#22

CODE MONKEYS

THE

SEGA

DEVELOPMENT

SYSTEM

CONFIDENTIAL INFORMATION

by Tim Wilson

Setting Up the Accolade Sega Development System

1. Insert the Sega System card into the Sega with the component side towards the front of the system.

2. Insert the cable as follows:
   a) DB-25 end into the PC parallel port addressed at $378 and S379, usually LPT2: if you have a monochrome card with a parallel port and a port on the motherboard. To use the port at $3BC and $3BD, you must type HOST 2 to start the program.
   b) DB-9 end into the Sega rear port.
   c) Wire with spade connector; connect to wire on system card. If you have a 2 Mb board with a Molex connector, attach the cable to the connector, taking care to observe proper polarity.

3. Install the PC software (HOST and SHOW) into a directory in your path.

4. Turn on the Sega, and run the PC program called HOST (or type HOST 2 for the other parallel port)..

5. Type **?** at the **SegaHost>** prompt to see the help screens.

The file formats for the loading commands (L, LS, and VL) are:

Code -	binary executable image (Any binary, or 2500 A.D. option X)
Data -	binary image (Any binary, or 2500 A.D. option X)
Symbols -	2500 A.D. format option S
Video -	as needed by you, no preset format required.

Operational Notes

The development system hooks into the Sega vertical blank interrupt vector, as well as the TRAP $F and TRACE vectors. A monitor/debugger resides in the upper 2K or so of the RAM card. It is protected from being overwritten by any monitor command. Except for breakpoints and tracing, the vertical blank interrupt must be enabled (command $81xx bit 5) and the interrupt mask must be $x5xx or below. You can use a semaphore to logically disable the interrupt while leaving the development system active.

The vectors are set and daisy-chained automatically by the development system, and need not be handled manually. You may load your object file at memory location zero, and the development system will automatically store your VBlank (external level 6 at $78), TRACE, and TRAP $F vectors in the monitor area and connect itself. Your vectors will be called when the monitor/debugger is inactive. However, the monitor will be unable to stop your code from overwriting the vector table at runtime, thus disabling the development system.

When displaying memory at the vector table, the monitor displays the data from the shadow area for the three vectors, as if the monitor were not present.

All commands which deal with video RAM assume that register $8F is set to word sized auto-incrementing ($8F02). If this is not the case, the video commands will produce unexpected results.

Before beginning a debugging session, you should usually execute a R (reset) command from the host to allow the stack pointer and PC to be set properly. If you need to gain control of the program from the very beginning, set a breakpoint at the first instruction, reset the Sega, and do a BW (break wait). You can then trace through the program as needed.

NOTE: **DO NOT CHANGE** the data direction register or data register of the rear port while using the development system.

A View of the SEGA Genesis System

General Features

1. 320-by-224 visible screen.
2. 16-colors per character or sprite from a palette of 512.
3. Four selectable palettes (per character or sprite).
4. All character mapped screens.
5. Two screens active with priority (overlaid) with a third non-scrolling screen.
6. Bidirectional scrolling of screens.
7. 80-sprite "clusters", each 1 to 16 characters in size.
8. 16-sprite cluster arrangements (all possible rectangular arrangements of 1 to 16 characters, 4 max. per dimension).
9. Sprites are 16-colors each, and have individually selectable palettes.
10. 68000 processor at 8 MHz for main program.
11. 64K of user RAM.
12. Custom video controller for graphics and sprites.
13. Yamaha sound controller.

Video Hardware

1. Custom video processor with 64K of internal RAM for character screens, definitions, and sprites. All video screens and objects are character-mapped.
2. Video chip can also read cartridge ROM and user RAM.
3. Both scan line and V-blank interrupts available.
4. Screen sizes (in characters) of 32 x 32, 64 x 32, and 128 x 32.
5. 16-color graphics and sprites.
6. Palette of 512 colors.
7. Video controller and it's RAM are accessed through two addresses: $C00000 and $C00004 . Commands are sent to port $C00004, and data to the other port.
8. The Z-80 is able to communicate with the video controller.
9. Word-sized character pointers.

Audio Hardware

1. The sound controller is a Yamaha YM-2612. At present, we do not have a list of its specifications.
2. A Z-80 processor with 8K of RAM resides at the $A00000 address space. The address buss can be requested by the 68000 to allow transferring of data to the Z-80.
3. The Z-80 can read and write to the 68000 ROM and RAM.

