POP     IX
L0D87   POP     BC
INC     HL
INC     DE
DEC     B
LD      A,B
OR      A
JR      NZ,L0D4C
JR      L0DEA

L0D96   ;      LD      B,00H              ; B=00 already from L0D18 loop
L0D98   PUSH    BC
LD      A,C              ; C=C+B*3
ADD     A,B
ADD     A,B
ADD     A,B
LD      C,A

```

```

LD      DE,0018H      ; cheap multiply by 18H?
LD      H,D           ; HL=B*18H? (could be optimized?)
LD      L,D
LD      A,B
L0DA5   DEC      A
        JP      M,L0DAC
        ADD     HL,DE
        JR      L0DA5

L0DAC   LD      IY,(VDP_Temp)
        LD      A,(IY+12H)
        ADD     A,B
        CP      18H
        JR      NC,L0DE3
        SRL     A
        SRL     A
        SRL     A
        LD      D,A
        LD      E,C
        PUSH    DE
        LD      BC,001CH
        ADD     HL,BC
        LD      BC,(VDP_Temp)
        ADD     HL,BC
        PUSH    HL
        LD      IY,0003H
        CALL    _BlkWrtVRAM3      ; Pattern generator table
        POP     HL
        LD      DE,0068H
        ADD     HL,DE
        POP     DE
        LD      IY,0003H
        CALL    _BlkWrtVRAM4
L0DE3   POP     BC
        INC     B
        LD      A,B
        CP      03H
        JR      NZ,L0D98
L0DEA   LD      IY,(VDP_Temp)
        LD      B,(IY+06H)
        LD      A,B
        CP      80H
        JR      Z,L0E17
        LD      C,(IY+07H)
        LD      H,(IY+11H)
        LD      L,(IY+12H)
        OR      A
        SBC     HL,BC
        JR      Z,L0E17
        LD      HL,(VDP_Temp)
        LD      DE,0008H
        ADD     HL,DE
        LD      E,(IY+06H)
        LD      D,(IY+07H)
        CALL    L0E28

```

L0E17	LD	IY, (VDP_Temp)
	LD	HL, (VDP_Temp)
	LD	DE, 0013H
	ADD	HL, DE
	LD	E, (IY+11H)
	LD	D, (IY+12H)
L0E28	LD	BC, 0303H
	JP	A080B
L0E2F	BIT	0, (IY+03H)
	JR	NZ, L0E4B
	OR	(IX+00H)
	LD	(IX+00H), A
	LD	A, H
	OR	(IX+08H)
	LD	(IX+08H), A
	LD	A, L
	OR	(IX+10H)
	JR	L0E5C
L0E4B	OR	A
	JR	Z, L0E51
	LD	(IX+00H), A
L0E51	LD	A, H
	OR	A
	JR	Z, L0E58
	LD	(IX+08H), A
L0E58	LD	A, L
	OR	A
	JR	Z, L0E5F
L0E5C	LD	(IX+10H), A
L0E5F	BIT	7, (IY+03H)
	RET	Z
	PUSH	IX
	LD	BC, 0068H
	ADD	IX, BC
	LD	C, B
	LD	B, (IY+02H)
	BIT	1, (IY+03H)
	JR	NZ, L0E7B
	LD	C, 0FH
L0E7B	LD	A, E
	OR	A
	JR	Z, L0E87
	LD	A, (IX+00H)
	AND	C
	OR	B
	LD	(IX+00H), A
L0E87	LD	A, H
	OR	A
	JR	Z, L0E93
	LD	A, (IX+08H)
	AND	C
	OR	B
	LD	(IX+08H), A

L0E93	LD	A,L
	OR	A
	JR	Z,L0E9F
	LD	A,(IX+10H)
	AND	C
	OR	B
	LD	(IX+10H),A
L0E9F	POP	IX
	RET	
L0EA2	PUSH	BC
	EXX	
	LD	L,(IX+02H)
	LD	H,(IX+03H)
	LD	A,(HL)
	INC	HL
	LD	C,(HL)
	INC	HL
	LD	B,(HL)
	INC	HL
	LD	E,(HL)
	INC	HL
	LD	D,(HL)
	EXX	
	ADD	A,A
	ADD	A,A
	LD	E,A
	LD	D,00H
	ADD	HL,DE
	INC	HL
	LD	C,(HL)
	INC	HL
	LD	B,(HL)
	INC	HL
	LD	E,(HL)
	INC	HL
	LD	D,(HL)
	LD	H,B
	LD	L,C
	POP	BC
	LD	A,C
	LD	C,B
	SRL	A
	SRL	A
	SRL	A
	SRL	A
	LD	B,A
	PUSH	BC
	PUSH	IX
L0ED3	PUSH	HL
	PUSH	DE
	LD	L,(IX+04H)
	LD	H,(IX+05H)
	INC	IX
	INC	IX

```
INC    HL
INC    HL
LD     E,(HL)
INC    HL
LD     D,(HL)
PUSH   DE
POP    IY
POP    DE
POP    HL
LD     A,(HL)
BIT    7,(IY+00H)
JR     Z,L0EF2
SET    7,A
LD     (IY+00H),A
INC    HL
LD     A,(DE)
EXX
LD     L,A
LD     H,00H
ADD    HL,BC
LD     (IY+01H),L
LD     (IY+02H),H
EXX
INC    DE
LD     A,(DE)
EXX
LD     L,A
LD     H,00H
ADD    HL,DE
LD     (IY+03H),L
LD     (IY+04H),H
EXX
INC    DE
DJNZ   L0ED3
POP    IY
LD     BC,0004H
ADD    IY,BC
POP    DE
LD     L,(IY+00H)
LD     H,(IY+01H)
INC    IY
INC    IY
PUSH   HL
POP    IX
PUSH   IY
PUSH   DE
LD     B,E
CALL   L06D8
POP    DE
POP    IY
DEC    D
JR     NZ,L0F1C
RET
```

L0EF2

L0F1C

DB 0,0,0,0,0,0,0

[illegible]

[illegible]

```

;*****
;
;
; CONTROLLER HANDLING ROUTINES BEGIN HERE
;
;*****
;*****
;
; This routine initializes the controller state tables,
; counters, and shadow bytes.
;
;*****
A1105

```

```

InitCtlState  OUT      (IO_Joy_Select),A ; Select joysticks
               XOR      A                ; Prepare to clear some memory
               LD        IX,(CtlState)   ; Clearing (8008H)+2 to (8008H)+11
               INC       IX
               INC       IX
               LD        IY,RawCtlState  ; Clearing 73D7H to 73EAH
               LD        B,0AH           ; Clearing 10 words/bytes
L1116          LD        (IX+00H),A       ; Clear byte at (IX)
               INC       IX
               LD        (IY+00H),A       ; Clear word at (IY)
               INC       IY
               LD        (IY+00H),A
               INC       IY
               DEC       B                ; Loop ten times
               JR        NZ,L1116
               LD        (PulseCnt1),A    ; Clear out controller counters
               LD        (PulseCnt2),A    ; and shadows
               LD        (Joy1Shad),A
               LD        (Joy2Shad),A
               LD        (Key1Shad),A
               JR        L1137

; keypad translation table

Keypad_Table  DB        0FH,06H,01H,03H
               DB        09H,00H,0AH,0FH
               DB        02H,0BH,07H,0FH
               DB        05H,04H,08H,0FH

DB 0

;*****
;          1F76   ReadCtlRaw
;
; Update the joystick and keypad shadow bytes.
;*****
A114A
_ReadCtlRaw   IN        A,(IO_Joy1)      ; Update joystick 1 shadow
               CPL
               LD        (Joy1Shad),A
               IN        A,(IO_Joy2)     ; Update joystick 2 shadow
               CPL
               OUT      (IO_KP_Select),A ; Select keypad mode
               LD        (Joy2Shad),A    ; 13 T-states (wait for mode change)
               PUSH     AF                ; 11 T-states
               POP      AF                ; 11 T-states, total 35 (original 31)
               IN        A,(IO_Joy1)     ; Update keyboard 1 shadow
               CPL
               LD        (Key1Shad),A
               IN        A,(IO_Joy2)     ; Update keyboard 2 shadow
               CPL
               OUT      (IO_Joy_Select),A ; Select joystick mode
L1137          LD        (Key2Shad),A
               RET

```

```
*****
; Read value of joystick port.
;
; ENTRY H = 0 for left, 1 for right
;
; EXIT: A = input data
*****
L113D      LD      A,H          ; (these 3 instrs give enough delay)
          OR      A
          JR      NZ,L1146
          IN      A,(IO_Joy1)
          JR      L1148

L1146      IN      A,(IO_Joy2)
L1148      CPL
          RET

*****
;      1FEB      ReadCtlState
;
; This routine reads the raw controller state, then updates the
; table pointed to by (8008H). The format of this table is:
;
; Byte 0 = left controller enable flag
; Byte 1 = right controller enable flag
; Bit 0 = left fire button enable flag
; Bit 1 = directionals enable flag
; Bit 2 = spinner enable flag
; Bit 3 = right fire button enable flag
; Bit 4 = keypad enable flag
; Bit 5 = unused
; Bit 6 = unused
; Bit 7 = controller enable flag
; Bytes 2-6 = left controller result state
; Bytes 7-11 = right controller result state
; +0 = left fire button state (00H or 40H)
; +1 = directionals state (00H-0FH)
; +2 = spinner count (00H-FFH)
; +3 = right fire button state (00H or 40H)
; +4 = keypad state (00H-0EH, or 0FH for none)
;
; State changes are delayed by one call to ReadCtlState, which
; will provide debouncing if this is called once per vertical
; retrace. Debouncing is done with a raw state table at 73D7H:
;
; 73D7-73E0 = left controller raw state
; 73E1-73EA = right controller raw state
; +0 = raw left fire button state
; +2 = raw directionals state
; +4 = unused? (73DB, 73DC, 73E5, 73E6)
; +6 = raw right fire button state
; +8 = raw keypad state
; +0 = raw controller bits
; +1 = state flag (1 = same in prev call, 0 = different)
;
```

```
; The HL register is preserved by this subroutine. (Burgertime depends on
; this for its screen saver timeout at the skill screen.)
```

```
;*****
```

```
_ReadCtlState  CALL    _ReadCtlRaw    ; Read controller state
                LD      IY,RawCtlState ; IY = left controller raw state
                LD      IX,(CtlState)  ; IX = left controller state
                PUSH    IX              ; Save IX
                LD      A,(IX+00H)      ; Check left controller flag
                BIT     7,A             ; Skip if disabled
                JR      Z,L11F3
                LD      B,A            ; Save flag in B
                INC     IX              ; IX = left controller area
                INC     IX
                AND     07H            ; Check joystick enable flags
                JR      Z,L11E8        ; Skip if disabled
                LD      A,(Joy1Shad)    ; Get left controller state
                LD      HL,PulseCnt1    ; Point to left spinner counter
                CALL    JBits          ; Process joystick bits
L11E8          LD      A,B             ; Check keypad enable flags
                AND     18H
                JR      Z,L11F3        ; Skip if disabled
                LD      A,(Key1Shad)    ; Get left keypad state
                CALL    KBits          ; Process keypad bits
L11F3          POP     IX              ; Recover IX
                LD      A,(IX+01H)      ; Check right controller flag
                BIT     7,A
                RET     Z              ; Return if disabled
                LD      B,A            ; Save flag in B
                LD      DE,000AH        ; IY = right controller raw state
                ADD     IY,DE
                LD      E,07H          ; IX = right controller state
                ADD     IX,DE
                AND     07H            ; Check joystick enable flags
                JR      Z,L1214        ; Skip if disabled
                LD      A,(Joy2Shad)    ; Get right controller state
                LD      HL,PulseCnt2    ; Point to right spinner counter
                CALL    JBits          ; Process joystick bits
L1214          LD      A,B             ; Check keypad enable flags
                AND     18H
                RET     Z              ; Return if disabled
                LD      A,(Key2Shad)    ; Get left keypad state
```

```
;      Process keypad bits
; A = keypad state
; B = enable flags
; IX points to small table
; IY points to large table
```

```
KBits          LD      C,A            ; C = keypad state
                BIT     3,B            ; Test right fire button bit
                JR      Z,L1248
```

```
;      Process right fire button state
```

```
KFire          PUSH    BC
```

```

AND    40H            ; Mask out fire button bit
LD     C,A            ; C = fire button bit
LD     B,(IY+06H)      ; B = previous bit
LD     A,(IY+07H)      ; A = current state
OR     A
LD     A,C
JR     NZ,L130B
CP     B              ; State 0 handling
JR     NZ,L130F        ; Has the bit changed (bounce)?

