



GENESIS
SOFTWARE MANUAL
SEGA ENTERPRISES

Copyright 1989 Sega of Japan

This manual may not, in whole or part, be copied, photocopied, reproduced, or translated without prior written consent from Sega Enterprises.

Genesis Technical Overview

February 28, 1990

GENESIS:

68000 @8MHz

- Main CPU
- 1 MByte (8 Mbit) ROM Area
- 64 KByte RAM Area

VDP (Video Display Processor)

- Dedicated video display processor
- Controls playfield and sprites
- Capable of DMA
- Horizontal and vertical interrupts

Z80 @4MHz

- Controls PSG (Programmable Sound Generator) & FM Chips
- 8 KBytes of dedicated Sound RAM

VIDEO:

- NOTE: Playfield and Sprites are character-based
- Display Area (visual)
 - 40 Chars wide x 28 chars high
 - Each char is 8 x 8 pixels
 - Pixel resolution = 320 x 224
 - 3 Planes
 - 2 Scrolling playfields
 - 1 Sprite plane
 - Definable priorities between planes
 - Playfields:
 - 6 Different sizes
 - 1 Playfield can have a "fixed" window
 - Playfield map
 - Each char position takes 2 bytes, that includes:
 - Char name (10 bits); points to char definition
 - Horizontal flip
 - Vertical flip
 - Color palette (2 bits); index into CRAM
 - Priority

- Scrolling:
 - 1 Pixel scrolling resolution
 - Horizontal:
 - Whole playfield as unit
 - Each character line
 - Each scan line
 - Vertical:
 - Whole playfield as unit
 - 2 Char wide columns
- Sprites:
 - 1 x 1 Char up to 4 x 4 chars
 - Up to 80 sprites can be defined
 - Up to 20 sprites displayed on a scan line
 - Sprite priorities
- Character Definitions
 - 4 bits/pixel; points to color register
 - 4 bytes/scanline of char
 - 32 bytes for complete char definition
 - Playfield and sprite chars are the same

COLOR:

- Uses CRAM (part of the VDP)
- 64 9-bit wide color registers
 - 64 colors out of 512 possible colors
 - 3 bits of Red
 - 3 bits of Green
 - 3 bits of Blue
- 4 palettes of 16 colors
 - 0th color (of each palette) is always transparent

OTHER:

- DMA
 - Removes the 68000 from the BUS
 - Can move 205 bytes/scanline during VBLANK
 - There are 36 scanlines during VBLANK
 - DMA can move 7380 bytes during VBLANK
- Horizontal & Vertical Interrupts

SOUND:

- Z80 Controls:
 - PSG (TI 76489 chip)
 - FM chip (Yamaha YM 2612)
 - 6-Channel stereo
 - Z80 can access ROM data
 - 8 KBytes RAM

HARDWARE:

- 2 Controllers
 - Joypad
 - 3 Buttons
 - Start button
- 1 External port
- 2 Video-outs (RF & RGB)
- Audio jack (stereo)
- Volume control (for audio jack)



TABLE OF CONTENTS

I.	MEMORY MAP	1
A.	Mega Drive 16 Bit Mode	1
1.	68K MEMORY MAP	1
2.	Z80 MEMORY MAP	2
3.	68000 ACCESS TO Z80 MEMORY	2
4.	I/O AREA	3
5.	CONTROL AREA	3
6.	VDP AREA	4
II.	VDP 315-5313	5
A.	TERMINOLOGY	6
B.	DISPLAY SPECIFICATION	6
C.	VDP STRUCTURE	8
1.	CTRL	8
2.	VRAM	8
3.	CRAM	8
4.	VSRAM	8
5.	DMA	8
D.	INTERRUPT	9
1.	VERTICAL INTERRUPT (V-INT)	9
2.	HORIZONTAL INTERRUPT (H-INT)	9
3.	EXTERNAL INTERRUPT (EX-INT)	10
E.	VDP PORT	10
1.	\$C00000 (DATA PORT)	10
2.	\$C00004 (CONTROL PORT)	11
3.	\$C00008 (HV COUNTER)	12
F.	VDP REGISTER	13
1.	MODE SET REGISTER NO.1	13
2.	MODE SET REGISTER NO.2	13
3.	PATTERN NAME TABLE BASE ADDRESS FOR SCROLL A	13
4.	PATTERN NAME TABLE BASE ADDRESS FOR WINDOW	13
5.	PATTERN NAME TABLE BASE ADDRESS FOR SCROLL B	14
6.	SPRITE ATTRIBUTE TABLE BASE ADDRESS	14
7.	BACKGROUND COLOR	14
8.	H INTERRUPT REGISTER	14
9.	MODE SET REGISTER NO. 3	15
10.	MODE SET REGISTER NO. 4	15
11.	H SCROLL DATA TABLE BASE ADDRESS	16
12.	AUTO INCREMENT DATA	16
13.	SCROLL SIZE	16
14.	WINDOW H POSITION	16
15.	WINDOW V POSITION	17
16.	DMA LENGTH COUNTER LOW	17
17.	DMA LENGTH COUNTER HIGH	17

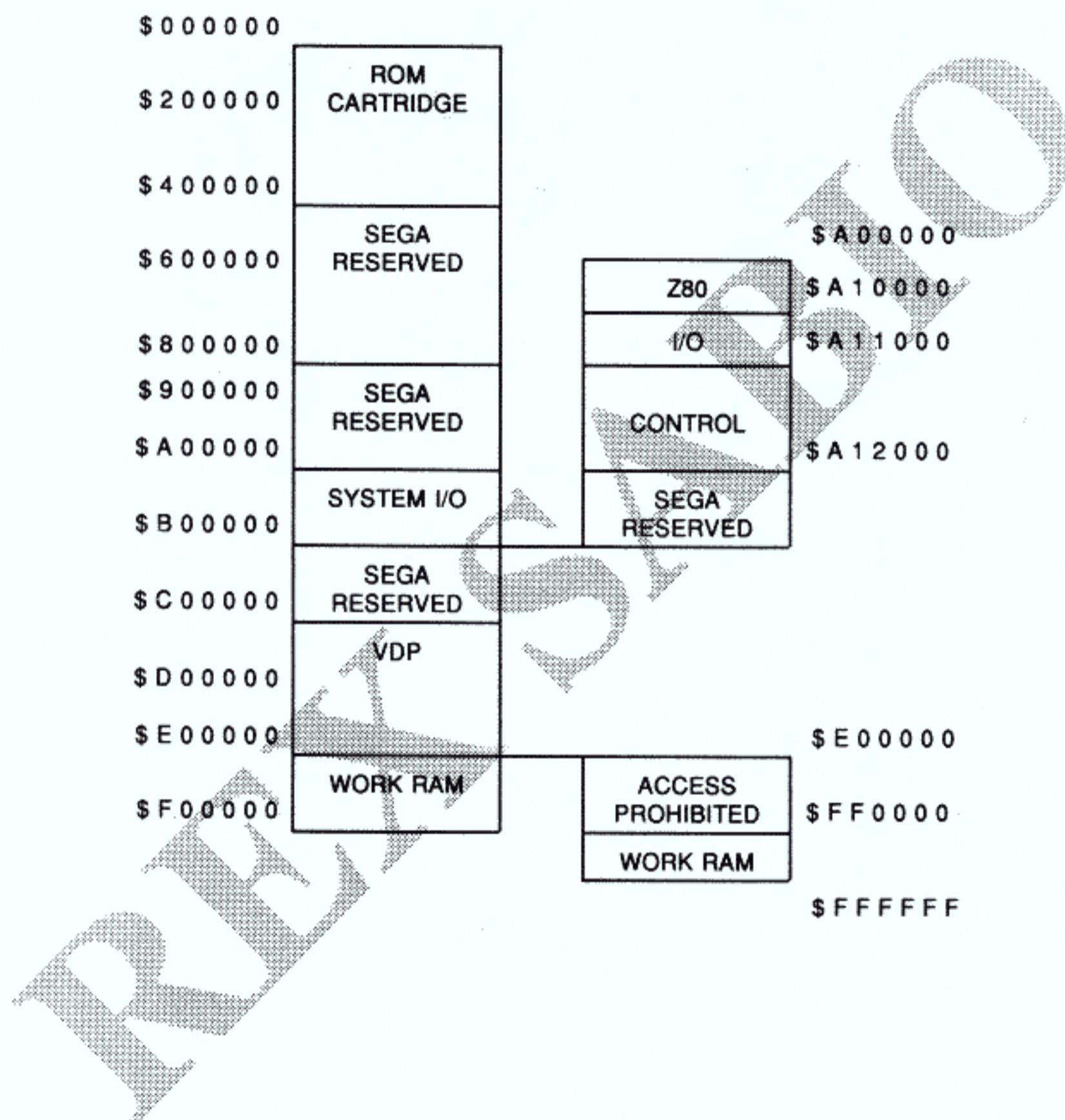
18.	DMA SOURCE ADDRESS LOW	17
19.	DMA SOURCE ADDRESS MID	17
20.	DMA SOURCE ADDRESS HIGH	17
G.	ACCESS VDP RAM	18
1.	RAM ADDRESS SETTING	18
2.	VRAM ACCESS	19
3.	CRAM ACCESS	22
4.	VSRAM ACCESS	23
5.	ACCESS TIMING	24
6.	HV COUNTER	25
H.	DMA TRANSFER	26
1.	MEMORY TO VRAM	26
2.	VRAM FILL	28
3.	VRAM COPY	31
4.	DMA TRANSFER CAPACITY	34
I.	SCROLLING SCREEN	35
1.	SCROLLING SCREEN SIZE	37
2.	HORIZONTAL SCROLLING	39
3.	VERTICAL SCROLLING	41
4.	SCROLL PATTERN NAME	42
5.	PATTERN GENERATOR	44
J.	WINDOW	46
1.	DISPLAY POSITION	46
2.	WINDOW PRIORITY	50
3.	WINDOW PATTERN NAME	51
K.	SPRITE	52
1.	DISPLAY POSITION	53
2.	SPRITE ATTRIBUTE	55
3.	SPRITE SIZE	56
4.	SPRITE's DISPLAY CAPACITY	57
5.	PRIORITY BETWEEN SPRITES	58
6.	SPRITE PATTERN GENERATOR	60
L.	PRIORITY	61
M.	COLOR PALETTE	67
N.	INTERLACE MODE	69
III.	BACKWARD COMPATIBILITY MODE	71
A.	MARK III (MS-Japan)	71
B.	MASTER SYSTEM	71
C.	RAM BOARD	71