Hand Controllers

1. The controllers are read by miscellaneous hardware and the button status is returned encoded in two bytes.
2. The three ports have selectable I/O direction and state.

Programming Considerations

The current ROM size is 512K, which is equivalent of one-and-a-half IBM 360K disks. Graphics data must be compressed in order to conserve space, and the use of languages such as C may be difficult due to the limited space. The video RAM and system RAM are only 64K bytes each, which limits the number and size of graphic elements stored in their displayable (unpacked) form.

Address/Register List

000000..3FFFFF	Cartridge space
A00000..A0FFFF	Z80 address space
A02000 ?	
A10001	Unknown - but used in some software
A10003	Control Pad A
A10005	Control Pad B
A10007	Rear port
A10009	Pad A Data Direction Register
A1000B	Pad B Data Direction Register
A1000D	Rear port Data Direction Register
A1000E.W A10014.W A1001A.W	
A11100	Z80 Buss request
A11200	Z80 Reset
A14000	Unknown - but used in some software
C00000	VidDat - video chip data port
C00004	VidCom - video chip command port
0 11 → C00008 C00018	RASTER POSITION REG (WRITE 404 THEN READ)
FF0000..FFFFFF	User RAM

General Video RAM Information

The video chip has 64K bytes of RAM which is used to store all of the character data, screens, control tables, and miscellaneous data. Only the vertical scroll table and the palette information are stored in the chip registers, and therefore do not occupy any of the limited RAM.

Internally, the RAM is a contiguous 64K block, but commands from the 68000 must use a modified addressing scheme, as follows:

A video address comprises two words. The high word contains the actual address MOD $4000, while the low word contains the quotient of the actual address / $4000. To signal the video chip that you are preparing to read video data, the address is sent as is, while to prepare for writing, the high word must be added to $4000. Some examples:

Actual address	VidCom formatted address	
	Read	Write
	OFFSET IN BANK | 16K BANK	
$0000	$00000000	$40000000
$1000	$10000000	$50000000
$2345	$23450000	$63450000
$8000	$00000002	$40000002
$C123	$01230003	$41230003
$F82D	$382D0003	$782D0003

There are several video commands available to map the video RAM to the user's liking. Character screens are positioned on 8K boundaries, while other elements have smaller increments available (see following section). Character definitions (data) may be located in any otherwise unmapped or unused space within video RAM, as the character/sprite pointers can cover the entire 64K video RAM.

A high speed DMA is available for copying data from the 68000 address space into the video RAM. This process temporarily shuts down the 68000, but it is significantly faster than transmitting the data through the single data port (herein called VidDat).

VIDCOM
$0000 0000
0001 } 64K VIDEORAM
0002
0003
0010 - VSCROLL REGISTERS
0020 - PALETTE REGISTERS
0040 - HANGS ?

IGNORED
0000 0000 0000 0000
01 } 64K VIDEO RAM
10
11
01: VSCROLL REGISTERS
10: PALETTE REGISTERS
1: HANGS

Reading Video Ram

Send and address 0000 - 3FFF and bank number 0..3 to VidCom. Read word sized data at VidDat. Auto incrementing is active (see below and $8Fxx command).

Example: Read long at $D000

```
          MOVE.L    #$10000003,VidCom
          MOVE      VidDat,D0
     And if desired...
          MOVE      VidDat,destination
```

Writing Video Ram

Send an address 4000-7FFF and a bank 0..3 to VidCom, then write data to VidDat. Auto incrementing is active (see below and $8Fxx command).

Example: Write $A5A5 to $C000

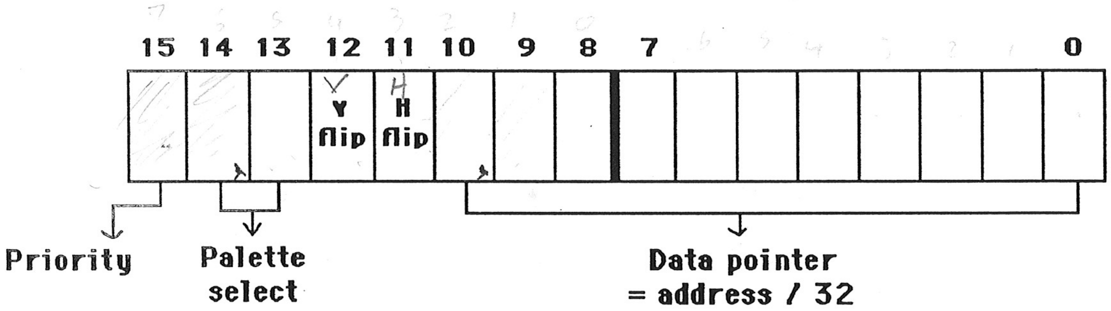
```
          MOVE.L    #$40000003,VidCom
          MOVE      #$A5A5,VidDat
     And if desired...
          MOVE      next data value,VidDat
```