LD     A,01H          ; Next state = 1
LD     (IX+03H),C      ; Store new fire button state
JR     L1313

L130B   CP     B        ; State 1 handling
JR     Z,L1316          ; Has the bit changed?
L130F   LD     (IY+06H),C ; Yes, update it
XOR    A              ; Next state = 0
L1313   LD     (IY+07H),A
L1316   POP    BC

LD     A,C
L1248   BIT    4,B      ; Test keypad bit
RET     Z              ; Don't do keypad if clear

;      Process keypad state

KKeypad  PUSH   BC
AND     0FH           ; Mask out keypad bits
LD     C,A           ; C = keypad bits
LD     B,(IY+08H)     ; B = previous bits
LD     A,(IY+09H)     ; A = current state
OR     A
LD     A,C
JR     NZ,L127A
CP     B              ; State 0 handling
JR     NZ,L127E        ; Have the bits changed (bounce)?

PUSH    HL
LD     HL,Keypad_Table ; Get key code
LD     B,00H
ADD    HL,BC
LD     A,(HL)
POP     HL
LD     (IX+04H),A      ; Store new key code
LD     A,01H          ; Next state = 1
JR     L1282

L127A   CP     B        ; State 1 handling
JR     Z,L1285          ; Have the bits changed?
L127E   LD     (IY+08H),C ; Yes, update them
XOR    A              ; Next state = 0
L1282   LD     (IY+09H),A
L1285   POP    BC
RET

```

```
;      Process joystick bits and spinner counter
; A = joystick state
; B = enable flags
; HL points to spinner counter
; IX points to small table
; IY points to large table

JBits      LD      C,A          ; C = keypad state
           BIT      1,B          ; Test directionals bit
           JR      Z,L1229

;      Process directionals state

JJoystick   PUSH     BC
           AND      0FH          ; Mask out directional bits
           LD      C,A          ; C = directional bits
           LD      B,(IY+02H)    ; B = previous bits
           LD      A,(IY+03H)    ; A = current state
           OR      A
           LD      A,C
           JR      NZ,L12DB
           CP      B            ; State 0 handling
           JR      NZ,L12DF      ; Have the bits changed (bounce)?

           LD      A,01H         ; Next state = 1
           LD      (IX+01H),C     ; Store new directionals state
           JR      L12E3

L12DB       CP      B            ; State 1 handling
           JR      Z,L12E6        ; Have the bits changed?
L12DF       LD      (IY+02H),C    ; Yes, update them
           XOR      A            ; Next state = 0
L12E3       LD      (IY+03H),A
L12E6       POP      BC

           LD      A,C
L1229       BIT      0,B          ; Test left fire button bit
           JR      Z,L1231

;      Process left fire button state

JFire       PUSH     BC
           AND      40H          ; Mask out fire button bit
           LD      C,A          ; C = fire button bit
           LD      B,(IY+00H)    ; B = previous bit
           LD      A,(IY+01H)    ; A = current state
           OR      A
           LD      A,C
           JR      NZ,L12AB
           CP      B            ; State 0 handling
           JR      NZ,L12AF      ; Has the bit changed (bounce)?

           LD      A,01H         ; Next state = 1
           LD      (IX+00H),C     ; Store new fire button state
```

```

                JR      L12B3

L12AB           CP      B                ; State 1 handling
                JR      Z,L12B6          ; Has the bit changed?
L12AF           LD      (IY+00H),C        ; Yes, update it
                XOR     A                ; Next state = 0
L12B3           LD      (IY+01H),A
L12B6           POP     BC

                LD      A,C
L1231           BIT     2,B              ; Test spinner bit
                RET     Z                ; Exit if not set
                LD      A,(HL)           ; Get spinner accumulator
                ADD     A,(IX+02H)        ; Add to (IX + 2)
                LD      (IX+02H),A
                XOR     A                ; Clear spinner accumulator
                LD      (HL),A
                RET

;*****
;
; EVENT TIMER HANDLING ROUTINES BEGIN HERE?
;
;*****

;*****
;
;      1FD3      RunTimers
;
; This routine updates all the event timers.
;*****
_RunTimers      LD      HL,(TimerList)  ; Get timer list pointer
L0F3A           BIT     5,(HL)           ; If the entry is active,
                CALL    Z,L0F49          ; do something with it
                BIT     4,(HL)           ; If end of list,
                RET     NZ               ; return
                INC     HL               ; Point to next entry
                INC     HL
                INC     HL
                JR      L0F3A            ; and loop until end of list

;      Update the timer -- first check for word or byte timer

L0F49           PUSH    HL               ; Save event timer pointer
                BIT     3,(HL)           ; Check "word" flag
                JR      Z,L0F79          ; Branch if byte timer
                BIT     6,(HL)           ; Check "free-running" flag
                JR      NZ,L0F5F         ; Branch if free-running

;      Do one-shot word timer

                INC     HL               ; Get timer value
                LD      E,(HL)
                INC     HL
                LD      D,(HL)
                DEC     DE               ; Decrement timer

```

```
LD      A,E          ; Update it if not =1
OR      D
JR      NZ,L0F8F
DEC     HL           ; Point back to flag byte
DEC     HL
JR      L0F8B        ; Set timer expired flag

;      Do free-running word timer
L0F5F    INC     HL          ; Get timer data word
LD      E,(HL)
INC     HL
LD      D,(HL)
EX      DE,HL          ; Data word points to real timer
LD      E,(HL)         ; Get current value
INC     HL
LD      D,(HL)
DEC     DE            ; Decrement timer
LD      A,E
OR      D
JR      NZ,L0F8F      ; Update it if not =1
INC     HL            ; Get restart value
LD      E,(HL)
INC     HL
LD      D,(HL)
DEC     HL            ; Restart the timer
DEC     HL
LD      (HL),D
DEC     HL
LD      (HL),E
POP     HL            ; Get back timer address
PUSH    HL
JR      L0F8B        ; Set timer expired flag

;      Do byte timer
L0F79    INC     HL          ; Point to second byte of timer
D1AC2 ; DB '5'
DEC     (HL)          ; Decrement timer
JR      NZ,L0F8D      ; Return if not =0
DEC     HL            ; Get back timer address
BIT     6,(HL)        ; Check "free-running" flag
JR      Z,L0F8B      ; If one-shot, set expired flag

;      Do free-running byte timer
INC     HL            ; Get second byte (restart value)
INC     HL
LD      A,(HL)
DEC     HL            ; Restart timer
LD      (HL),A
DEC     HL            ; Get back timer address
L0F8B    SET     7,(HL)      ; Set timer expired flag
L0F8D    POP     HL          ; Get back timer address
RET                    ; and return
```

```
L0F8F      LD      (HL),D          ; Store DE in timer data
           DEC     HL
           LD      (HL),E
           POP     HL              ; Get back timer addr and return
           RET

;*****
;      1F9A      PInitTimers
;
; Parameter block version of 1FC7
;
; Parm 1 (word) = event timer list pointer
; Parm 2 (word) = pointer to aux storage for free-running word timers
;*****
_PInitTimers LD      BC,P_InitTimers
            LD      DE,ParmArea
            CALL    PCopy
            LD      HL,(ParmArea)
            LD      DE,(ParmArea+2)

;*****
;      1FC7      InitTimers
;
; This routine initializes the event timer pointers
;
; Timer counter list consists of 3 byte entries
; +0 timer flags:
; bit 0 = ?
; bit 1 = ?
; bit 2 = ?
; bit 3 = 0 for byte, 1 for word
; bit 4 = 1 for end of list
; bit 5 = 1 if inactive entry
; bit 6 = 0 for one-shot, 1 for free-running
; bit 7 = is set to 1 when timer runs out
; +1,+2 timer data:
; byte one-shot, +1=counter
; byte free-running, +1=counter, +2=restart value
; word one-shot, +1=counter LSB, +2=counter MSB
; word free-running, +1/+2 = pointer to two words
; first word is counter
; second word is restart value
;
; ENTRY HL = event timer list pointer
; DE = pointer to aux storage for free-running word timers
;*****
_InitTimers LD      (TimerList),HL ; Store timer counter list pointer
            LD      (HL),30H       ; Set end of list flag
            EX      DE,HL
            LD      (TimerAux),HL
            RET

;*****
;      1F9D      PStopTimer
```

```

;
; Parameter block version of 1FCA
;
; Parm 1 (byte) = event timer index (0=first timer)
;*****
_PStopTimer    LD      BC,P_StopTimer
               LD      DE,ParmArea+4
               CALL    PCopy
               LD      A,(ParmArea+4)

;*****
;      1FCA    StopTimer
;
; This routine deactivates a timer.  If the timer is a free-
; running word timer, its auxiliary storage is deallocated.
;
; ENTRY A = event timer index (0=first timer)
;
; EXIT: HL = ? (they went to the effort to save something)
;*****
_PStopTimer    LD      HL,(TimerList) ; Get timer list pointer
               OR      A
               JR      Z,L0FD7         ; Branch if parameter =0?
L0FCF          BIT     4,(HL)          ; Test end-of-list flag
               RET     NZ              ; Return if end of list
               INC     HL              ; Point to next timer
               INC     HL
               INC     HL
               DEC     A
               JR      NZ,L0FCF        ; Loop until timer found
L0FD7          BIT     5,(HL)
               RET     NZ              ; Return if timer inactive
               SET     5,(HL)          ; Make timer inactive
               BIT     6,(HL)          ; Test "free-running" flag
               RET     Z               ; Return if one-shot
               BIT     3,(HL)          ; Test byte/word flag
               RET     Z               ; Return if byte
               INC     HL              ; Get "real" timer pointer
               LD      E,(HL)
               INC     HL
               LD      D,(HL)
               PUSH    DE              ; Save "real" timer pointer