IV.	SYSTEM I/O	72
A.	VERSION NUMBER	72
B.	I/O PORT	72
C.	MEMORY MODE	76
D.	Z80 CONTROL	76
1.	Z80 BusReq	76
2.	Z80 Reset	76
E.	Z80 AREA	77
V.	VRAM MAPPING	79
VI.	SOUND SOFTWARE MANUAL	
A.	280 MAPPING	
B.	68K CONTROL OF Z80	
C.	FM SOUND CONTROL	
D.	PSG CONTROL	
E.	D/A CONTROL	
VII.	APPENDIX	

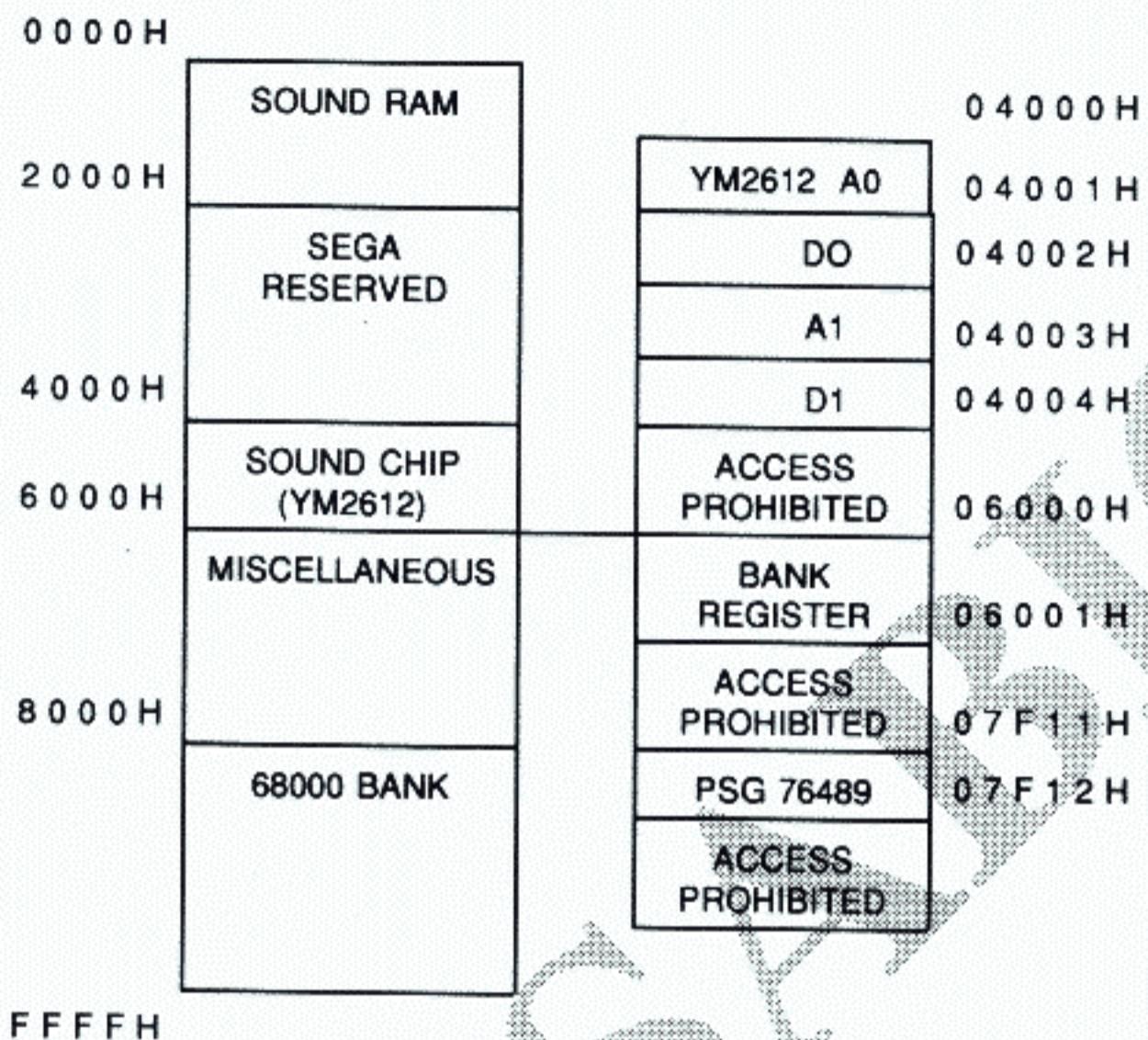
I. MEMORY MAP

A. Mega Drive 16 Bit Mode (as distinct from Master System Compatibility Mode)

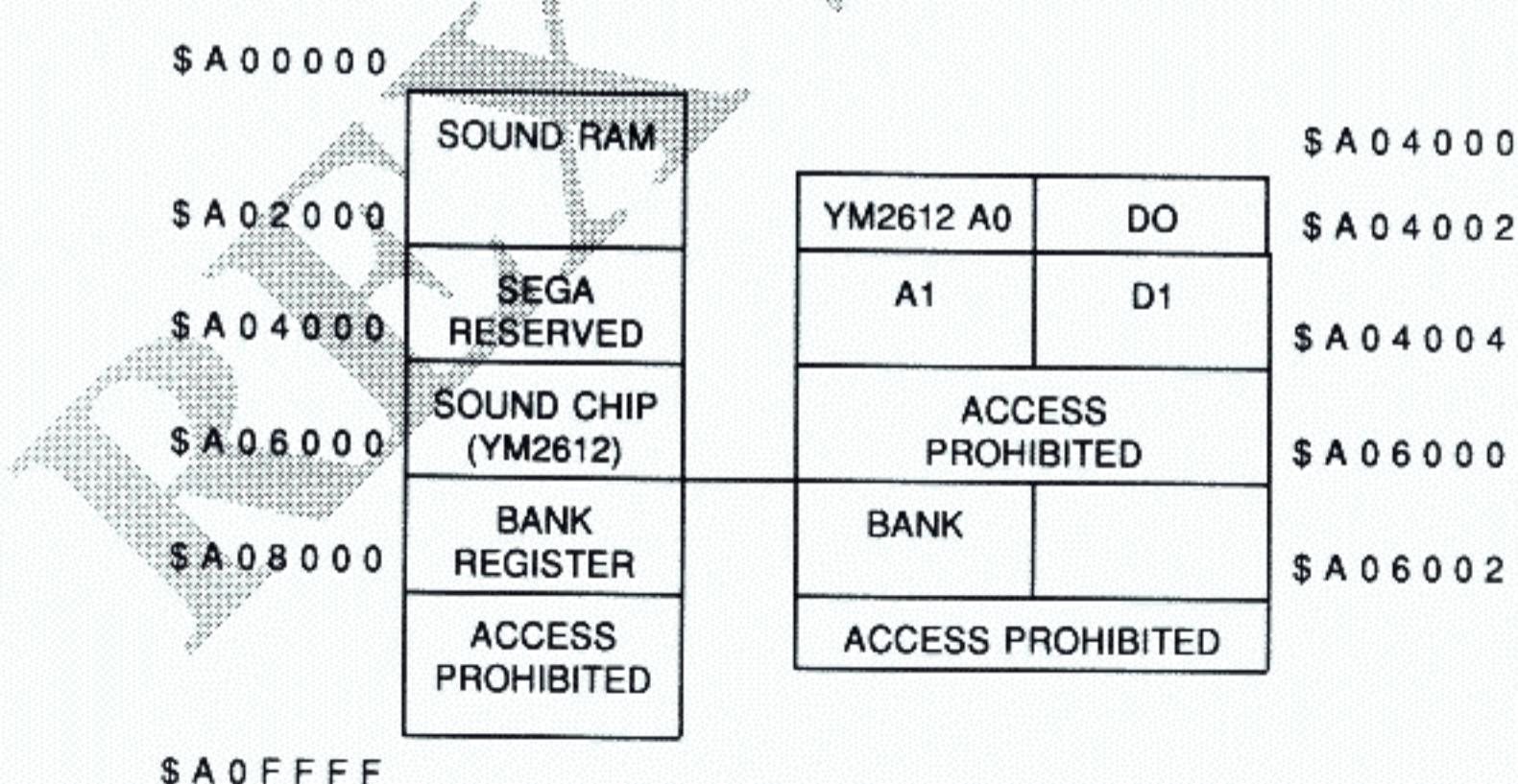
1. 68K MEMORY MAP



2. Z80 MEMORY MAP



3. 68000 ACCESS TO Z80 MEMORY



4. I/O AREA

\$ A10000	
\$ A10002	Version No.
\$ A10008	DATA (CTRL 1) DATA (CTRL 2) DATA (EXP)
\$ A1000E	CONTROL (1) CONTROL (2) CONTROL (E)