The auto increment value determines the offset (from the initial address) each successive piece data is written in video RAM. An offset of 2 (e.g. $8F02 command) would cause each VidDat write to be placed 2 bytes from the last.

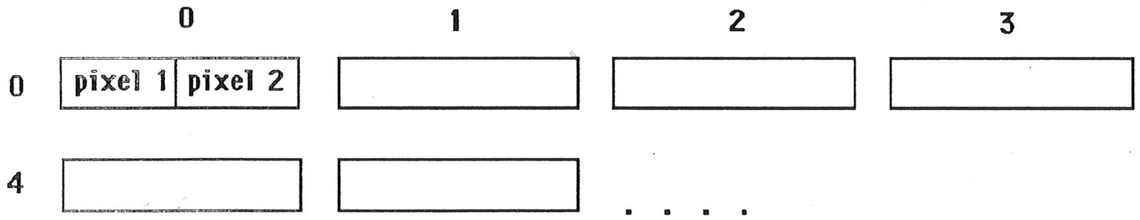
Character Screens

There are three character mapped screens available: the foreground, the background, and the “clip” screen. The foreground and background screens operate identically, except for the standard priority difference (i.e. foreground over background if both priority bits clear). The clip screen is a non-scrolling screen which is mutually exclusive with the foreground. When it is enabled (see 91xx and 92xx), it replaces the foreground screen in the selected area.

Character pointers: Each is a word in size. The address is the 64K video RAM address of the character definition divided by $20.



Each character definition is 32 bytes:



4-bits per pixel, 2 pixels per byte

Priority Levels

Highest

Sprite data
Foreground character data OR Clip Screen character data
Background character data

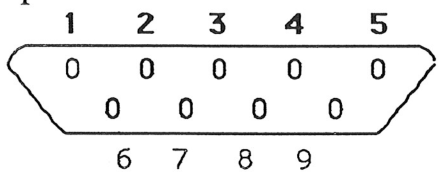
Lowest

If the priority bit in the character pointer of a lower priority image is set, and the bit of a upper priority image is clear, then the priority sequence of those two images is reversed. If both bits are set, the standard priority sequence remains.
A pixel value of 0 is always transparent, regardless of the setting of the priority bit.

Controller Ports

There are a total of three controller ports: two on the front and one on the rear. The two front ports are male DB-9 connectors.

The pinout for front ports is:



Pin
1. I/O Bit 0
2. I/O Bit 1
3. I/O Bit 2

Pin
4- I/O Bit 3
5- +5 volts
6- I/O Bit 4

Pin
7- I/O Bit 6
8- ground
9- I/O Bit 5

The rear port is physically identical, but since it is a female connector, the pin assignments are reversed.

The data I/O ports are one byte wide and memory mapped:

$A10003 = front left port
$A10005 = front right port
$A10007 = rear port

Each port also has a data direction register where each bit is the current I/O state (1= output, 0= input) of the corresponding data I/0 port bit. The control registers are also memory mapped:

$A10009 = front left port
$A1000B = front right port
$A1000D = rear port

I/O bit 6 controls which control pad switches will be returned when the I/O port is read. Therefore, this bit must be initialized at startup to be an output bit (all other bits remain inputs) by writing $40 to the data direction register.

When I/O Bit 6 is set, the I/O read returns:

x	x	C	B	Right	Left	Dn	Up

When I/O Bit 6 is clear, the I/O read returns:

x	x	Start	A	x	x	Dn	Up

NOTE: You must delay for a few cycles between the write and the read to the ports, otherwise the hardware will not have enough time to make the signals stable.