;      Update the auxiliary storage pointers of other timers

L0FEE          LD      HL,(TimerList) ; Get timer list pointer
               BIT     4,(HL)          ; Branch if end of list
               JR      NZ,L1020
               BIT     5,(HL)          ; Go to next timer if inactive
               JR      NZ,L0FFD
               LD      A,(HL)          ; Test for free-running word timer
               AND     48H
               CP      48H
L0FFD          INC     HL              ; HL=T+1
               INC     HL              ; HL=T+2

```

```

        JR      NZ,L1019      ; Go to next timer if not
        LD      A,(HL)       ; Compare aux storage adrs of timer
        CP      D            ; we're stopping and current timer
        JR      C,L1019      ; Go to next timer if the stopped
        JR      NZ,L100D     ; timer's storage address is higher
        DEC     HL           ; HL=T+1
        LD      A,(HL)
        CP      E
        INC     HL           ; HL=T+2
        JR      C,L1019
;
        RET      Z           ; Return if storage cross-linked
                                ; (BUG: data still on stack!)
L100D   LD      D,(HL)       ; Get current timer's storage
        DEC     HL           ; HL=T+1; address
        LD      E,(HL)
        DEC     DE           ; Subtract 4
        DEC     DE
        DEC     DE
        DEC     DE
        LD      (HL),E       ; Put it back
        INC     HL           ; HL=T+2
        LD      (HL),D
L1019   INC     HL           ; HL=T+3; Point to next timer
        JR      L0FEE

;      Delete the timer's auxiliary storage

L1020   XOR     A           ; Clear carry
        POP     DE           ; Get aux storage addr to delete
        PUSH    HL           ; Save address of timer record
        LD      HL,(TimerAux) ; Get end of aux storage
        SBC     HL,DE        ; Subtract start of timer
        LD      C,L          ; Set up count
        LD      B,H
        LD      L,E          ; Dest is deleted timer
        LD      H,D
        INC     HL           ; Source is deleted timer + 4
        INC     HL
        INC     HL
        INC     HL
        LDIR          ; Move rest of aux storage down
        LD      BC,-8        ; HL = old end of aux storage + 4
        ADD     HL,BC        ; HL - 8 = new end of aux storage
        LD      (TimerAux),HL ; Store new end of aux storage
        POP     HL           ; Recover address of timer record
        RET

```

```

;*****
;      1FA0   PStartTimer
;
; Parameter block version of 1FCD
;
; Parm 1 (byte) = 0 if one-shot, else free-running
; Parm 2 (word) = max timer count
;*****

```

```

_PStartTimer    LD      BC,P_StartTimer
                LD      DE,ParmArea+5
                CALL    PCopy
                LD      HL,(ParmArea+6)
                LD      A,(ParmArea+5)

;*****
;      1FCD    StartTimer
;
; This routine activates a timer.  If the timer is a free-
; running word timer, its auxiliary storage is allocated.
;
; ENTRY A = 0 if one-shot, else free-running
;      HL = max timer count
;
; EXIT: A = event timer index (0=first timer)
;*****
_PStartTimer    LD      C,A          ; C = timer number
                EX      DE,HL        ; DE = max count
                LD      HL,(TimerList) ; Get pointer to timer list
                XOR     A            ; Timer index = 0
                LD      B,A
L105A           BIT     5,(HL)        ; Test if timer inactive
                JR      Z,L109C      ; Skip to next timer if active
                PUSH    HL           ; Save timer pointer
                LD      A,(HL)       ; Get timer flags
                AND     30H          ; Clear all bits except end of list
;                OR      20H          ; Mark timer inactive (already set!)
                LD      (HL),A       ; Store timer flags
                XOR     A            ; Is it a byte timer?
                OR      D
                JR      NZ,L1074     ; Branch if word timer
                OR      C            ; If "free-running" parameter
                JR      Z,L106E
                SET     6,(HL)       ; then set "free-running" bit
L106E           INC     HL           ; Set current and max values
                LD      (HL),E
                INC     HL
                LD      (HL),E
                JR      L10B6        ; Exit

;      Initialize a word timer

L1074           SET     3,(HL)       ; Set "word timer" bit
                LD      A,C
                OR      A
                JR      Z,L1095      ; Branch if one-shot
                PUSH    DE           ; HL = timer pointer
                LD      DE,(TimerAux) ; DE = next aux storage pointer
                SET     6,(HL)       ; Set "free running" bit
                INC     HL           ; Store aux pointer in timer
                LD      (HL),E
                INC     HL
                LD      (HL),D
                EX      DE,HL        ; HL = aux storage pointer

```

```
        POP     DE                ; Initialize current value
        LD      (HL),E
        INC     HL
        LD      (HL),D
        INC     HL                ; Initialize max value
        LD      (HL),E
        INC     HL
        LD      (HL),D
        INC     HL                ; Update aux storage pointer
        LD      (TimerAux),HL
        JR      L10B6            ; Exit

;      Initialize one-shot word timer

L1095      INC     HL                ; Store current value
        LD      (HL),E
        INC     HL
        LD      (HL),D

;      Activate timer and return index in A

L10B6      POP     HL                ; Recover timer pointer
        RES     5,(HL)            ; Clear "inactive" bit
        LD      A,B                ; A = timer index
        RET

;      Skip to next timer

L109C      INC     B                ; Increment timer index
        BIT     4,(HL)            ; Test for end of list
        RES     4,(HL)            ; Clear "end of list" flag
        INC     HL                ; Point to next timer
        INC     HL
        INC     HL
        JR      Z,L105A            ; Branch if not end of list

;      Timer list full, add a new one

L10A6      LD      (HL),30H        ; Init timer to end of list
        JR      L105A            ; Go check the next timer

; If timers are used with interrupts, the following should be used instead:
;
;;      Skip to next timer
;
;L109C      BIT     4,(HL)            ; Test for end of list
;          JR      NZ,L10A6          ; Branch if end of list
;L10A0      INC     HL                ; Point to next timer
;          INC     HL
;          INC     HL
;          INC     B                ; Increment timer index
;          JR      L105A            ; Go check the next timer
;
;;      Timer list full, add a new one
;
```

```

;L10A6      PUSH    HL
;           INC     HL           ; Point to next timer
;           INC     HL
;           INC     HL
;           LD      (HL),30H     ; Init timer to end of list
;           POP     HL
;           RES     4,(HL)       ; Clear "end of list" flag
;           JR      L10A0       ; Go check the next timer

;*****
;          1FA3    PTestTimer
;
; Parameter block version of 1FD0
;
; Parm 1 (byte) = event timer index (0=first timer)
;*****
_PTestTimer LD      BC,P_TestTimer
            LD      DE,ParamArea+8
            CALL    PCopy
            LD      A,(ParamArea+8)

;*****
;          1FD0    TestTimer
;
; This routine tests a timer's completion flag.
;
; ENTRY A = event timer index (0=first timer)
;
; EXIT: A = 1 if timer expired, else 0
;       Z-flag = set according to A
;*****
_TestTimer LD      C,A           ; C = timer index
            LD      HL,(TimerList) ; HL = timer list pointer
            LD      DE,0003H      ; Timers are 3 bytes each
            OR      A             ; Skip loop if first timer
            JR      Z,L10DE
L10D6      BIT     4,(HL)         ; End of list?
            JR      NZ,L10E6      ; Return 0 if so
            ADD     HL,DE         ; Point to next timer
            DEC     C             ; Decrement count
            JR      NZ,L10D6      ; Loop if not desired timer
L10DE      BIT     5,(HL)         ; Return zero if timer inactive
            JR      NZ,L10E6
            BIT     7,(HL)         ; Return 1 if timer expired
            JR      NZ,L10E9
L10E6      XOR     A             ; Return 0
            RET

L10E9      BIT     6,(HL)         ; Test "free-running" bit
            JR      NZ,L10EF      ; If not free-running,
            SET     5,(HL)         ; set inactive flag
L10EF      RES     7,(HL)         ; Clear expired flag
            XOR     A
            INC     A             ; Return 1
            RET

```

```
L09D5      LD      (IY+01H),A
L09D8      LD      L,(IX+00H)
           LD      H,(IX+01H)
           LD      DE,0005H
           ADD     HL,DE
           EX      DE,HL
           LD      A,(DE)
           LD      L,A
           INC     DE
           LD      A,(DE)
           LD      H,A
           PUSH    HL
           LD      L,(IX+02H)
           LD      H,(IX+03H)
           LD      A,(HL)
           SLA     A
           LD      B,00H
           LD      C,A
           POP     HL
           ADD     HL,BC
           RET

D1A92      DB      'PRESS BUTTON ON KEYPAD.'

           DB 0

;          Default color table

A143B      HEX      00 00 00 F0 F0 F0 F0 F0
           HEX      F0 F0 F0 F0 D0 80 90 B0
           HEX      30 40

;          First row of COLECOVISION

A144D      HEX      60 61 68 69 70 71 78 79 80 81 88 89
           HEX      64 65 6C 74 75 7C 84 85 8C 8D

;          Second row of COLECOVISION

A1463      HEX      62 63 6A 6B 72 73 7A 7B 82 83 8A 8B
           HEX      66 67 6D 76 77 7D 86 87 8E 8F

D1479      DB      'TURN GAME OFF'

D1486      DB      'BEFORE INSERTING CARTRIDGE'

D14A0      DB      'OR EXPANSION MODULE.'