\$ A10014	T x DATA R x DATA (1) S - MODE
\$ A1001A	T x DATA R x DATA (2) S - MODE
\$ A10020	T x DATA R x DATA (E) S - MODE
\$ A1 FFFF	ACCESS PROHIBITED

5. CONTROL AREA

\$ A11000	MEMORY MODE
\$ A11002	ACCESS PROHIBITED
\$ A11100	Z80 BUSREQ
\$ A11102	ACCESS PROHIBITED
\$ A11200	Z80 RESET
\$ A11202	ACCESS PROHIBITED
\$ A1 FFFF	

6. VDP AREA

\$ C 0 0 0 0 0

\$ C 0 0 0 0 4

DATA

\$ C 0 0 0 0 8

CONTROL

\$ C 0 0 0 0 A

HV COUNTER

\$ C 0 0 0 1 0

ACCESS
PROHIBITED

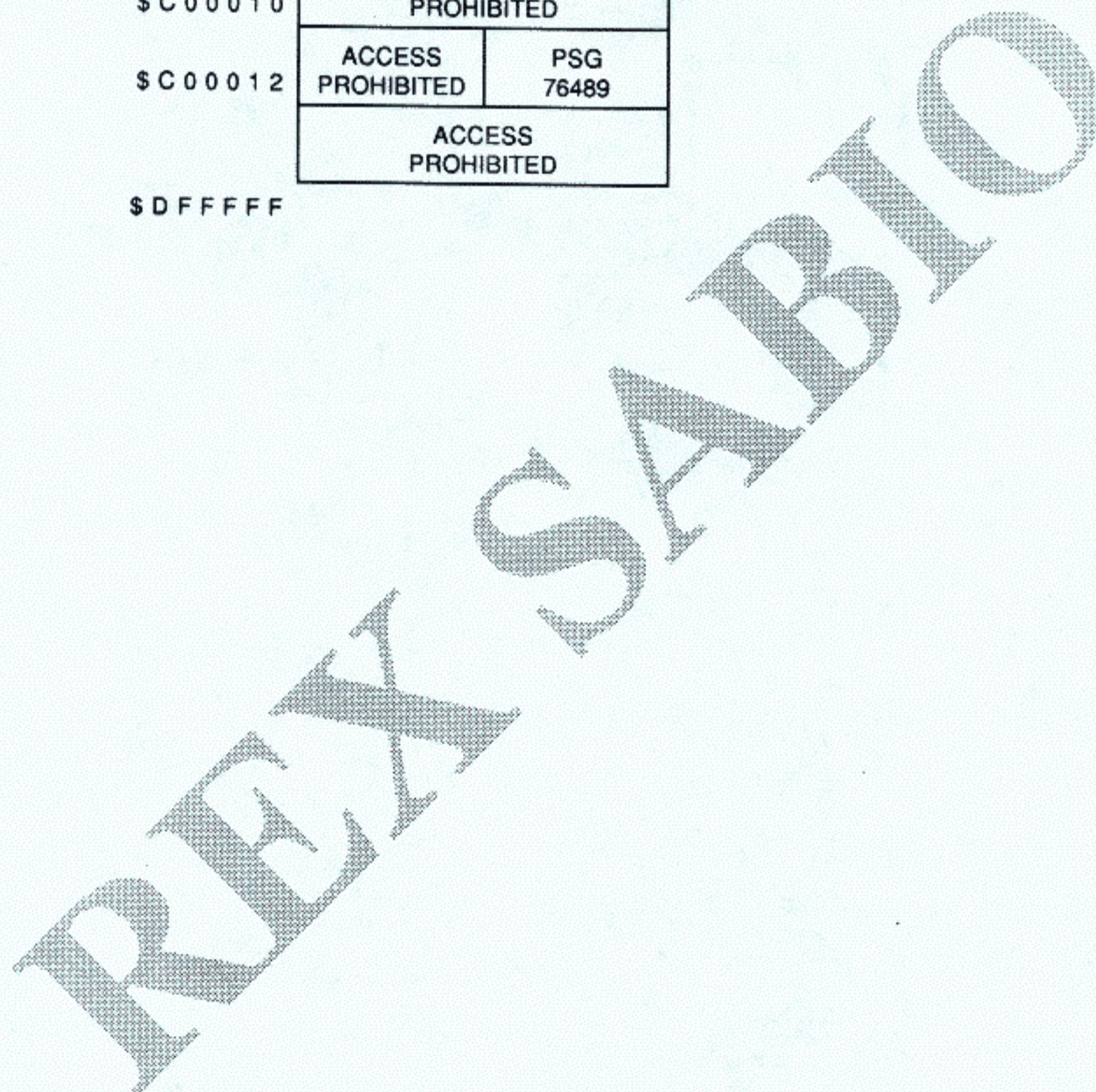
\$ C 0 0 0 1 2

ACCESS
PROHIBITED

PSG
76489

\$ D F F F F F

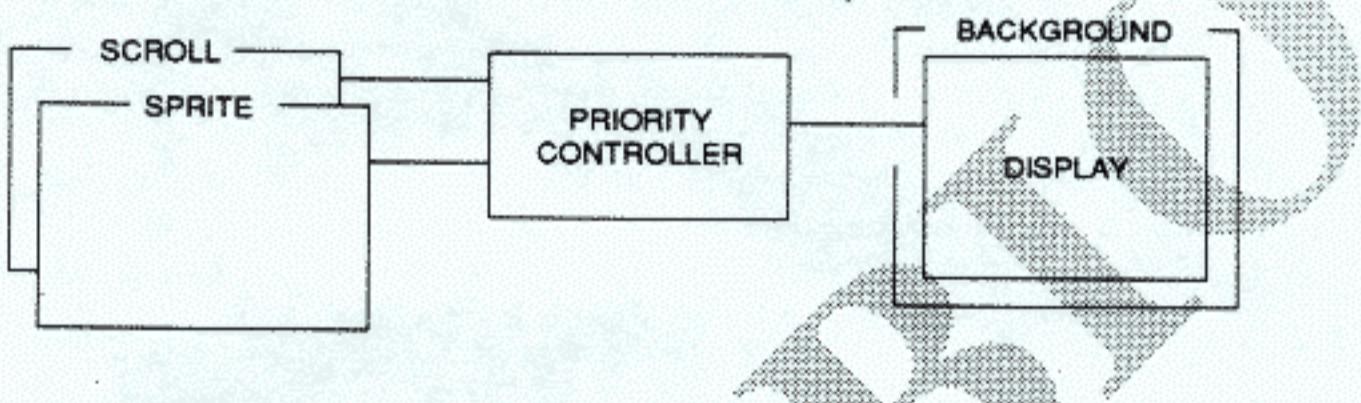
ACCESS
PROHIBITED



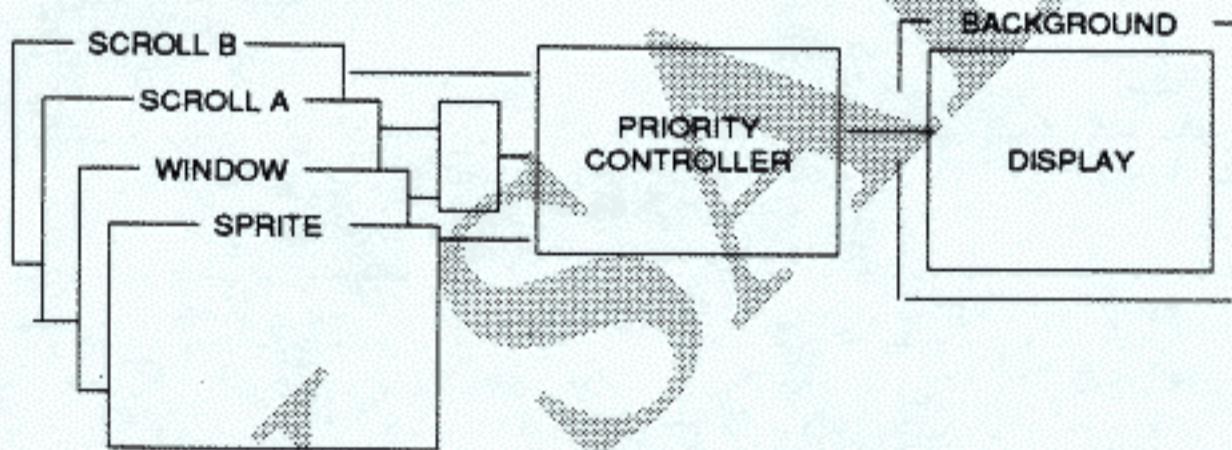
II. VDP 315-5313 (Video Display Processor)

