Sega Genesis Reference Sheets

by

Richard Seaborne

11/05/92

Extracted from Sega's "Genesis Software Manual, 02/06/90."

VRAM WRITE: **(VIDEO RAM)**

1st:	Write to $C00004:	%01AA_AAAA_AAAA_AAA0,	where A=Even VRAM Address LOW portion
2nd:	Write to $C00004:	%0000_0000_0000_00AA,	where A= Even VRAM Address HIGH portion
3rd:	Write to $C00000:	%DDDD_DDDD_DDDD_DDDD,	where D=VRAM DATA 32-Bits

VRAM READ: **(VIDEO RAM)**

1st:	Write to $C00004:	%00AA_AAAA_AAAA_AAA0,	where A=Even VRAM Address LOW portion
2nd:	Write to $C00004:	%0000_0000_0000_00AA,	where A= Even VRAM AddressHIGH portion
3rd:	Read f/ $C00000:	%DDDD_DDDD_DDDD_DDDD,	where D=VRAM DATA 32-Bits

CRAM WRITE: **(COLOR RAM)**

1st:	Write to $C00004:	%1100_0000_0AAA_AAAA,	where A=Even CRAM Address Range=0-127
2nd:	Write to $C00004:	%0000_0000_0000_0000	
3rd:	Write to $C00000:	%0000_BBB0_GGG0_RRR0,	where B=Blue, G=Green, R=Red

CRAM READ: **(COLOR RAM)**

1st:	Write to $C00004:	%0000_0000_0AAA_AAAA,	where A=Even CRAM Address, Range=0-127
2nd:	Write to $C00004:	%0000_0000_0010_0000	
3rd:	Read f/ $C00000:	%0000_BBB0_GGG0_RRR0,	where B=Blue, G=Green, R=Red

VSRAM WRITE: **(VERTICAL SCROLL RAM)**

1st:	Write to $C00004:	%0100_0000_0AAA_AAAA,	where A=Even CRAM Address , Range=0-79
2nd:	Write to $C00004:	%0000_0000_0001_0000	
3rd:	Write to $C00000:	%0000_0VVV_VVVV_VVVV,	where V=Vertical Scroll Position

VSRAM READ: **(VERTICAL SCROLL RAM)**

1st:	Write to $C00004:	%0000_0000_0AAA_AAAA,	where A=Even CRAM Address , Range=0-79
2nd:	Write to $C00004:	%0000_0000_0001_0000	
3rd:	Read f/ $C00000:	%0000_0VVV_VVVV_VVVV,	where V=Vertical Scroll Position

DMA ROM TO VRAM: **(DIRECT MEMORY ACCESS)**

1. Valid Source Address Range=$000000 - $3FFFFF AND $FF0000 - $FFFFFF
2. ROM DMA requires final DMA to be done from RAM, e.g. instruction following DMA should be "MOVE.W $FF0000,$C00004."
3. Tengen experience suggests DMA cannot cross a 64Kbyte boundary.

Setting DMA:

1st: Enable DMA through Register #1 (see Register Descriptions for DETAILS)
OR the RAM SHADOW of Register #1 with #%0001_0000 and write value to Reg. #1

2nd: Set Auto-Increment Amount through Register #15
Write # of bytes to be xferred per DMA address update, usually 2

3rd: Set total # of words to be sent to VRAM via DMA in Registers #19 and #20
Write LOW byte of DMA XFER LENGTH to Register #19
Write HIGH byte of DMA XFER LENGTH to Register #20

4th: Set DMA Mode and Source Address through Registers #21, #22, and #23
Write LOW byte of source ROM/RAM address to Register #21
Write MIDDLE byte of source ROM/RAM address to Register #22
OR DMA type with the HIGH byte of source ROM/RAM address with DMA MODE bits 6 and 7. Write result to

Register #23
DMA MODES:

%0A = ROM/RAM to VRAM, bit 6 serves as portion of source ROM/RAM address
%10 = VRAM FILL
%11 = VRAM TO VRAM

5th: Set Destination VRAM Address
Use first two steps for normal **VRAM WRITE** (see above).
1st: Write to $C00004: %01AA_AAAA_AAAA_AAA0, where A=Even VRAM Address LOW portion
2nd: OR bit 7 w/ #1 to set it for DMA Mode then do 3rd step.
3rd: Write to $C00004: %0000_0000_0000_00AA, where A= Even VRAM Address HIGH portion
6th: Disable DMA through Register #1
AND the RAM SHADOW of register #1 with #%1110_1111 and write value to Reg. #1

Video Display Processor (VDP) Write-Only Registers

Register #0: %000H_01M0 **ModeSet1**
H: 1=Enable Horizontal Interrupt (68000 Level 4), see Register #10 for Scanlines until interrupt (aka what raster line to interrupt on)
0=Disable Horizontal Interrupt
M: 1=H,V Counter Stop
0=Enable read H,V Counter

Register #1: %0DVM_W100 **ModeSet2**
D: 1=Enable Display (Blanking OFF)
0=Disable Display (Blanking ON)
V: 1=Enable Vertical Blank Interrupt (68000 Level 6), VBI ON
0=Disable Vertical Blank Interruppt, VBI OFF
M: 1=DMA Enable
0=DMA Disable
W: 1=V30 Cell Mode (PAL)
0=V28 Cell Mode (NTSC), ALWAYS USE THIS FOR U.S.