A14B4      DB      1DH          ; (c)
           DB      ' 1982 COLECO'

A14C1      DB      1EH,1FH      ; (tm)

;          Character cell data for COLECOVISION
```

```

A14C3      HEX      00 00 00 00 00 00 00 00
            HEX      3F 7F FF FF F3 F3 F0 F0
            HEX      00 80 C0 C0 C0 C0 00 00
            HEX      3F 7F FF FF F3 F3 F3 F3
            HEX      00 80 C0 C0 C0 C0 C0 C0
            HEX      F0 F0 F0 F0 F0 F0 F0 F0
            HEX      FF FF FF F0 F0 FF FF FF
            HEX      C0 C0 C0 00 00 00 00 00
            HEX      F1 F1 F1 7B 7B 7B 3F 3F
            HEX      E0 E0 E0 C0 C0 C0 80 80
            HEX      1F 3F 7F 79 78 7F 7F 3F
            HEX      80 C0 E0 E0 00 80 C0 E0
            HEX      F3 F3 FB FB FB FF FF FF
            HEX      C0 C0 C0 C0 C0 C0 C0 C0
            HEX      F3 F3 FF FF 7F 3F 00 00
            HEX      C0 C0 C0 C0 80 00 00 00
            HEX      F0 F0 FF FF FF FF 00 00
            HEX      00 00 C0 C0 C0 C0 00 00
            HEX      3F 1F 1F 1F 0E 0E 00 00
            HEX      80 00 00 00 00 00 00 00
            HEX      F0 F0 F0 F0 F0 F0 00 00
            HEX      1F 01 79 7F 3F 1F 00 00
            HEX      E0 E0 E0 E0 C0 80 00 00
            HEX      FF F7 F7 F7 F3 F3 00 00
            HEX      C0 C0 C0 C0 C0 C0 00 00

```

```

;          Character cell data for (c) and TM characters

```

```

A158B      HEX      3C 42 99 A1 A1 99 42 3C ; (C)
            HEX      1F 04 04 04 00 00 00 00 ; T
            HEX      44 6C 54 54 00 00 00 00 ; M

```

```

;          Character cell data for default font

```

```

A15A3      HEX      00 00 00 00 00 00 00 00 ; sp
            HEX      08 1C 1C 18 10 00 30 00 ; !
            HEX      33 33 22 00 00 00 00 00 ; "
            HEX      36 36 7F 36 7F 36 36 00 ; #
            HEX      0C 1F 2C 1E 0D 3E 0C 00 ; $
            HEX      00 63 66 0C 18 33 63 00 ; %
            HEX      38 64 64 38 6D 46 3B 00 ; &
            HEX      18 18 30 00 00 00 00 00 ; '
            HEX      0C 18 30 30 30 18 0C 00 ; (
            HEX      18 0C 06 06 06 0C 18 00 ; )
            HEX      08 6B 3E 1C 3E 6B 08 00 ; *
            HEX      00 08 08 3E 08 08 00 00 ; +
            HEX      00 00 00 00 18 18 30 00 ; ,
            HEX      00 00 00 3E 00 00 00 00 ; -
            HEX      00 00 00 00 00 18 18 00 ; .
            HEX      00 03 06 0C 18 30 60 00 ; /
D1623      HEX      1C 26 63 63 63 32 1C 00 ; 0
            HEX      18 38 18 18 18 18 7E 00 ; 1
            HEX      3E 63 07 1E 3C 70 7F 00 ; 2
            HEX      3F 06 0C 1E 03 63 3E 00 ; 3
            HEX      0E 1E 36 66 7F 06 06 00 ; 4

```

D16AB

```
HEX 7E 60 7E 03 03 63 3E 00 ; 5
HEX 1E 30 60 7E 63 63 3E 00 ; 6
HEX 7F 63 06 0C 18 18 18 00 ; 7
HEX 3C 62 72 3C 4F 43 3E 00 ; 8
HEX 3E 63 63 3F 03 06 3C 00 ; 9
HEX 00 00 18 18 00 18 18 00 ; :
HEX 00 18 18 00 18 18 30 00 ; ;
HEX 06 0C 18 30 18 0C 06 00 ; <
HEX 00 00 3E 00 3E 00 00 00 ; =
HEX 30 18 0C 06 0C 18 30 00 ; >
HEX 3C 66 06 0C 18 00 18 00 ; ?
HEX 3E 63 67 6B 6E 60 3E 00 ; @
HEX 1C 36 63 63 7F 63 63 00 ; A
HEX 7E 63 63 7E 63 63 7E 00 ; B
HEX 1E 33 60 60 60 33 1E 00 ; C
HEX 7C 66 63 63 63 66 7C 00 ; D
HEX 7F 60 60 7E 60 60 7F 00 ; E
HEX 7F 60 60 7E 60 60 60 00 ; F
HEX 1F 30 60 67 63 33 1F 00 ; G
HEX 63 63 63 7F 63 63 63 00 ; H
HEX 3F 0C 0C 0C 0C 0C 3F 00 ; I
HEX 03 03 03 03 03 63 3E 00 ; J
HEX 63 66 6C 78 7C 6E 67 00 ; K
HEX 60 60 60 60 60 60 7F 00 ; L
HEX 63 77 7F 7F 6B 63 63 00 ; M
HEX 63 73 7B 7F 6F 67 63 00 ; N
HEX 3E 63 63 63 63 63 3E 00 ; O
HEX 7E 63 63 63 7E 60 60 00 ; P
HEX 3E 63 63 63 6F 66 3D 00 ; Q
HEX 7E 63 63 67 7C 6E 67 00 ; R
HEX 3C 66 60 3E 03 63 3E 00 ; S
HEX 3F 0C 0C 0C 0C 0C 0C 00 ; T
HEX 63 63 63 63 63 63 3E 00 ; U
HEX 63 63 63 77 3E 1C 08 00 ; V
HEX 63 63 6B 7F 7F 77 63 00 ; W
HEX 63 77 3E 1C 3E 77 63 00 ; X
HEX 33 33 33 1E 0C 0C 0C 00 ; Y
HEX 7F 07 0E 1C 38 70 7F 00 ; Z
HEX 1E 18 18 18 18 18 1E 00 ; [
HEX 00 60 30 18 0C 06 03 00 ; \
HEX 3C 0C 0C 0C 0C 0C 3C 00 ; ]
HEX 08 1C 36 63 00 00 00 00 ; ^
HEX 00 00 00 00 00 00 7F 00 ; _
HEX 18 18 0C 00 00 00 00 00 ; `
HEX 00 00 3C 06 3E 66 3E 00 ; a
HEX 60 60 7C 66 66 66 7C 00 ; b
HEX 00 00 3C 66 60 66 3C 00 ; c
HEX 06 06 3E 66 66 66 3E 00 ; d
HEX 00 00 3C 66 7E 60 3C 00 ; e
HEX 1C 36 30 78 30 30 30 00 ; f
HEX 00 00 3E 66 66 3E 06 3C ; g
HEX 60 60 7C 66 66 66 66 00 ; h
HEX 18 00 38 18 18 18 3C 00 ; i
HEX 06 00 0E 06 06 66 3C 00 ; j
HEX 60 60 66 6C 78 6C 66 00 ; k
```

```

    HEX    38 18 18 18 18 18 3C 00 ; l
    HEX    00 00 76 7F 6B 63 63 00 ; m
    HEX    00 00 7C 66 66 66 66 00 ; n
    HEX    00 00 3C 66 66 66 3C 00 ; o
    HEX    00 00 7C 66 66 7C 60 60 ; p
    HEX    00 00 3E 66 66 3E 06 03 ; q
    HEX    00 00 7C 66 60 60 60 00 ; r
    HEX    00 00 3E 60 3C 06 7C 00 ; s
    HEX    00 18 3C 18 18 18 0C 00 ; t
    HEX    00 00 66 66 66 66 3E 00 ; u
    HEX    00 00 66 66 66 3C 18 00 ; v
    HEX    00 00 63 63 6B 7F 36 00 ; w
    HEX    00 00 66 3C 18 3C 66 00 ; x
    HEX    00 00 66 66 66 3E 06 7C ; y
    HEX    00 00 7E 0C 18 30 7E 00 ; z
    HEX    0C 18 18 30 18 18 0C 00 ; {
    HEX    18 18 18 00 18 18 18 00 ; |
    HEX    30 18 18 0C 18 18 30 00 ; }
    HEX    30 5A 0C 00 00 00 00 00 ; ~
    HEX    54 28 54 28 54 28 54 00 ; del

```

```

;          Character cell to font data mapping for COLECOVISION

```

```

A18A3      HEX    01 02 0E 0F 08 09 12 13 ; CV
           HEX    03 04 0E 0F 05 14 00 00 ; OI
           HEX    05 00 10 11 0A 0B 15 16 ; LS
           HEX    06 07 10 11 05 14 00 00 ; EI
           HEX    01 02 0E 0F 03 04 0E 0F ; CO
           HEX    03 04 0E 0F 0C 0D 17 18 ; ON
           HEX    FF                                ; end of list

```

```

;*****
;          1F82    FillVRAM
;
; Fills a block of VRAM with the same byte
;
; ENTRY HL = start VRAM address
;      DE = count
;      A  = fill byte
; EXIT  AF, C, DE destroyed
;*****

```

```

A18D4
_FillVRAM  LD      C,A                ; Save fill byte
           LD      A,L                ; Output low byte of VRAM address
           OUT     (IO_VDP_Addr),A
           LD      A,H                ; Output high byte of VRAM address
           OR      40H                ; set "this is an address" bit
           OUT     (IO_VDP_Addr),A
L18DD      LD      A,C                ; Get back fill byte (and delay?)
           OUT     (IO_VDP_Data),A    ; Store data byte
           DEC     DE                 ; Decrement count
           LD      A,D
           OR      E
           JR      NZ,L18DD           ; Store bytes until done
           JP      _VDP_Status        ; Read VDP status register

```

```

DB 0

;*****
;      1F85   InitVDP
;
; Initialize the VDP chip
;*****
A18E9
_InitVDP      LD      BC,0000H      ; Register 0: half-text 32x24 mode
              CALL    _WriteReg

              LD      BC,0180H      ; Reg 1, 32x24 mode, 16K DRAM, blank
              CALL    _WriteReg

NOP
NOP

A18F7                                ; (called by Tomarc the Barbarian)

              LD      A,03H          ; Pattern generator table
              LD      HL,0000H       ; VRAM address 0000H
              CALL    _BaseLoad

              LD      A,02H          ; Name table (display)
              LD      HL,1800H       ; VRAM address 1800H
              CALL    _BaseLoad

              XOR     A              ; Sprite attribute table
              LD      HL,1B00H       ; VRAM address 1B00H
              CALL    _BaseLoad

              LD      A,04H          ; Color table
              LD      HL,2000H       ; VRAM address 2000H
              CALL    _BaseLoad

              LD      A,01H          ; Sprite pattern generator
D1AC5      EQU     $+2              ; DB '8'
              LD      HL,3800H       ; VRAM address 3800H
              CALL    _BaseLoad

              LD      BC,0700H       ; Reg 7 (FG/BG color), set both to 0
              JP      _WriteReg

;*****
;      1F7F   InitFont
;
; Initialize pattern generator (font) table
;*****
_InitFont     LD      HL,A158B       ; Point to main ASCII bitmaps
              LD      DE,001DH       ; First character code = 1DH
              LD      IY,0063H       ; bug fixed (was 0060)
              CALL    _BlkWrtVRAM3   ; Pattern generator table

              LD      HL,A15A3       ; Point to blank bitmap
              LD      DE,0000H       ; First character code = 00H

```

```

_BlkWrtVRAM31  LD      A,03H          ; Pattern generator table
                JP      L1EA7        ; LD IY,0001H / JP _BlkWrtVRAM

;*****
;
; Search for '/' character in text pointed to by HL.
;
; ENTRY HL points to text
;
; EXIT: HL points to '/' character
;       BC = number of bytes before '/' character
;
;*****
L1946          LD      BC,0000H      ; Initialize count to zero
L1949          LD      A,(HL)        ; Get next byte
                CP      '/'          ; Return if '/' character
                RET      Z
                INC     HL           ; Point to next byte
                INC     BC           ; Increment count
                JR      L1949        ; Go back for more

;*****
;
; Center a line of text on the screen.
;
; ENTRY BC = length of text
;       DE = pointer to text
;       HL = screen position at left end of line
;
;*****
L1951          PUSH    BC            ; Move BC to IY (count)
                POP     IY
                LD      A,20H        ; A = (20H - C) / 2
                SBC     A,C          ; (centering offset)
                RRA
                LD      B,00H
                LD      C,A
                ADD     HL,BC        ; HL = HL + BC
                EX      DE,HL        ; HL = text ptr, DE = scrn position
                JP      _BlkWrtVRAM2 ; LD A,02H / JP _BlkWrtVRAM

;*****
;
; Delay routines
;
; 1968 is the infamous "seven second delay" and is apparently
; not called elsewhere in the ROM and shouldn't be called by
; "well behaved" cartridges. Which means that it would be safe
; to change, except that some Xonox cartridges call it. However,
; shortening the delay would probably make some of those games
; more playable, so we can change it in the name of humanity.
;
; 196B delays using a count in the HL register. The delay is
; approximately H/3 seconds.
;

```

```

;*****
;A1968      LD      HL,1700H      ; Too long
;A196B      LD      DE,00FFH      ; Load inner loop count
;L196E      DEC     DE
;           LD      A,D
;           OR      E
;           JR      NZ,L196E
;           DEC     HL
;           LD      A,H
;           OR      L
;           JR      NZ,A196B
;           RET

; This replacement is based on Kev's patch.  Every outer loop it checks
; both left fire buttons and aborts the delay loop if either is pressed.
;
; The main difference is that I save the AF and BC registers on the
; assumption that anyone "byte-fisted" enough to call the ROM instead of
; spending a few bytes to write his own might also expect it not to mess
; with the A register or B register.
;
; The other side-effect of this replacement delay routine is that the
; joystick ports are left in stick mode, but this is probably okay.