To read the first controller:

```
Initialize          MOVE.B   #$40,$A10009      ;Set DDR in init code

then to read        MOVE.B   #$40,$A10003      ;Read first set
                    NOP
                    NOP
                    MOVE.B   $A10003,D0        ;get status
                    MOVE.B   #0,$A10003        ;read the second set
                    NOP
                    NOP
                    MOVE.B   $A10003,D1        ;get status
```

NOTE: **DO NOT CHANGE** the data direction register or data register of the rear port while using the development system.

Palettes

Four palettes of 16 colors each are located at VidCom write command address SC0000000. This special address is used because the data is stored in video chip registers rather than video RAM.

Each palette is a table of 16 words:

				Blue	X	Green	X	Red	X

Only the upper three bits of each nibble are valid (i.e. values range $0....$E).
Read with VidCom read address of "offset" 0020, such as 00000020 to read palette 0 color 0, or 00060020 to read palette 0 color 3. Auto increment ($8Fxx) is active.

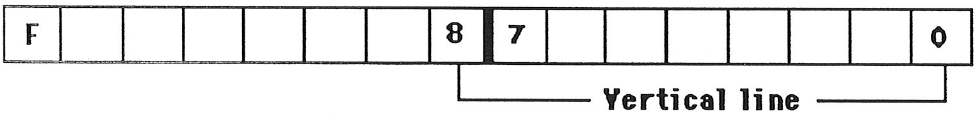
(See also 87xx.)

Vertical Scrolling

The vertical scroll table is located at VidCom write command address $40000010. This special address is used because the data is stored in video chip registers rather than video RAM.

Command sequence:

$40000010 to VidCom
<word>: to VidDat foreground screen scroll
<word>: to VidDat background screen scroll
And if column mode is enabled...
<word>: to VidDat Next foreground column



Value is the line number of data screen to display at line 0 of visual screen.

If column enable bit is ON, the first word scrolls the foreground left two columns. The next word scrolls the background left two columns. Succeeding words similarly affect foreground and background columns proceeding to the right side of the screen.

(See also 8Bxx.)

This table may be read (see Reading Video RAM) at VidCom read command address $00000010.

NOTE: The clip screen is unaffected by scroll settings.

Horizontal Scrolling

The scrolling table is composed of 16 words for each character line, beginning with line 0.

Each word contains the pixel offset (from the left edge) of the displayed character line or row.

Table Format:

<word> offset for foreground whole screen or line 0
<word> offset for background whole screen or line 0
And if enabled...
<word> offset for foreground screen line/group 1
<word> offset for background screen line/group 1

If the whole screen mode is set, the first word scrolls the whole foreground screen and the second word scrolls the whole background screen. If the scan line scroll mode is set, the first word scrolls foreground line 0; the next word scrolls background line 0. Succeeding words scroll the following lines in similar order. In the character line mode, eight scan line groups are scrolled by the first two words of each 16-word group. In the combined mode, only the first 16-word group is used to scroll each scan line. (The first character line group of offsets is repeated down the screen.)

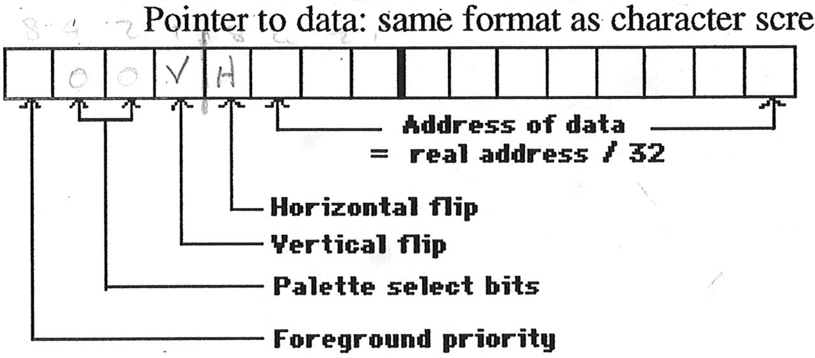
(See also 8Bxx.)

NOTE: The clip screen is unaffected by scroll settings.

Sprite Control

All 80 sprites are controlled by a 640-byte structure. Each sprite has an eight byte control block, as follows:

Byte Offset	**Data**
0,1	Y location word (9 bits) - $80 is screen location 0.
2	Cluster shape 0....$F (see table below)
3	Next sprite block to display; in order of priority (1...$4F). 0 signals the last sprite.
4,5	Pointer to data: same format as character screens



6,7	X location word (9 bits) - $80 is screen location 0. A value of 0 will cause disabling or clipping of lower priority sprites on lines occupied by higher priority sprites.