Register #2: %00AAA000 **MapABase**
A: Scroll A BAT address in VRAM = %AAA0_0000_0000_0000 = $A000, where A=0-E

Register #3: %00AAAAB0 **WindowBase**
A: Window BAT address in VRAM = %AAAA_0000_0000_0000 = $A000, where A=0-F
B: Not applicable for HIRES Genesis Mode, so discard. This is additional resolution level for Masters System Compatibility.

Register #4: %0000_0AAA **MapBBase**
A: Scroll B BAT address in VRAM = %AAA0_0000_0000_0000 = $A000, where A=0-E

Register #5: %0AAAAAAB **SpriteBase**
A: Sprite Attribute Table (SAT) address in VRAM = %AAAA_AA00_0000_0000 = $AA00, where A=0-3F
B: Not applicable for HIRES Genesis Mode, so discard. This is additional resolution level for Masters System Compatibility.

Register #6: %00000000 **0**
All zeroes

Register #7: %00PPCCCC **BkgrndColor**

P: Background Default Color Palette Number

C: Background Default Color Number

Register #8: %00000000 **0**

All zeroes

Register #9: %00000000 **0**

All zeroes

Register #10: %RRRRRRRR **Hint**

R: # of Raster lines until interrupt, following the VBLANK. AKA what scanline to interrupt on.

Register #11: %0000EVSS **ModeSet3**

E: 1=Enable External Interrupt

0=Disable External Interrupt

V: 1=2-Cell Vertical Scroll Mode

0=Full Vertical Scroll Mode

S: %00=Full Horizontal Scroll Mode

%01=N/A

%10=1-Cell Horizontal Scroll Mode

%11=Every Scan line Horizontal Scroll

Register #12: %H000SLLW **ModeSet4**

H: 1=40-Cell Wide Screen Mode (always use this)

0=32-Cell Wide Screen Mode (this is for Master System Compatibility)

S: 1=Enable Shadow & Hilight

0=Disable Shadow & Hilight

W: SHOULD BE SAME AS "H" above

1=40-Cell Wide Screen Mode (always use this)

0=32-Cell Wide Screen Mode (this is for Master System Compatibility)

L: %00=No Interlace Mode

%01=Interlace Mode

%10=N/A

%11=Full Interlace (Double Resolution)

Register #13: %00AAAAAA **HScrollBase**

A: Horizontal Scroll Table Address = %AAAA_AA00_0000_0000 = $AA00, where A = 0-3F

Register #14: %00000000 **0**

All zeroes

Register #15: %NNNNNNNN **AutoIncrement**

N: Automatic Increment amount after RAM accesses, N=0-255

Register #16: %00VV00HH **ScrollSize**

V: %00=32-Cell Tall BAT
%01=64-Cell Tall BAT
%10=N/A
%11=128-Cell Tall BAT

H: %00=32-Cell Wide BAT
%01=64-Cell Wide BAT
%10=N/A
%11=128-Cell Wide BAT

*NOTE: These values AFFECT BOTH SCROLLA AND SCROLLB

Register #17: %R00WWWWW **WindowHPos**

***WINDOW has is drawn in place of ScrollA**

R: 1=Window is in from left side from base point
0=Window is in from right side from base point

W: # of cells Scrolled in, normally 0 ....40 if window is shown

Register #18: %D00VVVVV **WindowVPos**

***WINDOW has is drawn in place of ScrollA**

D: 1=Window is in lower side from base point
0=Window is in upper side from base point

V: # of cells Scrolled down, normally 0 ....28 if window is shown

Register #19: DMA Length LOW BYTE **DMALengthLO**

Register #20: DMA Length HIGH BYTE **DMALengthHI**

Register #21: DMA Source Address LOW BYTE **DMASrcAdrLO**

Register #22: DMA Source Address MIDDLE BYTE **DMASrcAdrMID**

Register #23: %DDAAAAAA **DMASrcAdrHI**

A: DMA Source Address HIGH BYTE

D: %0A=Memory to VRAM Mode, A is used as additional addressing bit for high byte address
%10=VRAM FILL Mode
%11=VRAM TO VRAM Mode

BAT Cell Data Format
(aka Background Attribute Table)

BAT WORD: %PCCVHNNNNNNNNNN

P: 1=Priority over sprites
0=Not Priority over sprites

C: Color Palette used by cell, C=0-3

V: 1=Vertically Flipped Character
0=Not Vertically Flipped

H: 1=Horizontally Flipped Character
0=Not Horizontally Flipped

N: Character # used for this position. Actual VRAM address of character fetched is calculated as follows: VRAM ADDRESS = Character # X 32. Conversely, Character # = VRAM Address / 32. Each character requires 32 bytes to define.