L196C      JR      Z,L1978      ; exit if pressed

           LD      DE,00E8H      ; load inner loop counter (about 10% less)
L196E      DEC     DE            ; do inner loop
           LD      A,D
           OR      E
           JR      NZ,L196E

           DEC     HL            ; do outer loop
           LD      A,H
           OR      L
           JR      NZ,L196B

L1978      POP     BC            ; restore regs & return
           POP     AF
           RET

A1968      LD      HL,1700H      ; startup screen delay timer
A196B      PUSH    AF            ; original routine didn't mess with A or B!
           PUSH    BC

L196B      OUT     (0C0H),A      ; select left fire/stick mode
           IN      A,(0FCH)      ; read left fire button
           LD      B,A
           IN      A,(0FFH)      ; read right fire button
           AND     B              ; OR the two sticks together
           AND     40H           ; mask out the fire button bit
           JR      L196C

;*****
;           1F7C      SkillScrn

```

```

;
; Displays the skill select screen and returns
;*****
A1979
_SkillScrn    CALL    InitScrn        ; Clear VRAM and initialize the VDP

              LD      BC,0F04H        ; Reg 0FH (FG/BG color), value = 4
              CALL    _WriteReg       ; write it

              LD      HL,Skill_Msgs   ; Put up skill screen messages
              CALL    ScrnMsgs

              LD      HL,(ClrTabShad) ; Get color table VRAM addr
              LD      DE,0020H        ; 32 bytes
              LD      A,0F4H          ; Fill with F4H
              CALL    _FillVRAM       ; fill it

              LD      BC,01C0H        ; Unblank the screen
              JP      _WriteReg

D1AC9         EQU     $+3             ; DB 'S'
D1A7C         DB      'TO SELECT GAME OPTION,'
D1AA9         DB      '1 = SKILL 1/ONE PLAYER'

;
; HL=msg, IY=len, DE=scrn
Skill_Msgs    DW      D1A7C,22*1024+0025H ; 'TO SELECT GAME OPTION,'
              DW      D1A92,23*1024+0065H ; 'PRESS BUTTON ON KEYPAD.'
              DW      D1AA9,22*1024+00C5H ; '1 = SKILL 1/ONE PLAYER'
              DW      D1AA9,22*1024+0105H ; '1 = SKILL 1/ONE PLAYER'
              DW      D1AA9,22*1024+0145H ; '1 = SKILL 1/ONE PLAYER'
              DW      D1AA9,22*1024+0185H ; '1 = SKILL 1/ONE PLAYER'
              DW      D1AA9,22*1024+01E5H ; '1 = SKILL 1/ONE PLAYER'
              DW      D1AA9,22*1024+0225H ; '1 = SKILL 1/ONE PLAYER'
              DW      D1AA9,22*1024+0265H ; '1 = SKILL 1/ONE PLAYER'
              DW      D1AA9,22*1024+02A5H ; '1 = SKILL 1/ONE PLAYER'
              DW      D1ABF,01*1024+0105H ; '2'
              DW      D1AC0,01*1024+0145H ; '3'
              DW      D1AC1,01*1024+0185H ; '4'
              DW      D1AC2,01*1024+01E5H ; '5'
              DW      D1AC3,01*1024+0225H ; '6'
              DW      D1AC4,01*1024+0265H ; '7'
              DW      D1AC5,01*1024+02A5H ; '8'
              DW      D1ABF,01*1024+010FH ; '2'
              DW      D1AC0,01*1024+014FH ; '3'
              DW      D1AC1,01*1024+018FH ; '4'
              DW      D1AC6,03*1024+01F1H ; 'Two'
              DW      D1AC6,03*1024+0231H ; 'Two'
              DW      D1AC6,03*1024+0271H ; 'Two'
              DW      D1AC6,03*1024+02B1H ; 'Two'
              DW      D1ABF,01*1024+022FH ; '2'
              DW      D1AC0,01*1024+026FH ; '3'
              DW      D1AC1,01*1024+02AFH ; '4'
              DW      D1AC9,01*1024+01FBH ; 'S'
              DW      D1AC9,01*1024+023BH ; 'S'
              DW      D1AC9,01*1024+027BH ; 'S'

```

```

        DW      D1AC9,01*1024+02BBH      ; 'S'
        DW      0                          ; End of table

;*****
; Put up the "no cartridge" message
;*****

L13FF      LD      HL,No_Cart_Msgs ; Put up "TURN GAME OFF" messages
          CALL     ScrnMsgs

          LD      HL,8A00H           ; Wait a long time for the user to see
          CALL     A196B

          LD      BC,0180H          ; Then blank the screen
          CALL     _WriteReg

L1439      JR      L1439              ; and go to sleep
          NOP                      ; to allow for patching

;
; HL=msg, IY=len, DE=scrn
No_Cart_Msgs DW      D1479,13*1024+01AAH      ; 'TURN GAME OFF'
              DW      D1486,26*1024+01E4H      ; 'BEFORE INSERTING CARTRIDGE'
              DW      D14A0,20*1024+0227H      ; 'OR EXPANSION MODULE.'
              DW      0                          ; End of table

;*****
; Display a bunch of text all over the screen
;
; ENTRY: HL=table pointer
; EXIT:  BC=0
;
; The table is a bunch of 4-byte entries:
; 0-1    address of message text data
; 2      low byte of screen address
; 3      high byte of screen address in low 2 bits
;        message text length in high 6 bits
;*****

ScrnMsgs   LD      C,(HL)            ; Get message address to BC
          INC      HL
          LD      B,(HL)
          INC      HL

          LD      A,C                ; Test for end of table
          OR      B
          RET     Z

          LD      E,(HL)            ; Get display address to DE
          INC      HL
          LD      A,(HL)
          AND     03H
          LD      D,A

          LD      A,(HL)            ; Get message length to A
          INC      HL

```

```

        RRCA
        RRCA
        AND     3FH

        PUSH    BC           ; Save message address
        EX      (SP),HL      ; Msg addr to HL, save table pointer

        LD      C,A          ; Move message length to IY
        LD      B,0
        PUSH    BC
        POP     IY

        CALL    _BlkWrtVRAM2 ; LD A,02H/JP _BlkWrtVRAM

        POP     HL           ; Recover table pointer
        JR      ScrnMsgs     ; Loop for next message

;*****
;      1FAF
;
; Parameter block version of 1FE5
;
; Parm 1 (byte) = ?
; Parm 2 (lit)  = ?
;*****
L0655      LD      BC,D064F
           LD      DE,ParmArea
           CALL    PCopy
           LD      A,(ParmArea)
           LD      HL,(ParmArea+1)

;*****
;      1FE5
;
; ENTRY A = ?
;      HL = ?
;*****
L0664      LD      (73CAH),A
           XOR     A
;
           LD      (73CBH),A
           LD      (73CCH),A
           LD      (73D1H),HL
;
           LD      (73CDH),HL
           LD      (73CFH),HL
           JP      L0669

;*****
;      1F88      ReadSpinner
;
; This was included to support the "spinner" control that
; was originally planned to be in the regular controllers.
; It was later included in the Super Action controllers.
; Early versions of the controller circuit board even have
; circuit traces and holes for the spinner parts.
;
```

```
; Apparently the 20H bit is a direction bit, and the 10H
; bit is a "pulse" bit.
;*****
_ReadSpinner    LD      HL,PulseCnt1    ; Point to first pulse counter
                IN      A,(IO_Joy1)    ; Read joystick 1
                CALL    L117D          ; Do first pulse counter
                INC     HL              ; Point to second pulse counter
                IN      A,(IO_Joy2)    ; Read joystick 2

L117D           BIT     4,A              ; Test 10H bit
                RET     NZ              ; Return if set (no pulse)
L1182           BIT     5,A              ; Test 20H bit
                JR      NZ,L1189        ; Inc counter if set
                DEC     (HL)            ; Dec counter if clear
                RET

D1AC1 ; DB '4'
L1189           INC     (HL)
                RET

;*****
;      1FE8
;
;
;*****
L0679           LD      A,(D73C6)
                PUSH    AF
                XOR     A
                LD      (D73C6),A
L0682           LD      A,(73CCH)
                LD      HL,73CBH
                CP      (HL)
                JR      Z,L06BC
                LD      HL,(73CFH)
                LD      E,(HL)
                INC     HL
                LD      D,(HL)
                INC     HL
                LD      B,(HL)
                INC     HL
                PUSH    DE
                POP     IX
                PUSH    HL
                CALL    L06E3
                LD      A,(73CCH)
                INC     A
                LD      HL,73CAH
                CP      (HL)
                JR      NZ,L06B3
                XOR     A
                LD      HL,(73D1H)
                LD      (73CFH),HL
                POP     HL
                JR      L06B9
```

```

L06B3      POP      HL
           LD        (73CFH),HL
L06B9      LD        (73CCH),A
           JR        L0682

L06BC      POP      AF
           LD        (D73C6),A
           RET

DB 0,0,0,0,0,0,0,0,0,0
DB 0,0,0,0,0

;*****
;      1F8B      PBaseLoad
;
; Parameter block version of 1FB8
;
; P1 (byte) = base register code
; P2 (word) = raw unshifted VRAM address
;*****
_PBaseLoad LD      BC,P_BaseLoad
           LD      DE,ParamArea
           CALL    PCopy
           LD      A,(ParamArea)
           LD      HL,(ParamArea+1)