(See also 85xx)

NOTE: A maximum of 40 ($28) characters of sprite data may be displayed on any one scan line, regardless of the sprite X positions. (e.g. Overlapping sprites are still subject to the 40 character maximum limit)

Cluster Arrangement

1 to 16 characters of data can be displayed in the following shapes. The data is organized by column first, and then row.

Cluster #	Shape		Cluster #	Shape	
0	X	0000	8	XXX	1000
1	X X	0001	9	XXX XXX	1001
2	X X X	0010	A	XXX XXX XXX	1010
3	X X X X	0011	B	XXX XXX XXX XXX	1011
4	XX	01 00	C	XXXX	1100
5	XX XX	0101	D	XXXX XXXX	1101
6	XX XX XX	0110	E	XXXX XXXX XXXX	1110
7	XX XX XX XX	0111	F	XXXX XXXX XXXX XXXX	11 11

WWHH

Remember, a maximum of 40 characters can be displayed per line.

Control Block Table

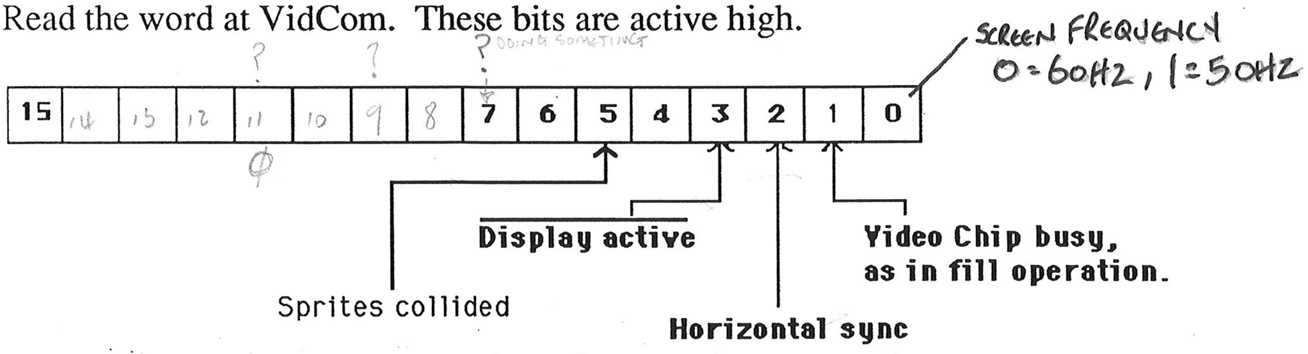
The table is 640 bytes long, with 80 8-byte control blocks available. The first entry is drawn first, then the "next sprite number" value is used to select which sprite block is drawn next (at lower priority).

The next sprite=0 control block is last, and signals the end of the table. It has the lowest priority.

(See also 85xx).

Video Status

Read the word at VidCom. These bits are active high.



The video chip busy bit can be tested during a fill operation to find out if the video chip is done. Sending another command to VidCom during a fill will interrupt the fill operation. Otherwise, this bit can be ignored.
Vsync interrupt apparently occurs within two scan lines after bit 3 changes.
Display Active (negative true) is <u>high</u> when $81xx bit 6 if off (<u>screen disabled</u>).

The sprite collision bit changes to one when sprites have collided, either on screen or off screen. Is is unknown at this time if there is additional sprites collision information, such as a collision table of some sort.

Scan Line Information

Read the word at $C00008. The upper byte is the current scan line position being drawn, while the lower byte may indicate the current column being drawn.

Video Chip Configuration Commands

These commands allow you to map or configure the video RAM and video chip to your specifications. Write each command or command sequence to VidCom. Most of these commands map the location of display components in the video chip RAM.