Cell/Tile/Character Definition

TILE STRUCTURE: BBBBBBBB
BBBBBBBB
BBBBBBBB
BBBBBBBB
BBBBBBBB
BBBBBBBB
BBBBBBBB
BBBBBBBB

Where each B=1 Nibble (half byte) that defines what color # this pixel is
The top-left most B is pixel #0 and the bottom-right most B is pixel #63
The data is stored consecutively in VRAM, AKA LINEAR PIXELPACK Format
Each pixel row requires 8 pixels which is 4 bytes. 8 rows = 32 bytes

1 pixel = 1 nibble
Character Definition is 8X8 PixelPack 2-D Matrix
Each pixel defines what color it is, 0-15, from the BAT specified palette (see above)

SAT Cell Data Format
(aka Sprite Attribute Table)

SAT WORD 0: %000000YYYYYYYYYY
SAT WORD 1: %0000WWTT0LLLLLLL
SAT WORD 2: %PCCVHNNNNNNNNNN
SAT WORD 3: %0000000XXXXXXXXX

Y: Vertical Position of Sprite (aka Y-Coordinate)

X: Horizontal Position of Sprite (aka X-Coordinate)

L: Link Data, where L=0-79. Specifies what sprite should be drawn after this one is drawn. The list is terminated with a #0.

P: 1=Priority over background (appears in front of BG cells)
0=Not Priority over background (appears behind BG cells)

C: Color Palette used by sprite, where C=0-3

V: 1=Verticall Flipped Sprite
0=Not Verticall Flipped

H: 1=Horizontally Flipped Sprite
0=Not Horizontally Flipped

N: Character # Sprite Starts Using, where N is a multiple of 32 bytes (character size).
VRAM Address Sprite Data Starts At = Character # X 32
Character # = VRAM Address Spriite Data Starts At / 32

W: %00=1-Cell Wide Sprite (8 pixel wide sprite)
%01=2-Cell Wide Sprite (16 pixel wide sprite)
%10=3-Cell Wide Sprite (24 pixel wide sprite)
%11=4-Cell Wide Sprite (32 pixel wide sprite)

T: %00=1-Cell Tall Sprite (8 pixel tall sprite)
%01=2-Cell Tall Sprite (16 pixel tall sprite)
%10=3-Cell Tall Sprite (24 pixel tall sprite)
%11=4-Cell Tall Sprite (32 pixel tall sprite)

Sprite Limitations

Max. 20 sprites per scanline
Max. 80 sprite handles
X-Coordinate Range = 1-$1FF (1-511)
Y-Coordinate Range = 0-$1FF (0-511)
Visible BG Screen's 0,0 is at Sprite 128,128. Therefore, add 128 to all sprite coordinates to make them on the screen

Sprite Cell/Tile Definition

Although each sprite is made up of 8X8 character cells that are identical to the BG TILE STRUCTURE (described above), sprites can be of different sizes. The different sized sprites still use up to 16 of the 8X8 pixel characters. The sprite character address in VRAM really points to the first character used by the sprite. The sprite will automatically use the appropriate number of characters following the first character to define itself. It builds itself in vertical columns, using one character each time from contiguous VRAM. The below chart should make this clear.

1X1 Sprite: 0	2X1 Sprite: 01	3X1 Sprite: 012	4X1 Sprite: 0123
1X2 Sprite: 0 1	2X2 Sprite: 02 13	3X2 Sprite: 024 135	4X2 Sprite: 0246 1357
1X3 Sprite: 0 1 2	2X3 Sprite: 03 14 25	3X3 Sprite: 036 147 258	4X3 Sprite: 0369 147A 258B
1X4 Sprite: 0 1 2 3	2X4 Sprite: 04 15 26 37	3X4 Sprite: 048 159 26A 37B	4X4 Sprite: 048C 159D 26AE 37BF

Color Palette Data Format

CRAM WORD: %0000BBB0GGG0RRR0

B: Blue, where B=0-7

G: Green, where G=0-7

R: Red, where R=0-7

There are 4 color palettes definied in the Genesis Color RAM (CRAM); each is defined right after the previous one.

CRAM Palette #0 Address: $00

CRAM Palette #1 Address: $20

CRAM Palette #2 Address: $40

CRAM Palette #3 Address: $60

end of CRAM: $7F

Joystick

Joystick Byte: %SACBRLDU

S: Button START

A: Button A

C: Button C

B: Button B

R: Right Direction Pad

L: Left Direction Pad

D: Down Direction Pad

U: Up Direction Pad

VDP STATUS REGISTER

Reading $C0004 - $C0005 as a word yields the status if the Video Display Processor.