The VDP controls screen display. VDP has graphic Modes IV and V, where Mode IV is for compatibility with the MASTER SYSTEM and V is for the new Mega drive functions. There are no advantages to using Mode IV, so it is assumed that all Mega drive development will use Mode V. In Mode V, the VDP display has 4 planes: SPRITE, SCROLLA/WINDOW, SCROLLB, and BACKGROUND.

GRAPHIC IV MODE (COMPATIBILITY MODE)



GRAPHIC V MODE (16 BIT MODE)



A. TERMINOLOGY

1. A unit of position on X, Y coordinates is called a DOT.
2. A minimum unit of display is called a PIXEL.
3. CELL means an 8 (pixel) x 8 (pixel) pattern.
4. SCROLL indicates a repositionable screen-spanning play field.
5. CPU usually indicates the 68000.
6. VDP stands for Video Display Processor.
7. CTRL stands for Control.
8. VRAM stands for VDP RAM, the 64K bytes area of RAM accessible only through the VDP.
9. CRAM stands for Color RAM, 64 9 bit words inside the VDP chip.
10. VSRAM stands for Vertical Scroll RAM, 40 10 bit words inside the VDP chip.
11. DMA stands for Direct Memory Access, the process by which the VDP performs high speed fills or memory copies.
12. PSG stands for Programmable Sound Generator, a class of low-capability sound chips. The Mega drive contains a Texas Instruments 76489 PSG chip.
13. FM stands for Frequency Modulation, a class of high-capability sound chip. The Mega drive contains a Yamaha 2612 FM chip.

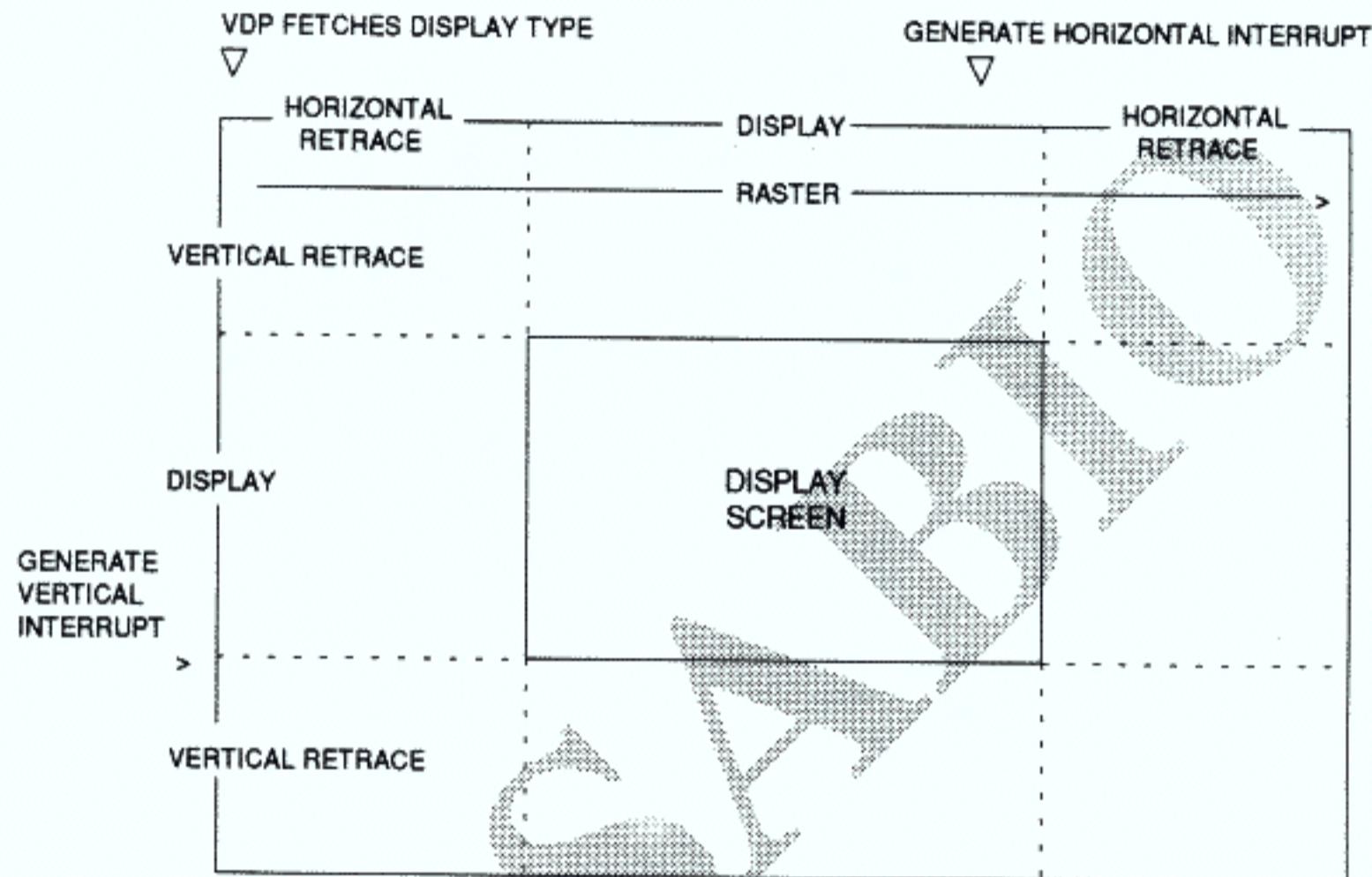
B. DISPLAY SPECIFICATION

DISPLAY SPECIFICATION OUTLINE

DISPLAY SIZE	THERE ARE TWO MODES: 32 * 28 CELL (256 * 224 PIXEL) 40 * 28 CELL (320 * 224 PIXEL)
CHARACTER GENERATOR	8 * 8 CELLS 1300-1800 depending on general system configuration.
SCROLL PLAYFIELDS	Two scrolling play fields, whose size in cells is selectable from: 32 * 32, 32 * 64, 32 * 128 64 * 32, 64 * 64 128 * 32
SPRITE	Sprite size is programmable on a sprite by sprite basis, with the following choices: 8 * 8, 8 * 16, 8 * 24, 8 * 32 16 * 8, 16 * 16, 16 * 24, 16 * 32 24 * 8, 24 * 16, 24 * 24, 24 * 32 32 * 8, 32 * 16, 32 * 24, 32 * 32 There are 64 sprites available when the screen is in 32 cell wide mode, or 80 when the screen is in 40 cell wide mode.
WINDOW	1 window associated with the Scroll A play field.
COLORS	64 colors/512 possibilities.

For PAL (the European television 50 Hz standard), a vertical size of 30 cells (240 dots) is selectable.

The VDP supports both NTSC and PAL television standards. In both cases, the screen is divided into active scan, where the picture is displayed, and vertical retrace (or vertical blanking) where the monitor prepares for the next display.