80XX Command

7	6	5	4	3	2	1	0

Bit#	Result	
0	on:	Scrambles display
1		Unknown - but used in some software
2	on:	Normal display
	off:	Dims display and palettes
3		
4	on:	Scan line interrupt enable (External. interrupt 4 at $70)
5	on:	Blanks left character column
6		
7	on:	Unknown - Seems to disable vertical scroll on a portion of the screen

81XX Command

7	6	5	4	3	2	1	0

Bit #	Result	
0		
1	on:	Unknown - When bit 2 is off, shows patterns
2	on:	Seems required for normal display
3	on:	PAL Display Mode
	off:	NTSC Display Mode
4	on:	Enables fill command and Vid DMA
5	on:	VBlank interrupt enable (External Interrupt 6 at $78)
6	on:	Screen enable
	off:	Blank screen to background color
7	on:	Unknown - Mapping change possibly for Z80

82XX Command

7	6	5	4	3	2	1	0

<u>Bit #</u> **<u>Result</u>**

0
1
2
3 -----
4 |--- Selects bank for foreground character screen (e.g. character pointers) on 8k boundary
5 -----
6
7

83XX Command

7	6	5	4	3	2	1	0

<u>Bit #</u> **<u>Result</u>**

0
1
2 -----
3 |--- Selects bank for "clip" screen on 4k boundary (see 91xx, 92xx)
4 |
5 -----
6
7

84XX Command

7	6	5	4	3	2	1	0

```
Bit #  Result
 0 -----
 1      |---   Selects bank for background character screen on 8k boundary
 2 -----
 3
 4
 5
 6
 7
```

85XX Command

7	6	5	4	3	2	1	0

```
Bit #  Result
 0
 1 -----
 2      |
 3      | ---   Selects Sprite Control block location on 1k boundaries
 4      |
 5      |
 6 -----
 7
```

86XX Command

7	6	5	4	3	2	1	0

Bit #	Result
0	Unused command. ???
1	
2	
3	
4	
5	
6	
7	

87XX Command

7	6	5	4	3	2	1	0

Bit # **Result**

```
0-----
1    |
2    | ---   Index into palettes for background color displayed for pixel 0 data.
3    |
4    |
5-----
6
7
```

88XX Command

7	6	5	4	3	2	1	0

Bit #	Result
0	Unused command. ???
1	
2	
3	
4	
5	
6	
7	

89XX Command

7	6	5	4	3	2	1	0

Bit #	Result
0	Unused command. ???
1	
2	
3	
4	
5	
6	
7	

8AXX Command

7	6	5	4	3	2	1	0

```
Bit #  Result
  0     ----
  1        |
  2        |        This register controls the frequency with which scan line
  3        |------- interrupts occur.  The basic formula is: An interrupt will
  4        |        occur every 1 + n lines, where n is the register value.
  5        |        When set to 0, an interrupt occurs every scan line.  When
  6        |        set to 8, the first interrupt will occur on line 8, then line 16, etc.
  7     ----        This value may be changed during the interrupt for non-regular
                    spacing requirements.
```

8BXX Command

7	6	5	4	3	2	1	0

```
Bit #  Result
  0     --------    Horizontal scroll mode select
               |    00 - whole screen
               |--- 01 - scan line/character line.  First eight lines
               |          are repeated down the screen.
               |    10 - character line
  1     --------    11 - scan line
  2            on:  V-Scroll columnar scroll mode enable
  3
  4
  5
  6            on:  Unknown - causes halt sometimes
  7
```

8CXX Command

7	6	5	4	3	2	1	0

Bit #	Result	
0	on:	Enables 320 pixel horizontal format
	off:	Enables 256 pixel format
1	on:	Interlace mode enable
2	on:	With interlace bit,may display FG character lines interlaced with BG character lines
3	on:	Shiny/shadow mode enable
4		
5		
6		
7	on:	Intensity (also some scan change)

Shiny/shadow mode

When enabled, there are two cases:

A) Any colored, displayed pixel data from sprites or character map pages will be shown dimmed if its pointer does not have its priority bit set AND the pixel does not intersect any other character that has its priority bit set. If the priority bit is set in either case, the pixel will be displayed normally, with proper display priority still in effect.

B) Sprites may also now be used to dim or highlight, on a pixel-by-pixel basis, any character map data they overlay. When a sprite uses palette 3, a pixel data value of $E will highlight the underlying character screen data, while a pixel data value of $F will dim the underlying data provided the sprite has priority over the character data that it covers. Any overlaid sprite data of lower priority is erased. All colors, including black, will be highlighted by palette 3, pixel $E sprite data. Dimming occurs to all colors, but is unnoticeable on black. All of palette 3 is still usable with normal results by screen character data.