VDP STATUS: %000000EFISOVHDP

- E: 1=Write FIFO empty
- F: 1=Write FIFO full
- I: 1=Vertical Interrupt Happened
- S: 1=Sprite Overflow occured - > 21 sprites on a scanline in Mode V (40 cell wide mode)
- C: 1=Collision happened between non-zero pixels in sprites. USE to decide if a collision check should be processed.
- O: 1=Odd Frame in Interlace Mode
  0=Even Frame in Interlace Mode
- V: 1=Currently in Vertical Blank
- H: 1=Currently in Horizontal Blank
- D: 1=DMA Busy
- P: 1=PAL Mode
  0 = NTSC Mode (U.S. Mode)

Normal Sega Genesis Settings

Mode V Video:

BAT: 320 X 224 (40 characters X 28 characters) screen built from 8X8 pixel character cells
2 Scroll Fields, A and B.
Virtual Screen Size can be:

0. 32X32 characters
1. 32X64 characters
2. 32X128 characters
3. 64X32 characters * MINIMUM NORMAL MODE V GENESIS RESOLUTION
4. 64X64 characters
5. 128X32 characters

SAT: 80 sprite handles

Priority: Without Priority bits overriding default, SPRITE > SCROLLA > SCROLLB

ROM for Program and Data: $000000 - $3FFFFF, max. 4 Megabytes (32 Megabits)
RAM for Run-Time Data: $FF0000 - $FFFFFF, max. 64Kilobytes (512 Kilobits)
Video Display Processor (VDP): $C00000 - $DFFFFF use to access 64Kbytes of VRAM and additional special RAM (CRAM,VSRAM,etc.)

- Data: $C00000 - $C00003
- Control: $C00004 - $C00007
- HV Counter: $C00008 - $C00009
- PSG: $C00010 - $C00011 , ONLY HIGH BYTE IS VALID (bits 8-F)

System I/O: $A00000 - $AFFFFF

- Z80: $A00000 - $A0FFFF
- Sound RAM: $A00000 - $A01FFF
- YM2612 Sound Chip: $A04000 - $A05FFF
  - A0,D0: $A04000 - $A04001
  - A1,D1: $A04002 - $A04003
- Bank Register: $A06000
- I/O: $A10000 - $A10FFF
- Version #: $A10000 - $A10001, %MVDREEEE
  - M: 0=Domestic,1=Overseas
  - V: 0=NTSC Video Mode,1=PAL Video Mode
  - D: 0=No Disk Connected,1=Disk Connected
  - R: RSV=N/A
  - E: Version #, should be #0
- Data (CTRL1): $A10002 - $A10003
- Data (CTRL2): $A10004 - $A10005
- Data (EXP): $A10006 - $A10007
- Control 1: $A10008 - $A10009
- Control 2: $A1000A - $A1000B
- Control E: $A1000C - $A1000D
- TxData 1: $A1000E - $A1000F
- RxData 1: $A10010 - $A10011
- S-Mode 1: $A10012 - $A10013
- TxData 2: $A10014 - $A10015
- RxData 2: $A10016 - $A10017
- S-Mode 2: $A10018 - $A10019
- TxData E: $A1001A - $A1001B
- RxData E: $A1001C - $A1001D
- S-Mode E: $A1001E - $A1001F
- Control: $A11000 - $A11FFF
- Memory Mode: $A11000 - $A11001
- Z80 Bus Request: $A11100 - $A11101
- Z80 Reset: $A11200 - $A11201

Z80 Memory Map:

- Sound RAM: $0000 - $1FFF
- YM2612 Sound Chip: $4000 - $4003
  - A0: $4000
  - D0: $4001
  - A1: $4002
  - D1: $4003
- Bank Register: $6000
- PSG76489: $7F11

```
*           INCLUDE sndequ.src

Z80_RAM EQU       $00A00000
Z80_BUS_REQ EQU           $00A11100           ;Z80 BUS REQUEST ADDRESS
Z80_RESET EQU   $00A11200           ;Z80 RESET
INIT_SP   EQU       $00FFFFFC           ;Initial Stack Pointer
WRAM      EQU       $00FF0000           ;Start of working RAM
*
*   VDP ports
*
VDP_DATA                EQU     $C00000
VDP_STATUS      EQU     $C00004
VDP_REGSET      EQU     $C00004
VDP_ADRSET      EQU     $C00004
VDP_HVCNT       EQU     $C00008
VDP_PSG                 EQU     $C00011
*
*   VDP registers
*
VDP_REG0                EQU     $0000
VDP_REG1                EQU     $0100
VDP_REG2                EQU     $0200
VDP_REG3                EQU     $0300
VDP_REG4                EQU     $0400
VDP_REG5                EQU     $0500
VDP_REG6                EQU     $0600
VDP_REG7                EQU     $0700
VDP_REG8                EQU     $0800
VDP_REG9                EQU     $0900
VDP_REG10       EQU     $0A00
VDP_REG11       EQU     $0B00
VDP_REG12       EQU     $0C00
VDP_REG13       EQU     $0D00
VDP_REG14       EQU     $0E00
VDP_REG15       EQU     $0F00
VDP_REG16       EQU     $1000
VDP_REG17       EQU     $1100
VDP_REG18       EQU     $1200
VDP_REG19       EQU     $1300
VDP_REG20       EQU     $1400
VDP_REG21       EQU     $1500
VDP_REG22       EQU     $1600
VDP_REG23       EQU     $1700
*
*   VDP codes
*
VDP_W_VRAM      EQU     $0001
VDP_W_CRAM      EQU     $0003
VDP_W_VSRAM     EQU     $0005
VDP_R_VRAM      EQU     $0000
VDP_R_CRAM      EQU     $0008
```

```
VDP_R_VSRAM     EQU     $0004
VDP_DMA                 EQU     $0020
*
*   VDP map
*
VDP_M_INFO      EQU     $0000                   ; playfield bitmap RAM area
VDP_L_INFO      EQU     $5500                   ; 680 stamps
VDP_M_PF0       EQU     $5500
VDP_M_PF1       EQU     $5500
VDP_L_PFS       EQU     $2800                   ; 320 stamps(cockpit area)

VDP_M_WINDOW  EQU       $B000
VDP_M_HSCROLL EQU       $F400
VDP_M_SPRITES  EQU      $F000
VDP_L_SPRITES   EQU     $0280                   ; MOBs are 80*8
VDP_M_SCROLLB EQU       $E000
VDP_M_SCROLLA EQU       $C000
*
DATA1   EQU     $00A10003
DATA2   EQU     $00A10005
DATA3   EQU     $00A10007
CTRL1   EQU     $00A10009
CTRL2   EQU     $00A1000B
CTRL3   EQU     $00A1000D
SCTRL1  EQU     $00A10013
SCTRL2  EQU     $00A10019
SCTRL3  EQU     $00A1001F
*
INTDIS  EQU     $2700   ;SR value to disable interrupts
INTEN2  EQU     $2100   ;SR value to enable EXTERN, HBLANK and VBLANK
INTEN4  EQU     $2300   ;SR value to enable HBLANK and VBLANK
INTEN6  EQU     $2500   ;SR value to enable VBLANK only
*
*   MACROS
*
        XREF    .vdp_adr_sav
        XREF    .tmp_set_adr
SET_REG         MACRO           ; reg,val
        MOVE.W #$8000+\1+\2,VDP_REGSET
        ENDM

VDP_ADR         MACRO           ; code,addr,tmp
        MOVE.W \2,\3
        ANDI.W  #$3FFF,\3
MTMP    SET     (((\1)<<14)&$C000)
        IFNE    MTMP
        ORI.W   #MTMP,\3
        ENDC
        MOVE.W \3,.vdp_adr_sav
        MOVE.W \3,VDP_ADRSET
        MOVE.W \2,\3
        LSR.W   #8,\3
        LSR.W   #6,\3
```

```
MTMP    SET     (((\1)<<2)&$00F0)
        IFNE    MTMP
        ORI.W   #MTMP,\3
        ENDC
        MOVE.W \3,.vdp_adr_sav+2
        MOVE.W \3,.tmp_set_adr
        MOVE.W .tmp_set_adr,VDP_ADRSET
        ENDM

WR_VDP MACRO            ; val
\@:     BTST    #0,VDP_STATUS
        BNE.S   \@      ; wait for FIFO to be not FULL
        MOVE.W \1,VDP_DATA
        ENDM

RD_VDP MACRO            ; dest
        MOVE.W VDP_DATA,\1
        ENDM

HWAIT   MACRO           ; An
\@:     BTST    #2,(\1) ; wait for HBLANK
        BEQ.S   \@
        ENDM

NHWAIT  MACRO           ; An
\@:     BTST    #2,(\1) ; wait for not HBLANK
        BNE.S   \@
        ENDM
```

Loading Full Game Image From Address $000000
SEGALOAD Filename

Loading Fragment Image to Specified Address:
SEGALOAD /Axxxxxx Filename

Loading Binary Image:
SEGALOAD /B Filename
SEGALOAD /Axxxxxx Filename

Getting Full Game Image From ROMulator:
SEGALOAD /E
...File is saved in WORKING DIRECTORY as SEGALOAD.BIN

Getting ROM Images From ROMulator:
1-Megabit ROMs: **SEGALOAD /1**
2-Megabit ROMs: **SEGALOAD /2**
4-megabit ROMs: **SEGALOAD /4**

Here is the program HELP printout:

SEGALOAD v2.00 (c)1992 Western Technologies, Inc.
Usage: SEGALOAD [options] filename [filename..] [options]
Where filename is the name of the file to download to the Genesis.
Valid options are:

/1 - Use LPT1 (default) /2 - Use LPT2 /3 - Use LPT3
/Annnn - Start Address of load for binary format in hex (default is 0)
/B - File is Binary format
/C - File is COFF format (default)
/D - Use a differential download (compare, and DL what is different)
/E - Extract to SEGALOAD.BIN (Max address will be 0x200000)
/G - Load a .GEN file, display it.(Kills current program in card)
/H - Hard Reset the Genesis (Use only AFTER you've done a load!!!)
/NL- No load to Genesis (When you just want to convert a file)
/NG- No GO - Will load to the Genesis, but will not perform a GO
/NT- Disable TURBO LOAD - for incompatible hardware
/P - comPare the file to the current contents of the dev card
/Rn - Extract ROM images. n is 1,2, or 4 for 1,2 and 4 meg ROMS
/S - File is Snasm's CPE file format
/T - Display firmware ver, test memory on dev card.(T8 forces 8M)
/Ynn - Yank $nn bytes of data from the card. Use /Y? for more info

The /A option may be given more than once - use to load mutiple binary files
I.E. SEGALOAD /B File1.bin /AFE18 File2.bin /A1FFF0 File3.bin
Will load file1.bin at 0, File2.bin at 0xFE18 and file3.bin at 0xFFFF0, note that the address and filetype must appear BEFORE the filename.