;*****
;      1FB8      BaseLoad
;
; This routine loads a VDP base register with the
; specified base address. The base address is stored
; in one of the shadow registers at 73F2.
;
; ENTRY A = base register code
;      HL = raw unshifted VRAM address
;*****
A1B1D
_BaseLoad LD      C,A          ; VDP register code to BC
           LD      B,00H
           LD      IX,VDPBaseShad
           ADD     IX,BC        ; Index into shadow address table
           ADD     IX,BC
           LD      (IX+00H),L    ; Store shadow copy of base address
           LD      (IX+01H),H
           LD      A,(VDP0Shad) ; Get current graphics mode
           BIT     1,A
           JR      Z,L1B5C      ; If not in hi-res graphics, go shift
           LD      A,C
           CP      03H          ; Special handling for pattern gen
           JR      Z,L1B40
           CP      04H          ; and color tables
           JR      NZ,L1B5C      ; Else branch to go shift

;      Set color table base address in hi-res graphics mode

```

```

        LD      B,03H          ; VDP register 3 = color table
        LD      A,L            ; If VRAM addr = 0,
        OR      H
        LD      C,7FH          ; set base register to 7FH (1FC0H)
        JR      Z,L1B72
        LD      C,0FFH         ; Else set it to FFH (3FC0H)
        JR      L1B72

;      Set pattern generator base address in hi-res graphics mode

L1B40    LD      B,04H          ; VDP register 4 = pattern generator
        LD      A,L            ; If VRAM addr = 0,
        OR      H
        LD      C,03H          ; set base register to 03H (1800H)
        JR      Z,L1B72
        LD      C,07H          ; Else set it to 07H (3800H)
        JR      L1B72

L1B5C    LD      IY,D1B76       ; Get address of shift table
        ADD     IY,BC           ; Index into it
        ADD     IY,BC
        LD      A,(IY+00H)      ; Get shift count
        LD      B,(IY+01H)      ; Get VDP register number
L1B6A    SRL     H               ; Shift the base address right
        RR      L               ; to adjust for VDP register
        DEC     A
        JR      NZ,L1B6A
        LD      C,L             ; Get shifted VDP register data
L1B72    JP      _WriteReg       ; Write it to the VDP chip

; Shift table for VDP base addresses (shift, reg num)

D1B76    DB      7,5           ; 0 = Sprite attribute table
        DB      11,6           ; 1 = Sprite pattern generator
        DB      10,2           ; 2 = Name table
        DB      11,4           ; 3 = Pattern generator
        DB      6,3            ; 4 = Color table

; ReadVRAM with the bug fixed

XReadVRAM LD      A,C           ; no bug if C=00H
        OR      A
        JR      Z,XReadVRAM_a

        INC     B               ; else account for the missing page

XReadVRAM_a JP      ReadVRAM

;*****
;      1F8E    PBlkReadVRAM
;
; Parameter block version of 1FBB
;
; Parm 1 (byte) = VDP table number (0-4)
; Parm 2 (byte) = VDP table index LSB

```

```

; Parm 3 (byte) = VDP table index MSB
; Parm 4 (lit) = pointer to data buffer
; Parm 5 (word) = VDP table entry count
;*****
P_BlReadVRAM
P_BlkWrtVRAM    DB      5
                DW      1,1,1,-2,2

_PBlReadVRAM    LD      BC,P_BlReadVRAM
                LD      DE,ParmArea
                CALL    PCopy
                LD      A,(ParmArea)
                LD      DE,(ParmArea+1)
                LD      IY,(ParmArea+5)
                LD      HL,(ParmArea+3)

;*****
;      1FBB    BlkReadVRAM
;
; ENTRY A = VDP table number (0-4)
;      DE = VDP table index
;      HL points to data buffer
;      IY = VDP table entry count
;*****
A1BA3
_BlReadVRAM     CALL    VRAM_Index
                JR      XReadVRAM

DB 0,0

;*****
;      VRAM_Index
;
; This routine returns the actual VRAM address and size
; of a block of VDP table entries.
;
; ENTRY DE = VDP table index
;      IY = number of VDP table entries
;      A = VDP table number (0-4)
;
; EXIT: BC = number of bytes to move to VRAM
;      DE = VRAM address
;      HL register is not changed
;*****
A1BAA
VRAM_Index      LD      (D73FE),IY      ; Save byte count
                LD      IX,VDPBaseShad ; VDP base addr shadow array
                LD      C,A             ; BC = VRAM table index
                LD      B,00H
                CP      04H             ; Check if color table
                JR      NZ,L1BC0
                LD      A,(VDP0Shad)    ; Yes, check for hi-res graphics
                BIT     1,A             ; Don't shift color table unless
                JR      Z,L1BEC         ; using hi-res graphics
L1BC0           LD      IY,D1BFF        ; Point to shift table

```

```

                ADD    IY,BC          ; Index into shift table
                LD     A,(IY+00H)      ; Get shift value
                OR     A
                JR     Z,L1BEC         ; Branch if no shift
L1BCD           SLA     E              ; Shift VRAM offset
                RL     D
                DEC    A
                JR     NZ,L1BCD        ; until done
                PUSH   BC              ; Save VRAM table index
                LD     BC,(D73FE)      ; Get original byte count
                LD     A,(IY+00H)      ; Get shift value
                OR     A
                JR     Z,L1BEB         ; Branch if no shift
L1BE0           SLA     C              ; Shift byte count
                RL     B
                DEC    A
                JR     NZ,L1BE0        ; until done
                LD     (D73FE),BC      ; Save shifted byte count
L1BEB           POP    BC              ; Get back VRAM table index
L1BEC           PUSH   HL              ; Save HL
                ADD    IX,BC           ; Index into base addr shadow array
                ADD    IX,BC
                LD     L,(IX+00H)      ; Get base VRAM address from shadow
                LD     H,(IX+01H)
                ADD    HL,DE           ; Add shifted offset to base addr
                EX     DE,HL           ; Put VRAM address in DE
                POP    HL              ; Restore HL
                LD     BC,(D73FE)      ; Get byte count
                RET

```

; Shift table for indexes into VDP tables (1, 4, or 8 bytes per entry)

```

D1BFF          DB      2              ; 0 = Sprite attribute table
                DB      3              ; 1 = Sprite pattern generator table
                DB      0              ; 2 = Name table
                DB      3              ; 3 = Pattern generator table
                DB      3              ; 4 = Color table

```

; Copy high 4 bits of (HL) into low 4 bits of (HL)

```

L01A6          LD      A,(HL)
                AND     0F0H
                LD      B,A
                RRCA
                RRCA
                RRCA
                RRCA
                OR      B
                LD      (HL),A
                RET

```

;*****

; 1F91 PBlkWrtVRAM

; Parameter block version of 1FBE

```

;
; Parm 1 (byte) = VDP table number (0-4)
; Parm 2 (byte) = VDP table index LSB
; Parm 3 (byte) = VDP table index MSB
; Parm 4 (lit) = pointer to data to be written
; Parm 5 (word) = VDP table entry count
;*****
_PBlkWrtVRAM    LD      BC,P_BlkWrtVRAM
                LD      DE,ParamArea
                CALL    PCopy
                LD      A,(ParamArea)
                LD      DE,(ParamArea+1)
                LD      IY,(ParamArea+5)
                LD      HL,(ParamArea+3)

                DB      01H          ; LD BC,nnnn
_BlkWrtVRAM2    LD      A,02H       ; name table (character cells)

;*****
;      1FBE    BlkWrtVRAM
;
; ENTRY A = VDP table number (0-4)
;      DE = VDP table index
;      HL points to data to be written
;      IY = VDP table entry count
;      73C7H = 01H to write to RAM sprite attr table
;*****
A1C27
_BlkWrtVRAM    PUSH    AF           ; Save VDP table number
                OR      A           ; If not sprite attribute table,
                JR      NZ,L1C4E     ; always write to the VDP
                LD      A,(WrtRAMSprt) ; Check RAM sprite attr table flag
                DEC     A
                JR      NZ,L1C4E     ; Write to VDP if not 01H
                EX      (SP),HL      ; Clean up stack and save data pointer
                LD      HL,(RamSprtTab) ; Get pointer at 8002H
                ADD     HL,DE        ; DE*4 = sprite table
                ADD     HL,DE
                ADD     HL,DE
                ADD     HL,DE        ; dest addr = (8002H) + DE
                EX      DE,HL
                PUSH    IY           ; BC = table entry count
                POP     BC
                SLA     C            ; C = C * 4 for sprite table
                SLA     C            ; I hope BC was <= 3FH!
                POP     HL          ; Restore (source) data pointer
                LDIR                     ; Copy it
                RET

_BlkWrtVRAM3    LD      A,03H       ; Pattern generator table
                DB      01H          ; LD BC,nnnn
_BlkWrtVRAM4    LD      A,04H       ; Color table
                JR      _BlkWrtVRAM

                DB 0,0

```

```
L1C4E      POP      AF          ; Restore VDP table number
A1C4F      ; (Called by Aquattack)
          CALL      VRAM_Index   ; Adjust for VDP table size
          JR        XWrtVRAM     ; Write the data to VRAM

;*****
;      1F94      PInitRAMSprt
;
; Parameter block version of 1FC1
;
; Parm 1 (byte) = number of sprites
;*****
_PInitRAMSprt LD      BC,P_InitRAMSprt
              LD      DE,ParamArea
              CALL    PCopy
              LD      A,(ParamArea)

;*****
;      1FC1      InitRAMSprt
;
; This routine initializes the RAM sprite index table
; pointed to by 8004H with values from 00H to A-1
;
; ENTRY A = number of sprites
;*****
_InitRAMSprt LD      B,A          ; B = count
              XOR     A           ; Init fill to zero
              LD      HL,(RAMSprtIdx) ; Get pointer to RAM sprite index table
L1C6B      LD      (HL),A        ; Initialize array entry
              INC     HL          ; Increment array pointer
              INC     A           ; Increment fill value
              CP      B           ; Go back until done
              JR      NZ,L1C6B
              RET

; WrtVRAM with the bug fixed

XWrtVRAM   LD      A,C           ; no bug if C=00H
          OR      A
          JR      Z,XWrtVRAM_a

          INC     B              ; else account for the missing page

XWrtVRAM_a JP      _WrtVRAM

DB 0,0

;*****
;      1F97      PCopyRAMSprt
;
; Parameter block version of 1FC4
;
; Parm 1 (byte) = number of sprites
;*****
```

```

_PCopyRAMSprt  LD      BC,P_CopyRAMSprt
               LD      DE,ParamArea
               CALL    PCopy
               LD      A,(ParamArea)