8DXX Command

7	6	5	4	3	2	1	0

```
Bit #  Result
 0-----
 1     |
 2     |---    Address of Horizontal scroll table 1k boundary
 3     |
 4     |
 5-----
 6
 7
```

8EXX Command

7	6	5	4	3	2	1	0

```
Bit #  Result
 0              Unused command ???
 1
 2
 3
 4
 5
 6
 7
```

8FXX Command

7	6	5	4	3	2	1	0

```
Bit #  Result
 0-----
 1     |
 2     |
 3     |---   Auto increment value for video chip.  Resets counter to 0 when
 4     |      changed.
 5     |
 6     |
 7-----
```

This register controls the writing of successive values to VidDat. The value is the offset, in bytes from the last address used, that each word written to VidDat will be placed into video RAM. A value of 2 will cause successive words to be placed contiguously in video RAM, while a value of $10 would cause each word to be written $10 bytes from the last.

90XX Command

7	6	5	4	3	2	1	0

```
Bit           Result
 0-----       00    characters per line 32
       |---   01    characters per line 64
       |      10    repeats top 32 character line entire screen
 1-----       11    characters per line 128
 2
 3
 4            Unknown - but used in some software
 5            0 = 32 LINES, 1=64 LINES
 6
 7
```

91XX Command

7	6	5	4	3	2	1	0

<u>Bit #</u> <u>Result</u>

```
0-----
1    |
2    |
3    |---    Number of two character-wide columns to clip, 0 = none.
4    |
5    |
6-----
7            on:   Display xx columns from foreground left edge,
                   then display remaining screen area from the clip screen
             off:  Display clip screen data xx columns from left edge,
                   then display foreground screen data.
```

92XX Command

7	6	5	4	3	2	1	0

<u>Bit #</u> <u>Result</u>

```
0-----
1    |
2    |
3    |---    Number of character rows to clip, 0 = none.
4    |
5    |
6-----
7            on:   Display xx character lines from foreground top,
                   then display remaining screen area from the clip screen
             off:  Display clip screen data xx character lines from top,
                   then display foreground screen data.
```

93xx 94yy 9780 Command

Video Memory Fill

Send the following commands to VidCom to fill video RAM. The data byte (or hi byte of word data) output for the fill value is written to video memory according to the auto-increment skip value. 81xx bit 4 must be on for this command to work.

To VidCom:

93xx 94yy where yyxx is fill length -1

9780

<Addr> <Bank> with bit 7 of low word set

<fill data> (Byte, or high byte of word) Send to VidDat

Example:

Fill $C000 to $C3FF with A5

```
        MOVE      #$8F01,VidCom        ;Set byte auto inc
        MOVE.L    #$93FF9403,VidCom    ;Length - 1
        MOVE      #$9780,VidCom        ;End command
        MOVE.L    #$40000083,VidCom    ;Address
        MOVE      #$A500,VidDat        ;Data
```

93xx 94yy 95aa 96bb 97cc Command

Video Memory DMA (copy)

You can instruct the video controller to DMA the main address space for data as follows. 81xx bit 4 must be on for this command to work.

To Vidcom:

93xx 94yy Where yyxx is the number of words to copy. Auto Inc is active.
LOW HIGH

95aa 96bb 97cc Where ccbbaa is the long address of the source data divided by 2.

<addr> <bank> Address of video memory to copy to, and bank number with bit 7 set.

Example:
To cause the video chip to read all four palettes ($40 words) from $FFE000 in the 68000 RAM area.

```
MOVE      #$8F02,VidCom         ;Set byte auto inc
MOVE.L    #$93409400,VidCom     ;Length
MOVE.L    #$950096F0,VidCom     ;Address lo and med
MOVE      #$977F,VidCom         ;Address hi
MOVE.L    #$C0000080,VidCom     ;Address
MOVE      #$A500,VidDat         ;Data
```

SOURE DATA CANMOT CROSS BOUNDRY

This command forces a one time copy of the data to video RAM. IT CANNOT CROSS a 64k * autoinc value boundary (e.g., if autoinc = 2, can't cross 128k).

During vertical blanking (when the video display is not fetching data), the transfer rate is 2 Mb per second. However, during display time, the DMA access is interleaved with video display access, and the rate drops significantly. The same slowdown (during display time) will affect direct writes from the 68000 to video RAM.

THIS WILL CAUSE Z80 INTERUPT PROBLEMS IF DMA IS GOING ON WHEN THE Z80 WANT AN INTEROPT ?