Numbers of rasters in a screen are as follows:

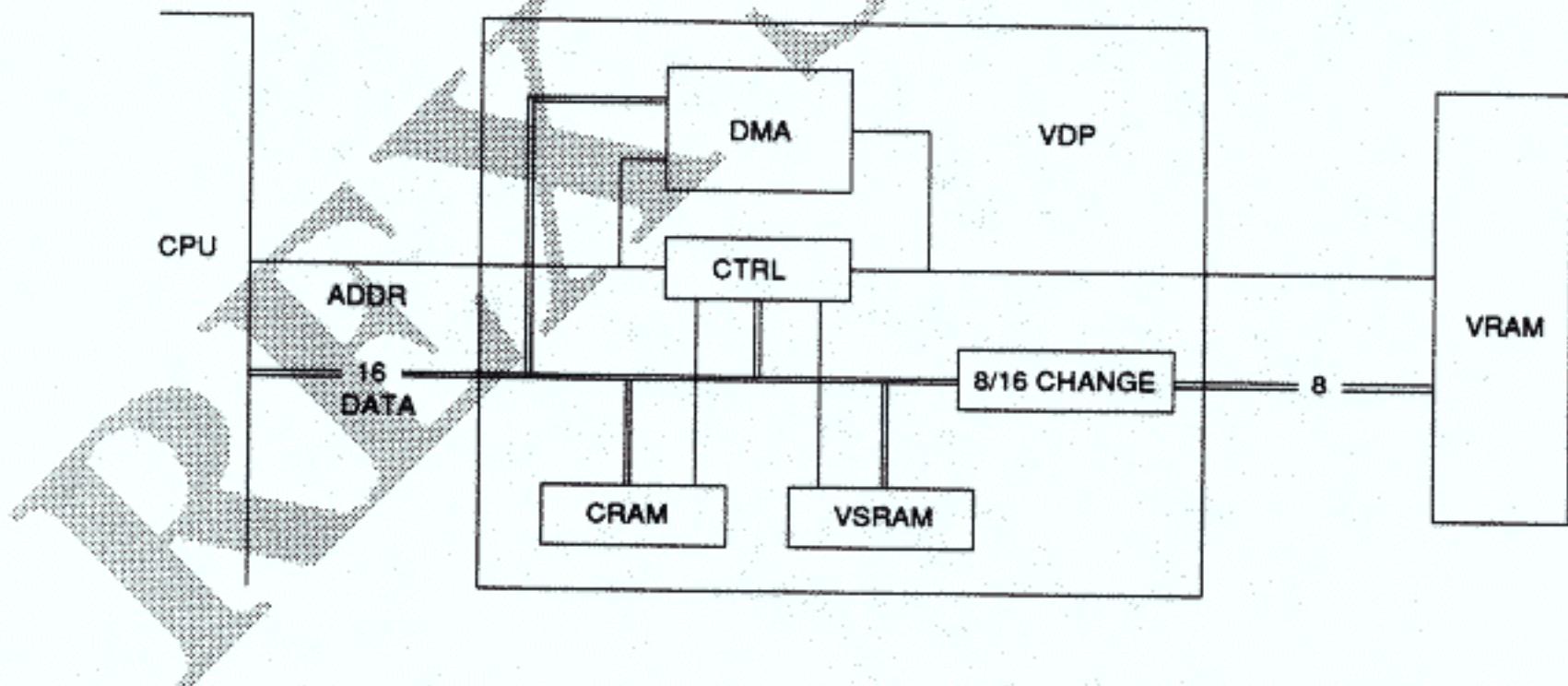
	LINES/SCREEN	VCELL NO.	LINE NO. (DISPLAY)	LINE NO. (RETRACE)
NTSC	262	28	224 RASTER	38 RASTER
PAL	312	28	224 RASTER	98 RASTER
		30	240 RASTER	82 RASTER

C. VDP STRUCTURE

The CPU controls the VDP by special I/O memory locations.

1. CTRL (control)
This controls REGISTER, VRAM, CRAM, VSRAM, DMA DISPLAY, etc.
2. VRAM (VDP RAM)
General purpose storage area for display data.
3. CRAM (COLOR RAM)
64 colors divided into 4 palettes of 16 colors each.
4. VSRAM (Vertical Scroll RAM)
Up to 20 different vertical scroll values each for scrolling play fields A and B.
5. DMA (Direct Memory Access)
The VDP may move data at high speed from CPU memory to VRAM, CRAM, and VSRAM instead of the CPU, by taking the 68000 off the bus and doing DMA itself.

The VDP can also fill the VRAM with a constant, or copy from VRAM to VRAM without disturbing the 68000.



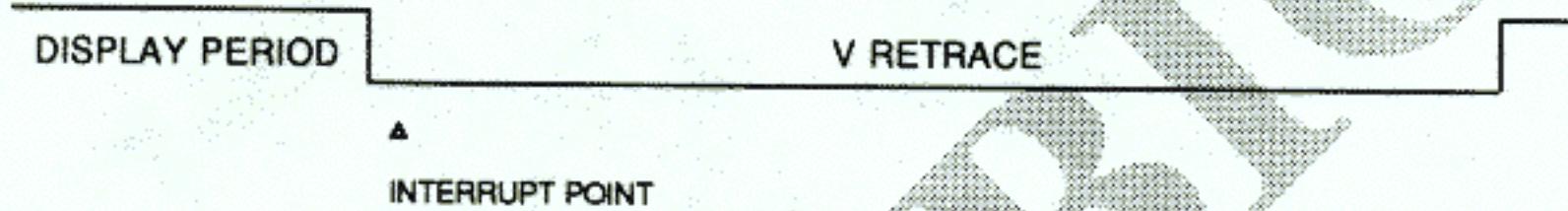
D. INTERRUPT

There are three interrupts, Vertical, Horizontal, and External. You can control each interrupt by the IEO, IE1, and IE2 bits in the VDP registers. The interrupts use the AUTO-VECTOR mode of the 68000 and are at levels 6, 4, and 2 respectively; the level 6 vertical interrupt having the highest priority.

IE0 V Interrupt	(LEVEL 6)
IE1 H Interrupt	(LEVEL 4)
IE2 External Interrupt	(LEVEL 2)
1: Enable	
0: Disable	

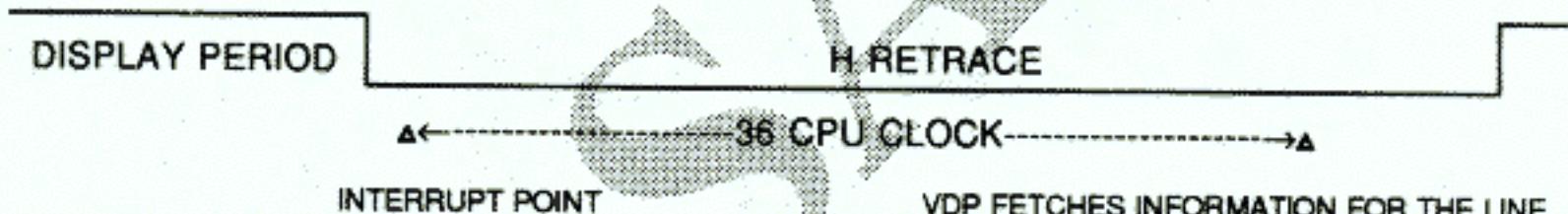
1. VERTICAL INTERRUPT (V-INT)

The vertical interrupt occurs just after V retrace.

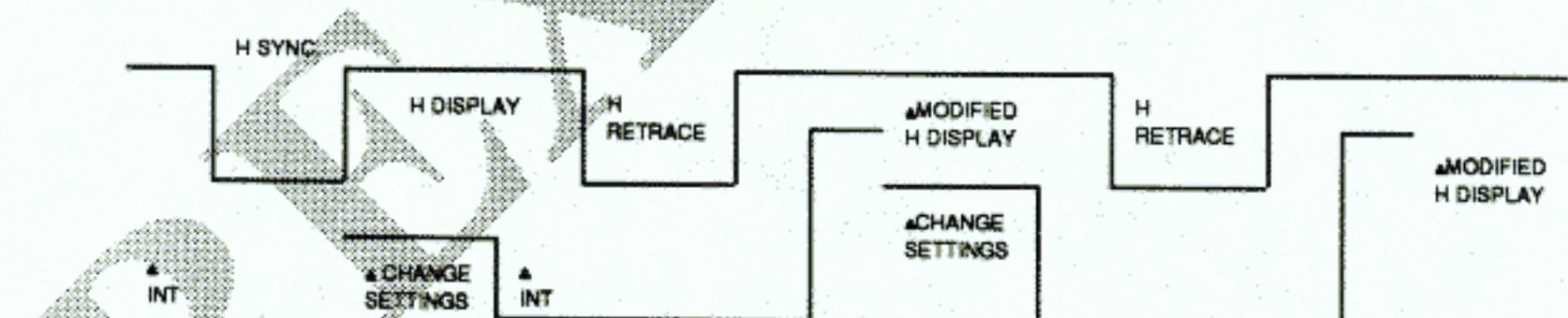


2. HORIZONTAL INTERRUPT (H-INT)

The Horizontal Interrupt occurs just before H retrace.



The VDP loads the required display information, including all required register values, for the line in about 36 clocks, thus the CPU can control the display of the next line but not the line on which the interrupt occurs.



The horizontal interrupt is controlled by a line counter in register #10.

If this line counter is changed at each interrupt, the desired spacing of interrupts may be achieved.

Thus: If register #10 equals 00h, then the interrupt occurs every line.

If register #10 equals 01h, then the interrupt occurs every other line.

If register #10 equals 02h, then the interrupt occurs every third line.

3. EXTERNAL INTERRUPT (EX-INT)

The external interrupt is generated by a peripheral device (gun, modem) and stops the H, V counter for later examination by the CPU.

HL INPUT PIN

▲
INTERRUPT HAPPEN (COUNTER RATCHED)

Please see other sections of this manual for information about the H, V counter and the initialization of the external interrupt.