;*****
;      1FC4    CopyRAMSprt
;
; This routine copies data from the RAM sprite attribute
; table at (8002H) to VRAM. The RAM sprite index table at
; (8004H) contains a list of one byte indexes into the RAM
; sprite attribute table. Each byte corresponds to one real
; (VRAM) sprite attr table entry and references one RAM
; sprite attr table entry to use as a template.
;
; ENTRY A = number of sprites
;*****
A1C82
_PCopyRAMSprt  LD      IX,(RAMSprtIdx) ; Get pointer to RAM sprite index table
               PUSH    AF
               LD      IY,SprtTabShad ; Get sprite table base VRAM address
               LD      E,(IY+00H)
               LD      D,(IY+01H)
               LD      A,E              ; Send LSB of address
               OUT     (IO_VDP_Addr),A
               LD      A,D
               OR      40H              ; Set write flag
               OUT     (IO_VDP_Addr),A ; Send MSB of address
               POP     AF
L1C9A          LD      HL,(RamSprtTab) ; Get pointer at 8002H
               LD      C,(IX+00H)      ; Get next index byte from 8004H table
               INC     IX
               LD      B,00H            ; HL = (8002H) + (IX)*4
               ADD     HL,BC
               ADD     HL,BC
               ADD     HL,BC
               ADD     HL,BC
               LD      B,04H            ; B = count for 4 bytes of data
               LD      C,IO_VDP_Data    ; C = output port
L1CAC          OUTI     ; Output a byte of data
               NOP      ; Wait for the VDP to catch up
               NOP
               JR      NZ,L1CAC          ; Loop until 4 bytes copied
               DEC     A
               JR      NZ,L1C9A          ; Loop until all sprites copied
               RET

DB 0,0,0,0,0,0

;*****
;      1FA6    PWriteReg
;
; Parameter block version of 1FD9
;
; Parm 1 (byte) = VDP register number

```

```

; Parm 2 (byte) = data to write to register
;*****

_PWriteReg      LD      BC,P_WriteReg
                LD      DE,ParmArea
                CALL    PCopy
                LD      HL,(ParmArea)
                LD      C,H
                LD      B,L

;*****
;          1FD9   WriteReg
;
; ENTRY B = VDP register number
;          C = data to write to register
;*****
A1CCA
_PWriteReg      LD      A,C          ; Send VDP register data
                OUT     (IO_VDP_Addr),A
                LD      A,B
                ADD     A,80H        ; Set "write register" bit
                OUT     (IO_VDP_Addr),A ; Send register number
                LD      A,B          ; If register 0...
                OR      A
                JR      NZ,L1CDB
                LD      A,C
                LD      (VDP0Shad),A ; ...update register 0 shadow
L1CDB           LD      A,B          ; If register 1...
                CP      01H ; If this is changed to DEC A, Smurf won't work!
                RET     NZ
                LD      A,C
                LD      (VDP1Shad),A ; ...update register 1 shadow
                RET

;          Decrement low 4 bits of (HL)

L0190           XOR     A
                RRD
                SUB     01H
                PUSH    AF
                RLD
                POP     AF
                RET

;          Decrement high 4 bits of (HL) (NOT USED!)
;
;L019B          XOR     A
;                RLD
;                SUB     01H
;                PUSH    AF
;                RRD
;                POP     AF
;                RET

```

```

;*****

```

```

;      1FA9    PWrVRAM
;
; Parameter block version of 1FDF
;
; Parm 1 (lit) = pointer to data to be written
; Parm 2 (word) = VRAM address
; Parm 3 (word) = byte count
;*****
_PWrVRam      LD      BC,P_WrVRAM
              LD      DE,ParmArea
              CALL    PCopy
              LD      HL,(ParmArea)
              LD      DE,(ParmArea+2)
              LD      BC,(ParmArea+4)

;*****
;      1FDF    WrVRAM
;
; ENTRY HL points to data to be written
;      DE = VRAM address
;      BC = byte count (see note)
; EXIT: HL = first byte after data that was written
;      AF, B, D destroyed
;
; Note: there is a bug which causes 256 too few bytes to be
;      copied if the B and C registers are both non-zero,
;      and there are games which do depend on this!
;      BlkWrVRAM is also affected by this, but is
;      probably never called with enough bytes to matter.
;*****
A1D01
_WrVRAM      LD      A,E          ; Send LSB of address
              OUT     (IO_VDP_Addr),A
              LD      A,D
              ADD     A,40H        ; Send MSB of address + 40H
              OUT     (IO_VDP_Addr),A
              LD      D,B          ; D = MSB of byte count
              LD      B,C          ; B = LSB of byte count
              LD      C,IO_VDP_Data ; C = port address
L1D14        OUTI     ; Output a byte of data
              NOP        ; Wait for the VDP to catch up
              NOP
              JR      NZ,L1D14     ; LSB loop
              DEC     D           ; Decrement MSB of count
              RET      M          ; MSB loop
              JR      NZ,L1D14     ; BUG: should have been JR L1D14
              RET

;
;      HL=msg,IY=len,DE=scrn
Title_Msgs   DW      A144D,22*1024+0085H ; first row of COLECOVISION
              DW      A1463,22*1024+00A5H ; second row of COLECOVISION
              DW      A14C1,02*1024+009BH ; TM
              DW      A14B4,13*1024+02AAH ; 'c 1982 COLECO'
              DW      0           ; End of table

```

```

;*****
;      1FAC    PReadVRAM
;
; Parameter block version of 1FE2
;
; Parm 1 (lit) = pointer to data buffer
; Parm 2 (word) = VRAM address
; Parm 3 (word) = byte count
;*****

_PReadVRAM    LD      BC,P_ReadVRam
              LD      DE,ParamArea
              CALL    PCopy
              LD      HL,(ParamArea)
              LD      DE,(ParamArea+2)
              LD      BC,(ParamArea+4)

;*****
;      1FE2    ReadVRAM
;
; ENTRY HL points to data buffer
;      DE = VRAM address
;      BC = byte count (see note)
; EXIT: HL = first byte after data that was read
;      AF, B, D destroyed
;
; Note: there is a bug which causes 256 too few bytes to be
;      copied if the B and C registers are both non-zero,
;      and there are games which do depend on this!
;      BlkReadVRAM is also affected by this, but is
;      probably never called with enough bytes to matter.
;*****
A1D3E
_ReadVRAM     LD      A,E          ; Send LSB of address
              OUT     (IO_VDP_Addr),A
              LD      A,D          ; Send MSB of address
A1D43         EQU     $+1          ; (Used by many Coleco carts)
              OUT     (IO_VDP_Addr),A
              LD      D,B          ; D = MSB of byte count
              LD      B,C          ; B = LSB of byte count
A1D47         EQU     $+1          ; (Used by many Coleco carts)
              LD      C,IO_VDP_Data ; C = port address
L1D49         INI             ; Input a byte of data
              NOP             ; Wait for the VDP to catch up
              NOP
              JR      NZ,L1D49     ; LSB loop
              DEC     D           ; Decrement MSB of count
              RET     M           ; MSB loop
              JR      NZ,L1D49     ; BUG: should have been JR L1D49
              RET

DB 0,0,0,0

;*****
;      1FDC    VDP_Status

```

```
;
; This routine returns the VDP status byte in A.
;
; EXIT: A = VDP status byte
;*****
A1D57
_VDP_Status      IN      A,(IO_VDP_Status) ; Get VDP status
                 RET                      ; and return

;*****
;      1F6A      FlipRL
;
; Right/Left flip a list of images
;
; ENTRY A = VDP table code (0-4)
;      BC = number of VDP table entries
;      DE = source VDP table index
;      HL = destination VDP table index
;*****
A1D5A
_FlipRL          LD      IX,__FlipRL
                 JR      L1D70

;*****
;      1F6D      FlipUD
;
; Up/Down flip a list of images
;
; ENTRY A = VDP table code (0-4)
;      BC = number of VDP table entries
;      DE = source VDP table index
;      HL = destination VDP table index
;*****
A1D60
_FlipUD          LD      IX,__FlipUD
                 JR      L1D70

;*****
;      1F70      Rotate
;
; Rotate a list of images by 90 degrees clockwise
;
; ENTRY A = VDP table code (0-4)
;      BC = number of VDP table entries
;      DE = source VDP table index
;      HL = destination VDP table index
;*****
A1D66
_Rotate          LD      IX,__Rotate
                 JR      L1D70

;*****
;      1F73      Expand
;
; Expand a list of 8x8 images to 16x16
```

```

;
; ENTRY A = VDP table code (0-4)
;      BC = number of VDP table entries
;      DE = source VDP table index
;      HL = destination VDP table index
;*****
__Expand      LD      IX,__Expand
L1D70         EXX                     ; Swap register sets
             EX      AF,AF'          ; AF' = VDP table
             PUSH    IX              ; Save routine address
L1D74         EX      AF,AF'          ; AF = VDP table
             PUSH    AF
             EX      AF,AF'
             POP     AF
             EXX                     ; DE = original DE
             PUSH    DE
             EXX
             POP     DE
             LD      HL,(VDP_Temp)    ; HL = (VDP_Temp)
             CALL    L1E92            ; LD IY,0001H / JP _BlkReadVRAM
             POP     IX              ; IX = routine address
             PUSH    IX
             JP      (IX)            ; Jump into the routine

__Rotate      LD      HL,(VDP_Temp)    ; HL = temp
             LD      BC,0008H         ; DE = temp + 8
             LD      E,L
             LD      D,H
             ADD     HL,BC
             EX      DE,HL
             CALL    DoRotate          ; Rotate the image
             CALL    L1E72            ; Copy the image back to VRAM
             CALL    L1E5D            ; Check for hi-res graphics
             JR      Z,L1E02          ; Exit if not hi-res graphics
             CALL    L1E89            ; Read the source color table entry
             CALL    L1E9A            ; Write the dest color table entry
L1E02         JR      L1D8CEXX         ; Done with routine

__FlipRL      LD      HL,(VDP_Temp)    ; DE = temp
             LD      BC,0008H         ; HL = temp + 8
             LD      E,L
             LD      D,H
             ADD     HL,BC
             EX      DE,HL
             CALL    DoFlipLR          ; Flip the image
             CALL    L1E72            ; Copy the image back to VRAM
             CALL    L1E5D            ; Check for hi-res graphics
             JR      Z,L1DB3          ; Exit if not hi-res graphics
             CALL    L1E89            ; Read source color table entry
             CALL    L1E9A            ; Write dest color table entry
L1DB3         EXX                     ; Swap register set
             JR      L1D8CHL          ; Done with routine

__FlipUD      LD      HL,(VDP_Temp)    ; DE = temp
             LD      BC,0008H         ; HL = temp + 8

```

```

LD      E,L
LD      D,H
ADD     HL,BC
EX      DE,HL
CALL    DoFlipUD      ; Flip the image
CALL    L1E72          ; Copy the image back to VRAM
CALL    L1E5D          ; Check for hi-res graphics
JR      Z,L1D8CEXX     ; Exit if not hi-res graphics
CALL    L1E89          ; Read source color table entry
LD      HL,(VDP_Temp)  ; HL = temp
LD      BC,0008H       ; DE = temp + 8
LD      E,L
LD      D,H
ADD     HL,BC
EX      DE,HL
CALL    DoFlipUD      ; Flip the color table
CALL    L1E9A          ; Write dest color table entry
L1D8CEXX EXX           ; Swap register set
L1D8CHL INC    HL      ; Increment dest index
L1D8C   INC    DE      ; Increment VDP table index
DEC     BC            ; Decrement count
LD      A,B           ; Check for count = 0
OR      C
EXX     ; Swap register set
JR      NZ,L1D74      ; Go back until count = 0
POP     IX            ; Clean up stack
RET