Appendix A

A11000 = ?

Z-80 Coprocessor

Z80 Buss request $A11100
Send a $0100 to request the Z80 Buss (which is mapped in the 68000 $A00000 space) "THIS PULLS THE Z80 $\overline{BUSRQ}$ LINE LOW
Read location to verify grant, 0 = granted (bit 9) Valid only when not reset.
SEND A $0000 TO RELEASE THE Z80 "THIS SETS THE Z80 $\overline{BUSRQ}$ LINE HIGH

Z80 Reset line $A11200
Send a 0000 to hold the Z80 reset line low. Send a $0100 to start the Z80. A delay between the low/high transition should be performed in order to allow the reset cycle to be complete. Example:

```
MOVE       #0,$A11200         ;reset
MULU       D0,D0              ;delay
MOVE       #$0100,$A11200  ;start 'er up
```

Addresses $A00000 through $A01FFF are the 8K Z80 RAM (Z80 addresses $0000 through $1FFF).

Addresses $A04000 through $A04003 are the sound chip register/data address pairs, which are located at $4000 through $4003 for the Z80.

Address $A06000 (Z80 address $6000) is the Z80 memory mapping register. This register controls what area of RAM or ROM will be "seen" by the Z80 in its upper 32K of address space.. Set this "window" by writing to the map register at $6000. Nine successive bytes must be written to this register. Only bit 0 is significant for each byte, and the bits are written from lowest significant bit to the highest. These nine bits represent the upper nine bits of a 24 bit absolute address in the 68000 RAM or ROM.

Addresses $A08000 through $A0FFFF are the Z80's 32K "window" into the 680000's address space

Address $C00011, when accessed from either the Z-80 or the 68000, seems to write data directly to the sound attenuation registers.

The Z-80 reset line is held low on power up; i.e. in a permanent reset state. All data transfer or buss request operations are valid only when the Z-80 is in a non-reset state.

POWERUP STATE OF THE Z80
$\overline{RESET}$ = 0V
$\overline{BUSRQ}$ = 5V
$\overline{WAIT}$ = 5V
$\overline{BUSAK}$ =
$\overline{RD}$ =
$\overline{WR}$
$\overline{INT}$ = 5V
$\overline{NMI}$ = 5V
$\overline{HALT}$ =
$\overline{MREQ}$ = 5V
$\overline{IORQ}$ =

NOTE: When reading or writing from the 68000 to the Z-80 address space, use BYTE sized operations only.

Miscellaneous information:
Int 38h occurs at twice the screen refresh rate (120 Hz for NTSC systems)
Set IM 1 on startup.
The Z80 clock speed appears to be 3.57 MHz, the colorburst rate.
An invalid read from the upper 32K "window" of the Z-80 seems to always return a value of $FF.

Digitized Audio

The Yamaha sound chip has a 8 bit D/A converter which can be used for replaying digitized audio samples. This feature is available in the first pair of register and data addresses at $4000 (register number) and $4001 (value for register).

Enable the D/A converter by writing a $80 to register $2B. Disable the mode by writing a $00 to register $2B.

```
LD   A,2Bh      ;Set D/A mode active
LD   4000h,A    ;by writing to register $2B
LD   A,80h
LD   4001h,A    ;with a $80
```

Once enabled, the audio samples are sent to register $2A from a software timing loop. A typical program will delay with an empty DNJZ loop with a count appropriate to the sampling rate of the data. Then send the data to register $2A.

```
LD   A,2Ah      ;Write the amplitude to $2A
LD   4000h,A
LD   A,C        ;Let's say that the data is in C
LD   4001h,A
```

Appendix B

Items requiring further study:

1) Twelve words of information may be read from $C00004 (VidCom) through $C0000F. While some of this information is currently known, there may be significant status and/or configuration information present in these data.

2) Access to $C00011 from the Z-80 or 68000 seems to write information directly to the audio chip's attenuation registers. Could this chip be memory mapped in this address space?

3) The video commands $86xx, $88xx, $89xx, and $8Exx are completely unknown. It is unlikely that they are unused, just that their effect is unknown because of interrelationships with other settings.

4) The video commands listed below have unknown bits, or bits whose definition is not entirely certain.

$80xx bits 0,1, and 7
$81xx bits 1,2, and 7
$8Bxx bit 6
$8Cxx bits 1 and 2
$90xx bit 4