E. VDP PORT

The VDP ports are at location 68000 in the 68000 memory space.

	UPPER BYTE	LOWER BYTE
\$ C 0 0 0 0 0	DATA PORT	
\$ C 0 0 0 0 2	"	
\$ C 0 0 0 0 4	CONTROL PORT	
\$ C 0 0 0 0 6	"	
\$ C 0 0 0 0 8	H/V COUNTER	
\$ C 0 0 0 0 A	PROHIBITED	
\$ C 0 0 0 0 C	PROHIBITED	
\$ C 0 0 0 0 E	PROHIBITED	
\$ C 0 0 0 1 0	PROHIBITED	PSG

1. \$ C 0 0 0 0 0 (DATA PORT)

READ/WRITE: VRAM, VSRAM, CRAM

\$C00000	DT15	DT14	DT13	DT12	DT11	DT10	DT9	DT8	(D15~D8)
	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0	(D7~D0)

* \$C00000 and \$C00002 are functionally equivalent.

2. \$C00004 (CONTROL PORT)

READ: STATUS REGISTER

\$C00004	*	*	*	*	*	*	EMPT	FULL	(D15~D8)
	F	SOVR	C	ODD	VB	HB	DMA	PAL	(D7~D0)

* NO USE

EMPT... 1: WRITE FIFO EMPTY

0:

FULL... 1: WRITE FIFO FULL

0:

F... 1: V interrupt happened

SOVR... 1: Sprites overflow occurred, too many in one line

Over 17 in 32 cell mode.

Over 21 in 40 cell mode.

C... 1: Collision happened between non-zero pixels in two sprites.

0:

ODD... 1: Odd frame in interlace mode.

0: Even frame in interlace mode.

VB... 1: During V blanking

0:

HB... 1: During H blanking

0:

DMA... 1: DMA BUSY

0:

PAL... 1: PAL MODE

0: NTSC MODE

WRITE1: REGISTER SET

\$C00004	1	0	0	RS4	RS3	RS2	RS1	RS0	(D15~D8)
	D7	D6	D5	D4	D3	D2	D1	D0	(D7~D0)

* \$C00004 and \$C00006 are functionally equivalent.

RS4 ~ RS0: Register No.

D7 ~ D0: Data

* You must use word or long word access to the VDP ports when setting the registers.
Long word access is equivalent to two word accesses, with D31 - D16 written first.

WRITE2: ADDRESS SET

1st \$ C 0 0 0 0 4	CD1	CD0	A13	A12	A11	A10	A9	A8	(D15~D8)
	A7	A6	A5	A4	A3	A2	A1	A0	(D7~D0)

2st \$ C 0 0 0 0 4	0	0	0	0	0	0	0	0	(D15~D8)
	CD5	CD4	CD3	CD2	0	0	A15	A14	(D7~D0)

CD5 ~ CD0: ID CODE

A15 ~ A0: DESTINATION RAM ADDRESS

ACCESS MODE	CD5	CD4	CD3	CD2	CD1	CD0
VRAM WRITE	0	0	0	0	0	1
CRAM WRITE	0	0	0	0	1	1
VSRAM WRITE	0	0	0	1	0	1
VRAM READ	0	0	0	0	0	0
CRAM READ	0	0	1	0	0	0
VSRAM READ	0	0	0	1	0	0

* You must use word or long word when performing these operations.

3. \$ C 0 0 0 0 8 (HV COUNTER)

NON INTERLACE MODE

\$ C 0 0 0 0 8	VC7	VC6	VC5	VC4	VC3	VC2	VC1	VC0	(D15~D8)
	HC8	HC7	HC6	HC5	HC4	HC3	HC2	HC1	(D7~D0)

INTERLACE MODE

\$ C 0 0 0 0 8	VC7	VC6	VC5	VC4	VC3	VC2	VC1	VC8	(D15~D8)
	HC8	HC7	HC6	HC5	HC4	HC3	HC2	HC1	(D7~D0)

HC8 ~ HC1: H COUNTER

VC8 ~ VC0: V COUNTER

F. VDP REGISTER

VDP has write only register No. 0~No. 23 and read only status register totalling 25 register. There are two modes for register setting; one is mode 4, the other mode 5. We tell you about mode 5 in this section and about mode 4 in MARK III section.

If you change mode in one frame you can get various effects.

1. MODE SET REGISTER NO.1

REG. #0	MSB	0	0	0	IE1	0	1	M3	0	LSB
---------	-----	---	---	---	-----	---	---	----	---	-----

IE1 ... 1: Enable H interrupt (68000 LEVEL 4)

0: Disable H interrupt (REG #10)

M3 ... 1: H, V counter stop

0: Enable read, H, V counter

2. MODE SET REGISTER NO.2

REG. #1	MSB	0	DISP	IE0	M1	M2	1	0	0	LSB
---------	-----	---	------	-----	----	----	---	---	---	-----

DISP ... 1: Enable Display

0: Disable Display

IE0 ... 1: Enable V interrupt (68000 LEVEL 6)

0: Disable V interrupt

M1 ... 1: DMA Enable

0: DMA Disable

M2 ... 1: V 30 cell mode (PAL mode)

0: V 28 cell mode (PAL mode; always 0 in NTSC mode)

3. PATTERN NAME TABLE BASE ADDRESS FOR SCROLL A

REG. #2	MSB	0	0	SA15	SA14	SA13	0	0	0	LSB
---------	-----	---	---	------	------	------	---	---	---	-----

VRAM ADDR %XXX0_0000_0000_0000

4. PATTERN NAME TABLE BASE ADDRESS FOR WINDOW

REG. #3	MSB	0	0	WD15	WD14	WD13	WD12	WD11	0	LSB
---------	-----	---	---	------	------	------	------	------	---	-----

WD11 should be 0 in H40 cell mode

VRAM ADDR %XXXX_X000_0000_0000 (H32 cell mode)

VRAM ADDR %XXXX_0000_0000_0000 (H40 cell mode)

5. PATTERN NAME TABLE BASE ADDRESS FOR SCROLL B

	MSB	LSB						
REG. #4	0	0	0	0	0	SB15	SB14	SB13

VRAM ADDR %XXX0_0000_0000_0000

6. SPRITE ATTRIBUTE TABLE BASE ADDRESS

	MSB	LSB						
REG. #5	0	AT15	AT14	AT13	AT12	AT11	AT10	AT9

AT9 should be 0 in H40 cell mode

VRAM ADDR %XXXX_XXX0_0000_0000 (32 cell)

VRAM ADDR %XXXX_XX00_0000_0000 (40 cell)

	MSB	LSB					
REG. #6	0	0	0	0	0	0	0

7. BACKGROUND COLOR

	MSB	LSB						
REG. #7	0	0	CPT1	CPT0	COL3	COL2	COL1	COL0

CPT 1,0: COLOR PALLET

COL 3~0: COLOR CODE

	MSB	LSB					
REG. #8	0	0	0	0	0	0	0

	MSB	LSB					
REG. #9	0	0	0	0	0	0	0

8. H INTERRUPT REGISTER

	MSB	LSB						
REG. #10	HIT7	HIT6	HIT5	HIT4	HIT3	HIT2	HIT1	HIT0

This register makes H interrupt timing by number of faster.

H interrupt is enabled by IE = 1.

9. MODE SET REGISTER NO. 3

REG. #11	MSB	LSB
	0 0 0 0 IE2	VSCR HSCR LSCR

IE2... 1: Enable external interrupt (68000 LEVEL 2)
2: Disable external interrupt

* See INTERRUPT and SYSTEM I/O

VSCR: V scroll mode

VSCR	FUNCTION
0	FULL SCROLL
1	EACH 2CELL SCROLL

HSCR, LSCR: H scroll mode

HSCR	LSCR	FUNCTION
0	0	FULL SCROLL
0	1	PROHIBITED
1	0	EACH 1CELL SCROLL
1	1	EACH 1LINE SCROLL

* BOTH SCROLL A and B

10. MODE SET REGISTER NO. 4

REG. #12	MSB	LSB
	RS0 0 0 0 S/TE	LSM1 LSM0 RS1

RS0... 0: HORIZONTAL 32 CELL MODE
1: HORIZONTAL 40 CELL MODE

RS1... 0: HORIZONTAL 32 CELL MODE
1: HORIZONTAL 40 CELL MODE

* You should set same No. in RSO, RS1.

32 cell 0000_XXX0
40 cell 1000_XXX1

S/TE ... 1: Enable SHADOW and HIGHLIGHT
0: Disable SHADOW and HIGHLIGHT

LSM1, LSM0: Interlace mode setting

LSM1	LSM0	FUNCTION
0	0	NO INTERLACE
0	1	INTERLACE
1	0	PROHIBITED
1	1	INTERLACE (DOUBLE RESOLUTION)

11. H SCROLL DATA TABLE BASE ADDRESS

	MSB									LSB
REG. #13	0	0	HS15	HS14	HS13	HS12	HS11	HS10		

VRAM ADDR %XXXX_XX00_0000_0000

	MSB									LSB
REG. #14	0	0	0	0	0	0	0	0		

12. AUTO INCREMENT DATA

This register controls bias number of increment data.