GENESIS INITIALIZATION PROGRAM
INIT.SRC

```
        INCLUDE genesis.inc
*
        xdef    cold_start      ;What to call to reset entire machine state
    xdef  hot_start     ;Routine game code can call for a WARM/HOT restart
    xref  MAIN
*
*
*
*  :::::::::::: HARDWARE VECTORS :::::::::
*
        SECTION SEGAGENESISSTART
*
        dc.l    INIT_SP  ;stack pointer
        dc.l    reset    ;reset
        dc.l    buserr
        dc.l    adrerr
        dc.l    illins

        dc.l    zerdiv
        dc.l    chkins
        dc.l    trapv
        dc.l    privio
        dc.l    tracei
        dc.l    l1010e
        dc.l    l1111e
        dc.l    reserv
        dc.l    reserv
        dc.l    forerr
        dc.l    unintv
        dc.l    reserv   ;vector 16.
        dc.l    reserv
        dc.l    reserv
        dc.l    reserv
        dc.l    reserv
        dc.l    reserv
        dc.l    reserv
        dc.l    reserv
        dc.l    spuint   ;must start at $60
        dc.l    int1
        dc.l    exint
        dc.l    int3
        dc.l    hbint
        dc.l    int5
        dc.l    vbint
        dc.l    int7
        dc.l    trap     ;vector 32., trap 0
        dc.l    trap
        dc.l    trap
        dc.l    trap
        dc.l    trap
```

```
                dc.l    trap
                dc.l    trap
                dc.l    trap
                dc.l    trap
                dc.l    trap
                dc.l    trap
                dc.l    trap
                dc.l    trap        ;vector 44., trap 0c
                dc.l    trap
                dc.l    trap
                dc.l    trap
                dc.l    reserv      ;vector 48.
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
                dc.l    reserv
*
*GENESISSTARTCODE
*
                dc.b    'SEGA GENESIS    '
                dc.b    '(C) TENGEN 1990 '
.10:            dc.b    'RBI 3 Baseball'
                ds.b    $30-(*-.10)
.20:            dc.b    'RBI 3 Baseball'
                ds.b    $30-(*-.20)
                dc.b    'GM T-48096 -00'
                dc.w    $0000       ; checksum
                dc.b    'OJ'
                ds.b    14
                dc.l    $00000000,$0007FFFF
                dc.l    $00FF0000,$00FFFFFF
                dc.b    '           ' ; ext ram
                dc.b    '           ' ; modem
                ds.b    40
                dc.b    'UJ '       ; release ok in JAPAN,USA,EUROPE
                ds.b    13
*
__main:
reset:
************************************************
* the Sega sacred code comes first
************************************************
```

```
                tst.l       $a10008  ; POWER ON check controller A,B
                bne.s       h_s
                tst.w       $a1000c  ; POWER ON check controller C
h_s:
                bne         hot_start

cold_start:
                lea.l       reg_set(pc),a5              ; initial data address set
                movem.w (a5)+,d5-d7                     ; register initialize d5-d7.word
                movem.l  (a5)+,a0-a4                    ; register initialize a0-a4.long

security:
                move.b    -$10ff(a1),d0                     ; ** a1 = $a11100 **
                andi.b      #$000f,d0             ;  -$10ff(a1) = $a10000
                beq.s       japan                           ; Ver. No. check
                move.l    #'SEGA',$2f00(a1)   ; "SEGA" moved to SECURITY part
* 2f00(a1) = $a14000
japan:
                move        (a4),d0                         ; vdp status dummy read
                moveq       #0,d0
                move.l      d0,a6
                move.l      a6,USP                          ; user stack pointer set
                moveq       #23,d1                          ; d1 = counter

*------<<< VDP REG. initialize >>>------
*    d5 = $8000 / d7 = $100

r_int1:
                move.b    (a5)+,d5              ; vdp reg. 0-23 set
                move.w    d5,(a4)                           ;    (DMA fill set)
                add         d7,d5
                dbra        d1,r_int1

*------<<< DMA FILL >>>---------

dma_fill:
* already sat reg. #18,19,23
                move.l      (a5)+,(a4)
                move        d0,(a3)                         ; set fill data ($0) to $c00000

*-------<<< Z80 initialize >>>----
*               a0 = $a00000 / a1 = $a11100 / a2 = $a11200
*               d0 = $0      / d7 = $100

z80_clr:
                move        d7,(a1)                         ; z80_busreq on
                move        d7,(a2)                         ; z80_reset off
z801:
                btst        d0,(a1)                         ; z80_bgack ok ?
                bne.s       z801

                moveq       #37,d2                          ; counter set
z802:
```

```
                move.b    (a5)+,(a0)+           ; z80 program set to z80 sram
                dbra      d2,z802

                move      d0,(a2)               ; z80_reset on
                move      d0,(a1)               ; z80_busreq off
                move      d7,(a2)               ; z80_reset off ( z80 start )

*--------<<< work ram clear >>>------------
* a6 = $0 / d0 = $0 / d6 = $3fff

clr_wk:
                move.l    d0,-(a6)
                dbra      d6,clr_wk