__Expand LD      HL,(VDP_Temp)  ; HL = temp
LD      BC,0008H       ; DE = temp + 8
LD      E,L
LD      D,H
ADD     HL,BC
EX      DE,HL
CALL    DoExpand      ; Expand the image
EX      AF,AF'        ; A = VDP table code
PUSH    AF
EX      AF,AF'
POP     AF
EXX     ; DE = dest index
PUSH    HL
EXX
POP     DE
LD      HL,(VDP_Temp)  ; HL = temp + 8
LD      BC,0008H
ADD     HL,BC
LD      IY,0004H       ; Write 4 entries
CALL    _BlkWrtVRAM    ; to the VRAM table
CALL    L1E5D          ; Check for hi-res graphics
JR      Z,L1E55        ; Exit if not hi-res graphics
CALL    L1E89          ; Read source color table entry
LD      HL,(VDP_Temp)  ; HL = temp
LD      BC,0008H       ; DE = temp + 8
LD      E,L
LD      D,H

```

```

      ADD    HL,BC
      EX     DE,HL
      CALL   DoExpandCT      ; Expand the color table
      LD     A,04H           ; A = VDP color table code
      EXX                    ; DE = dest index
      PUSH   HL
      EXX
      POP    DE
      LD     HL,(VDP_Temp)    ; HL = temp + 8
      LD     BC,0008H
      ADD    HL,BC
      LD     IY,0004H         ; Write 4 entries back
      CALL   _BlkWrtVRAM
L1E55  EXX                    ; Swap register set
      INC    HL               ; Increment dest index
      INC    HL               ;   four times
      INC    HL               ;
      JR     L1D8CHL          ; Done with routine

L1E5D  EX     AF,AF'          ; A = VDP table code
      PUSH   AF
      EX     AF,AF'
      POP    AF
      CP     03H              ; Pattern generator table?
      JP     NZ,L10E6          ; Return zero if not
      LD     HL,VDP0Shad      ; Check for hi-res mode
      XOR    A                ; Prepare to return zero
      BIT    1,(HL)
      RET    Z                ; Return zero if not
      INC    A                ; Return 1 if in hi-res mode
      RET                    ; (Z-flag set according to A-reg)

L1E72  EX     AF,AF'          ; A = VDP table code
      PUSH   AF
      EX     AF,AF'
      POP    AF
      EXX                    ; DE = dest index
      PUSH   HL
      EXX
      POP    DE
      LD     HL,(VDP_Temp)    ; HL = temp + 8
      LD     BC,0008H
      ADD    HL,BC
      JR     L1EA7

L1E89  EXX                    ; DE = source index
      PUSH   DE
      EXX
      POP    DE
      LD     HL,(VDP_Temp)    ; HL = temp
L1E90  LD     A,04H           ; A = VDP color table code
L1E92  LD     IY,0001H        ; Read one entry
      JP     _BlkReadVRAM     ;   from the color table

L1E9A  EXX                    ; DE = dest index

```

```

        PUSH    HL
        EXX
        POP     DE
L1EA5   LD      HL,(VDP_Temp)    ; HL = temp
L1EA7   LD      A,04H           ; A = VDP color table code
        LD      IY,0001H        ; Write one entry
        JP      _BlkWrtVRAM     ; to the color table

DoExpand
        PUSH    HL              ; IX = source
        POP     IX
        PUSH    DE              ; IY = dest
        POP     IY
L1EB4   LD      B,08H           ; Initialize count
        LD      A,(IX+00H)      ; Get source byte
        INC     IX
        LD      D,A             ; Copy source byte into D
        LD      E,04H           ; Initialize bit count
L1EBC   RL      A               ; Shift source bit into H
        RL      H
        RL      D               ; Shift copy of bit into H
        RL      H
        DEC     E               ; Loop for four bits
        JR      NZ,L1EBC
L1EC9   LD      E,04H           ; Initialize bit count
        RL      A               ; Shift source bit into L
        RL      L
        RL      D               ; Shift copy of bit into L
        RL      L
        DEC     E               ; Loop for four bits
        JR      NZ,L1EC9
        LD      (IY+00H),H      ; Store bits in first row
        LD      (IY+10H),L
        INC     IY
        LD      (IY+00H),H      ; Store bits in second row
        LD      (IY+10H),L
        INC     IY
        DJNZ    L1EB4           ; Loop until done
        RET

DoExpandCT
L1EEE   LD      B,10H           ; Initialize count
        PUSH    HL              ; Save source addr
        LD      A,(HL)          ; Get next source byte
        INC     HL
        LD      (DE),A          ; Store it twice
        INC     DE
        LD      (DE),A
        INC     DE
        DEC     BC              ; Decrement count
        LD      A,B
        CP      08H             ; If count = 8
        JR      NZ,L1EFB
        POP     HL              ; get back source addr
L1EFB   DJNZ    L1EEE           ; Loop until done
        RET

```

```

DoFlipLR      LD      B,08H          ; Initialize count
L1F03         LD      C,(HL)         ; Get next byte
              LD      A,80H          ; Set terminator bit
L1F06         RL      C              ; Shift a bit out
              RRA                    ; Shift it in
              JR      NC,L1F06        ; Loop until done
              LD      (DE),A          ; Save byte
              INC     HL              ; Point to next bytes
              INC     DE
              DJNZ    L1F03           ; Loop until done
              RET

DoRotate      PUSH    HL              ; IX = source
              POP     IX
              EX      DE,HL           ; HL = destination
              LD      B,08H          ; Initialize count
L1F19         RL      (IX+00H)         ; Shift a column of bits from the
              RR      (HL)            ;   source into a row of bits in
              RL      (IX+01H)         ;   the destination
              RR      (HL)
              RL      (IX+02H)
              RR      (HL)
              RL      (IX+03H)
              RR      (HL)
              RL      (IX+04H)
              RR      (HL)
              RL      (IX+05H)
              RR      (HL)
              RL      (IX+06H)
              RR      (HL)
              RL      (IX+07H)
              RR      (HL)
              INC     HL              ; Increment destination
              DJNZ    L1F19           ; Loop until done
              RET

DoFlipUD      LD      BC,0008H        ; Dest = dest + 8
              ADD     HL,BC
              LD      B,C              ; Initialize count = 8
L1F53         DEC     HL              ; Decrement source
              LD      A,(HL)          ; Get source byte
              LD      (DE),A          ; Store in destination
              INC     DE              ; Increment destination
              DJNZ    L1F53           ; Loop until done
              RET

```

```

;*****
;

```

```

;      1F79      ReadCtl
;

```

```

; Read a joystick or keypad controller and a fire button
;

```

```

; ENTRY H = 0 for left control, 1 for right
;

```

```

;      L = 0 for joystick/left fire, 1 for keypad/right fire
;

```

```

; EXIT: H = fire button in 40H bit
;

```

```

;      L = joystick directionals or key code

```

```

;      E = old pulse counter (only if L=0)
;*****
_ReadCtl      LD      A,L          ; Check if reading keypad
              CP      01H
              JR      Z,L11AA      ; Branch if reading keypad
              LD      BC,PulseCnt1 ; Point BC to pulse counter
              LD      A,H
              OR      A
              JR      Z,L1199      ; branch if left controller
              INC     BC           ; Point BC to PulseCnt2
L1199         LD      A,(BC)       ; E = old pulse counter value
              LD      E,A
              XOR     A           ; Clear pulse counter
              LD      (BC),A
              CALL    L113D       ; Read joystick port
              LD      D,A         ; Save port bits in D
              AND     0FH
              LD      L,A         ; L = directional bits
              JR      L11BC

;      Read the keypad

L11AA         OUT     (IO_KP_Select),A ; Select keypad mode
              CALL    L113D       ; Read joystick port
              LD      D,A         ; Save port bits in D
              OUT     (IO_Joy_Select),A ; Select joystick mode
              AND     0FH         ; Mask off keypad bits
              LD      HL,Keypad_Table ; Index into keypad table
              LD      B,00H
              LD      C,A
              ADD     HL,BC
              LD      L,(HL)      ; L = key code (or 0FH if none)
L11BC         LD      A,D
              AND     40H
              LD      H,A         ; H = right fire button bit
              RET

```

```

; Jump vectors. Please call these instead of using addresses
; within the ROM itself. Since the code is so sloppy that
; hundreds of bytes can be squeezed out of it, it's too tempting
; to reassemble an optimized ROM with room to add something cool.

```

```

A1F61
DoSound      JP      _DoSound
L1F64        JP      L0488 ; video P04A3
L1F67        JP      L06C7 ; video P06D8
FlipRL       JP      _FlipRL
FlipUD       JP      _FlipUD
Rotate       JP      _Rotate
Expand       JP      _Expand
ReadCtlRaw   JP      _ReadCtlRaw
ReadCtl      JR      _ReadCtl
DB 0
SkillScrn    JP      _SkillScrn
InitFont     JP      _InitFont

```

FillVRAM	JP	_FillVRAM
InitVDP	JP	_InitVDP
ReadSpinner	JP	_ReadSpinner
PBaseLoad	JP	_PBaseLoad
PBlkReadVRAM	JP	_PBlkReadVRAM
PBlkWrtVRAM	JP	_PBlkWrtVRAM
PInitRAMSprt	JP	_PInitRAMSprt
PCopyRAMSprt	JP	_PCopyRAMSprt
PInitTimers	JP	_PInitTimers
PStopTimer	JP	_PStopTimer
PStartTimer	JP	_PStartTimer
PTestTimer	JP	_PTestTimer
PWriteReg	JP	_PWriteReg
PWrtVRAM	JP	_PWrtVRAM
PReadVRAM	JP	_PReadVRAM
L1FAF	JP	L0655 ; video P0664
PInitSound	JP	_PInitSound
PAddSound	JP	_PAddSound
BaseLoad	JP	_BaseLoad
BlkReadVRAM	JP	_BlkReadVRAM
BlkWrtVRAM	JP	_BlkWrtVRAM
InitRAMSprt	JP	_InitRAMSprt
CopyRAMSprt	JP	_CopyRAMSprt
InitTimers	JP	_InitTimers
StopTimer	JP	_StopTimer
StartTimer	JP	_StartTimer
TestTimer	JP	_TestTimer
RunTimers	JP	_RunTimers
NoSound	JP	_NoSound
WriteReg	JP	_WriteReg
VDP_Status	JP	_VDP_Status
WrtVRAM	JP	_WrtVRAM
ReadVRAM	JP	_ReadVRAM
L1FE5	JP	L0664 ; video
L1FE8	JP	L0679 ; video (no P)
ReadCtlState	JP	_ReadCtlState
InitSound	JP	_InitSound
AddSound	JP	_AddSound
UpdateSound	JP	_UpdateSound
L1FF7	JP	L04A3 ; video
L1FFA	JP	L06D8 ; video
Random	JP	_Random
D2000	END	