	MSB									LSB
REG. #15	INC7	INC6	INC5	INC4	INC3	INC2	INC1	INC0		

INC7~0: Bias number (0~\$FF)

This number is added automatically after RAM access.

13. SCROLL SIZE

	MSB									LSB
REG. #16	0	0	VSZ1	VSZ2	0	0	HSZ1	HSZ0		

VSZ1, 0: VSIZE

VSZ1	VSZ0	FUNCTION
0	0	V 32 cell
0	1	V 64 cell
1	0	PROHIBITED
1	1	V 128 cell

HSZ1, 0: HSIZ

HSZ1	HSZ0	FUNCTION
0	0	H 32 cell
0	1	H 64 cell
1	0	PROHIBITED
1	1	H 128 cell

* Both of scroll A and B

14. WINDOW H POSITION

	MSB									LSB
REG. #17	RIGT	0	0	WHP5	WHP4	WHP3	WHP2	WHP1		

RIGT... 0: Window is in left side from base point.

1: Window is in right side from base point.

WHP5~1 Base pointer 0 = Left side

1 = 1 cell right

2 ...

15. WINDOW V POSITION

	MSB								LSB
REG. #18	DOWN	0	0	WVP4	WVP3	WVP2	WVP1	WVP0	

DOWN ... 0: Window is in upper side from base point.

1: Window is in lower side from base point.

WPV4~0 Base pointer 0 = Upper side

1 = 1 cell down

2 ...

16. DMA LENGTH COUNTER LOW

	MSB								LSB
REG. #19	LG7	LG6	LG5	LG4	LG3	LG2	LG1	LG0	

17. DMA LENGTH COUNTER HIGH

	MSB								LSB
REG. #20	LG15	LG14	LG13	LG12	LG11	LG10	LG9	LG8	

LG15~0: DMA LENGTH COUNTER

18. DMA SOURCE ADDRESS LOW

	MSB								LSB
REG. #21	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	

19. DMA SOURCE ADDRESS MID.

	MSB								LSB
REG. #22	SA16	SA15	SA14	SA13	SA12	SA11	SA10	SA9	

20. DMA SOURCE ADDRESS HIGH

	MSB								LSB
REG. #23	DMD1	DMD0	SA22	SA21	SA20	SA19	SA18	SA17	

SA22~1: DMA SOURCE ADDRESS

DMD1, 0: DMA MODE

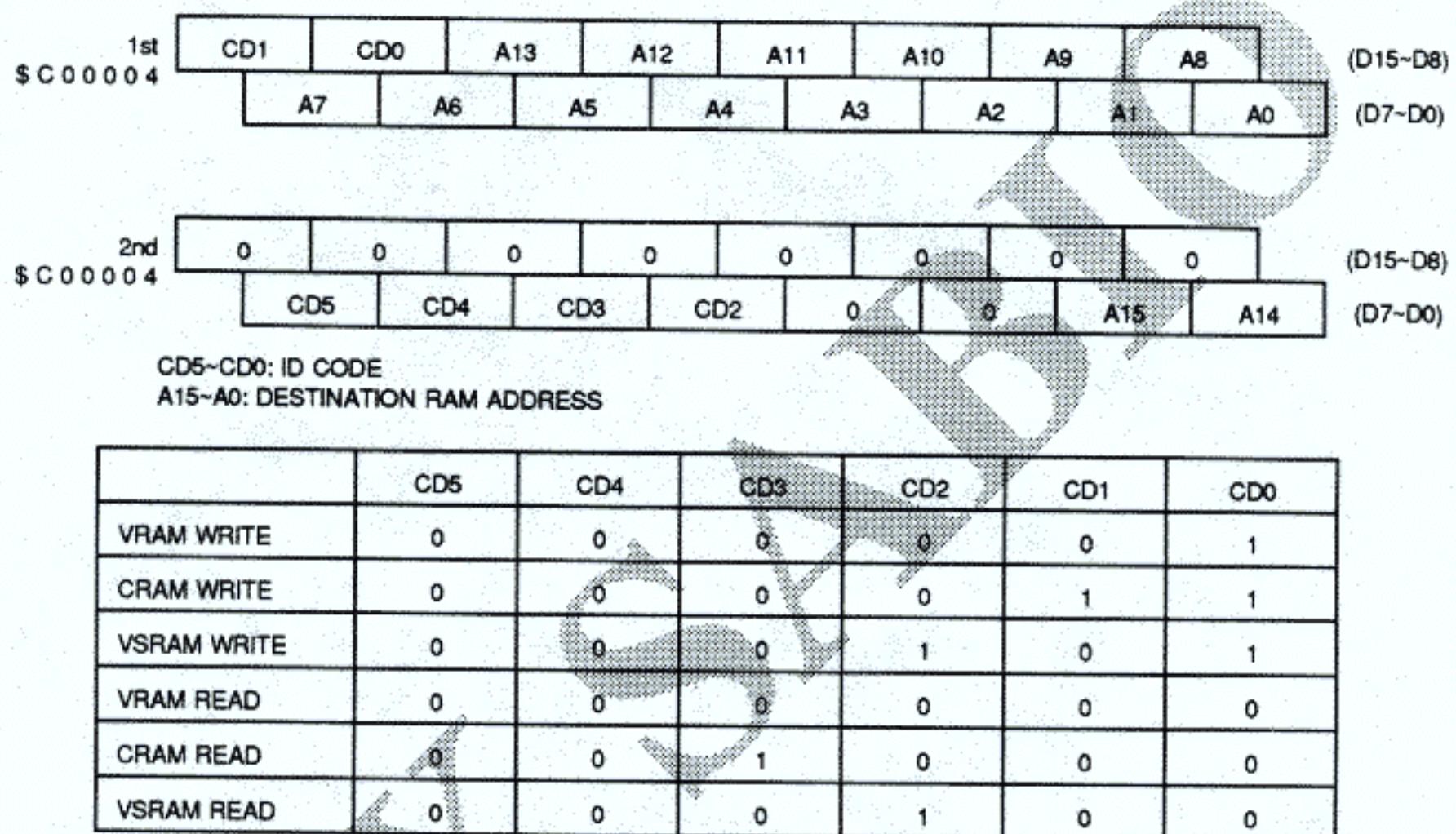
DMD1	DMD0	FUNCTION
0	SA23	MEMORY TO VRAM
1	0	VRAM FILL
1	1	VRAM COPY

G. ACCESS VDP RAM

1. RAM ADDRESS SETTING

You can access VRAM, CRAM and VSRAM after writing 32 bits of control data to \$C00004 or \$C00006.

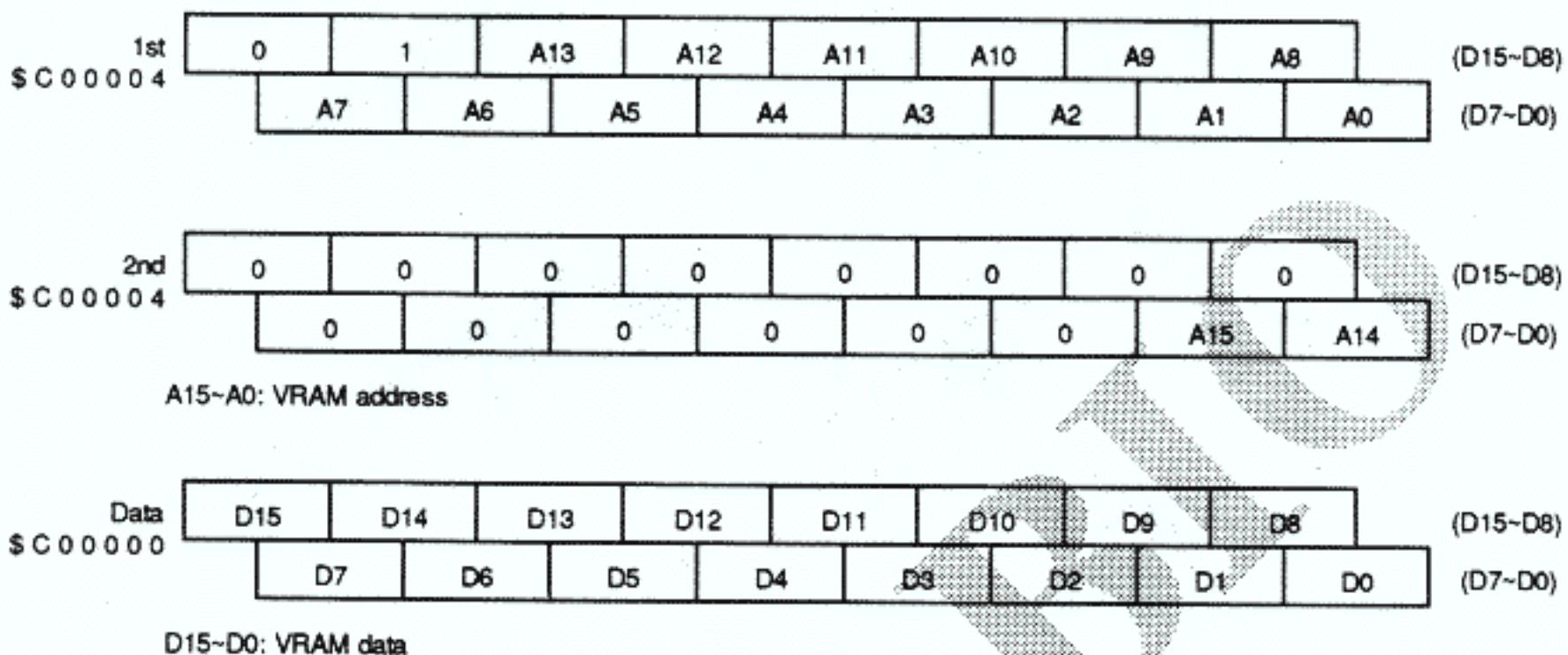
You have to use word or long word when addressing. If you use long word D31~D16 is 1st, D15~D0 2nd.