*-------<<< VDP color clear >>>-------------
* a3 = $c00000 / a4 = $c00004 / d0 = $0

clr_col:
                move.l    (a5)+,(a4)          ; vdp reg #1 = 4, #15 = 2
                move.l    (a5)+,(a4)
                moveq     #$1f,d3                       ; d3 = color ram size-1
c_col1:
                move.l    d0,(a3)
                dbra      d3,c_col1

* -------<<< V SCROLL clear >>>--------------
* a3 = $c00000 / a4 = $c00004 / d0 = $0

clr_vsc:
                move.l    (a5)+,(a4)
                moveq     #19,d4                        ; d4 = vscroll ram size-1
c_vsc1:
                move.l    d0,(a3)
                dbra      d4,c_vsc1

*---------<<< PSG clear >>>-------------------
* a3 = $c00000 / a4 = $c00004 / d0 = $0
* a5 = psg_dat (clear data) point

clr_psg:
                moveq     #3,d5                         ; counter set
c_psg1:
                move.b    (a5)+,$11(a3)
                dbra      d5,c_psg1

*----------<<< init registers >>>------------

                move      d0,(a2)     ; z80 reset
                movem.l   (a6),d0-d7/a0-a6      ; initialize all registers
                move      #$2700,sr


hot_start:
```

```
                bra.s       CHECK_VDP

*******************************************************
*               tables

reg_set:
* registers set data table
                dc.w        $008000,$003fff,$000100                         ; d5 / d6 / d7
                dc.l        $a00000,$a11100,$a11200,$c00000     ; a0 - a3
                dc.l        $c00004                                         ; a4
vreg_dt:
                dc.b        $04,$14,$30,$3c,$07,$6c,$00,$00
                dc.b        $00,$00,$ff,$00,$81,$37,$00,$01
                dc.b        $01,$00,$00,$ff,$ff,$00,$00,$80
dma_fill_mode:
                dc.l        $40000080           ;vdp_reg set data.dma_fill_mod
z80_reg:
                dc.b        $af         ; xor       a
                dc.b        $01,$d9,$1f         ; ld        bc,1fd9h
                dc.b        $11,$27,$00         ; ld        de,0027h
                dc.b        $21,$26,$00         ; ld        hl,0026h
                dc.b        $f9         ; ld        sp,hl
                dc.b        $77         ; ld        (hl),a
                dc.b        $ed,$b0     ; ldir
                dc.b        $dd,$e1     ; pop       ix
                dc.b        $fd,$e1     ; pop       iy
                dc.b        $ed,$47     ; ld        i,a
                dc.b        $ed,$4f     ; ld        r,a
                dc.b        $d1         ; pop       de
                dc.b        $e1         ; pop       hl
                dc.b        $f1         ; pop       af
                dc.b        $08         ; ex        af,af
                dc.b        $d9         ; exx
                dc.b        $c1         ; pop       bc
                dc.b        $d1         ; pop       de
                dc.b        $e1         ; pop       hl
                dc.b        $f1         ; pop       af
                dc.b        $f9         ; ld        sp,hl
                dc.b        $f3         ; di
                dc.b        $ed,$56     ; im1
                dc.b        $36,$e9     ; ld        (hl),e9h
                dc.b        $e9         ; jp (hl)
new_reg_data
                dc.l        $81048f02           ; vdp reg#1=04,#15=02
clr_col_data
                dc.l        $c0000000           ;color_ram address data
clr_vsc_data
                dc.l        $40000010           ;v_scroll ram address data

psg_dat:
                dc.b        $9f,$bf,$df,$ff

CHECK_VDP:
```

```
* this is protect to push
        tst     $c00004   ; start_bottum rapid_firely.

************************************************
* now for some real code
************************************************

                move.l          #$fffffe,a7 ;initialize stack pointer

        move.w  #$100,Z80_BUS_REQ       ;z80 bus request on
        move.w  #$100,Z80_RESET         ;z80 reset off
.00021: btst    #0,Z80_BUS_REQ          ;wait for bus request confirm
        bne     .00021

                moveq           #$40,d0         ;set controllers for read
                move.b          d0,CTRL1
                move.b          d0,CTRL2
                moveq           #$00,d0
                move.b          d0,SCTRL1
                move.b          d0,SCTRL2
                moveq           #$40,d0
                move.b          d0,DATA1
                move.b          d0,DATA2

        move.w      #$2000,sr       ;enable VBLANK
        jmp             MAIN

*
*int1:  stop   #INTDIS
**int3:  stop   #INTDIS
*int5:  stop   #INTDIS
*int7:  stop   #INTDIS
*buserr:  stop   #INTDIS
*illins:  stop   #INTDIS
*zerdiv:  stop   #INTDIS
*chkins:  stop   #INTDIS
*trapv:  stop   #INTDIS
*tracei:  stop   #INTDIS
*l1010e:  stop   #INTDIS
*l1111e:  stop   #INTDIS
*reserv:  stop   #INTDIS
*forerr:  stop   #INTDIS
*unintv:  stop   #INTDIS
*privio:  stop   #INTDIS
*spuint:  stop   #INTDIS
*trap:  stop   #INTDIS
**
*adrerr: stop   #INTDIS
**
*_exit:  stop    #INTDIS
*
```

```
*
int1:
int3:
int5:
int7:
buserr:
illins:
zerdiv:
chkins:
trapv:
tracei:
l1010e:
l1111e:
reserv:
forerr:
unintv:
privio:
spuint:
trap:
*
adrerr:
*
_exit:

mark11    jmp                mark11
```

INTERMETRIX FILES NEEDED FOR ASSEMBLER/LINKER

AS.BAT File:

```
asm68000 %1.src -d -o %1.ol
```

LK.BAT File:

```
llink -v -S -c linkcmd.cmd -i infiles.opt -o rhino.ab
```

FRM.BAT File:

```
form rhino.ab -d -f c -o rhino.cof
```

LINKCMD.CMD File:

```
RESERVE (#100000 to #3FFFFF);       --Reserve Unused Area of Genesis ROM space so code overflow is detected
RESERVE (#C00000 to #DFFFFF);       --Reserve VDP I/O Area
RESERVE (#A00000 to #AFFFFF);       --Reserve Z80 Area
MEMORY (#FFFFFF);                   --Genesis Max. RAM address is $FFFFFF, 64Kbytes
LOCATE (RAM1:#FF0000);              --Genesis RAM starts at $FF0000
LOCATE (SEGAGENESISSTART:#000000);
LOCATE {CODE1:#000356);             --Sega's Genesis Start code and the 68000 Exceptions/Vectors take $356 bytes,
so                                          --code cannot start earlier than $364.
```

INFILES.OPT File:

```
init.ol
interupt.ol
main.ol
ram.ol
```

MAKEFILE File:

```
SUFFIXES = .exe    .com .ol .mac .c .l .y .ec .h .bat .src        .srm .sol
.SUFFIXES:         $(SUFFIXES)
# rules for Intermetrix Compiler/Assembler/Linker/Formatter
OBJS =  init.ol interupt.ol main.ol ram.ol
space.ab : $(OBJS) linkcmd.cmd
            llink -v -S -c linkcmd.cmd -i infiles.opt -o rhino.ab
.src.ol :
       asm68000 $< -d              -o $@
.srm.sol :
       asm68000 $< -d -ex -o $@
.c.ol :
            c68000 $< -d -o $@
```

M.BAT File:

```
make
frm
```

MAP.BAT File:

```
gsmap rhino.ab -n -o
b rhino.map
```

D.BAT File:

```
segaload rhino
```

G.BAT File:

```
call as init
call as interupt
call as main
call as ram
call lk
call frm
win
```

DO.BAT File:

```
make
frm
segaload rhino
```

RAM.SRC File:

```
* Exampe Game RAM File (Ram)
        SECTION RAM

        INCLUDE genesis.inc
    xdef temp0
    xdef temp1
    xdef temp2
    xdef temp3
    xdef temp4
    xdef temp5
    xdef temp6
    xdef temp7
    xdef temp8
    xdef temp9
    xdef temp10
    xdef temp11
    xdef temp12
    xdef temp13
    xdef temp14
    xdef temp15
    xdef counter0
    xdef counter1
    xdef counter2
    xdef counter3
    xdef index0
    xdef index1
    xdef index2
    xdef index3
temp0:   ds 4
temp1:   ds 4
temp2:   ds 4
temp3:   ds 4
temp4:   ds 4
temp5:   ds 4
temp6:   ds 4
temp7:   ds 4
temp8:   ds 4
temp9:   ds 4
temp10:  ds 4
temp11:  ds 4
temp12:  ds 4
temp13:  ds 4
temp14:  ds 4
temp15:  ds 4
counter0: ds 4
counter1: ds 4
counter2: ds 4
counter3: ds 4
index0:  ds 4
index1:  ds 4
index2:  ds 4
index3:  ds 4
```

INTERUPT.SRC File:

```
* Example Interrupt/Exception File
        SECTION CODE1
        INCLUDE genesis.inc
        XDEF vbint
        XDEF hbint
        XDEF exint

*Vertical Blank Interrupt
vbint:
        movem.l   a0-a6/d0-d7,-(sp)
        movem.l   (sp)+,a0-a6/d0-d7
        rte

*Horizontal Blank Interrupt
hbint:
        rte

*External Interrupt
exint:
        rte
```

RAM.INC File:

```
    xref temp0
    xref temp1
    xref temp2
    xref temp3
    xref temp4
    xref temp5
    xref temp6
    xref temp7
    xref temp8
    xref temp9
    xref temp10
    xref temp11
    xref temp12
    xref temp13
    xref temp14
    xref temp15
    xref counter0
    xref counter1
    xref counter2
    xref counter3
    xref index0
    xref index1
    xref index2
    xref index3
```

MAIN.SRC File:

```
* Example Game Code File (Main)
            SECTION CODE1
            INCLUDE genesis.inc
            INCLUDE ram.inc
            xdef    MAIN          ;PROGRAM ROUTINE THAT this INIT code calls when it is done
MAIN:
      bra MAIN  ;Program can NEVER truly terminate operation
```