2. VRAM ACCESS

VRAM address range from 0 to 0FFFFH, 64K bytes total.

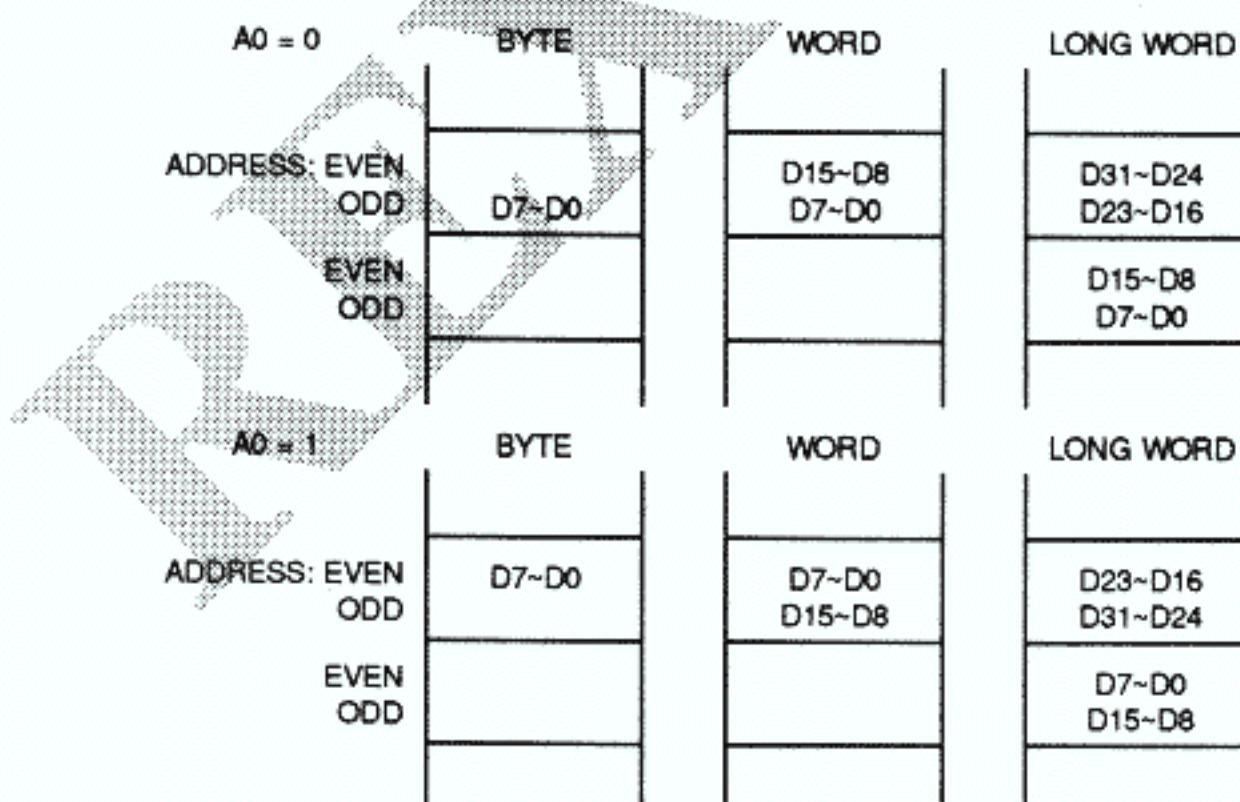
VRAM access addressing is as follows when writing:



When you use long word D31~D16 is 1st, D15~D0 is 2nd. When you do byte writing, data is D7~D0, and may be written to \$C00000 or \$C00004. *word and longword only*

VRAM address is increased by the value of REGISTER #15, independent data size. VRAM address A0 is used in the calculation of the address increment, but is ignored during address decoding.

VRAM addressing and decoding are as follows: the VRAM address decode uses A15~A1, and A0 specifies the data write format. Write data cannot cross a word boundary, high and low bytes are exchanged if A0=1.



(EXAMPLE) START ADDRESS: 0 REG. #15=2

		BYTE	WORD	LONG WORD
ADDRESS: 0				
1	1st	D7~D0	1st D15~D8 D7~D0	1st D31~D24 D23~D16
2			2nd D15~D8 D7~D0	2nd D31~D24 D23~D16
3	2nd	D7~D0	3rd D15~D8 D7~D0	3rd D31~D24 D23~D16
4			4th D15~D8 D7~D0	4th D31~D24 D23~D16
5	3rd	D7~D0	5th D15~D8 D7~D0	5th D31~D24 D23~D16
6				
7	4th	D7~D0		
8				
9	5th	D7~D0		

START ADDRESS: 0 REG. #15=1

		BYTE	WORD	LONG WORD
ADDRESS: 0				
1	2nd	D7~D0	2nd D7~D0 D15~D8	1st D7~D0 D15~D8
	1st	D7~D0		
2	4th	D7~D0	4th D7~D0 D15~D8	2nd D7~D0 D15~D8
3	3rd	D7~D0		3rd D7~D0 D15~D8
4	6th	D7~D0	6th D7~D0 D15~D8	4th D7~D0 D15~D8
5	5th	D7~D0		5th D7~D0 D15~D8
6	8th	D7~D0	8th D7~D0 D15~D8	
7	7th	D7~D0		
8	10th	D7~D0	10th D7~D0 D15~D8	
9	9th	D7~D0		

START ADDRESS: 1 REG. #15=2

ADDRESS:	BYTE		WORD		LONG WORD	
	1st	D7~D0	1st	D7~D0 D15~D8	1st	D23~D16 D31~D24
1	2nd	D7~D0	2nd	D7~D0 D15~D8	1st	D23~D16 D31~D24
3	3rd	D7~D0	3rd	D7~D0 D15~D8	2nd	D23~D16 D31~D24
5	4th	D7~D0	4th	D7~D0 D15~D8	2nd	D23~D16 D31~D24
7	5th	D7~D0	5th	D7~D0 D15~D8	3rd	D23~D16 D31~D24
9						

START ADDRESS: 1 REG. #15=1

ADDRESS:	BYTE		WORD		LONG WORD	
	1st	D7~D7	1st	D7~D0 D15~D8	1st	D23~D16 D31~D24
1	3rd	D7~D7	3rd	D7~D0 D15~D8	2nd	D23~D16 D31~D24
3	2nd	D7~D7	5th	D7~D0 D15~D8	3rd	D23~D16 D31~D24
5	5th	D7~D7	7th	D7~D0 D15~D8	4th	D23~D16 D31~D24
7	4th	D7~D7	9th	D7~D0 D15~D8	5th	D23~D16 D31~D24
9	7th	D7~D7				
	6th	D7~D7				
	8th	D7~D7				
	9th	D7~D7				
	8th	D7~D7				

VRAM READ

1st \$ C00004	0	0	A13	A12	A11	A10	A9	A8	(D15~D8)
	A7	A6	A5	A4	A3	A2	A1	A0	(D7~D0)
2nd \$ C00004	0	0	0	0	0	0	0	0	(D15~D8)
	0	0	0	0	0	0	0	A15	(D7~D0)
A15~A0: VRAM ADDRESS									
Data: \$ C00000	D15	D14	D13	D12	D11	D10	D9	D8	(D15~D8)
	D7	D6	D5	D4	D3	D2	D1	D0	(D7~D0)
D15~D0: VRAM DATA									

The data is always read in word units. A0 is ignored during the reading; no swap of bytes occurs if A0=1.

Subsequent reads are from address incremented by REGISTER #15. A0 is used in calculation of the next address.

3. CRAM ACCESS

The CRAM contains 128 bytes, addresses 0 to 7FH. For word wide writes to the CRAM, use:

1st \$ C00004	1	1	0	0	0	0	0	0	(D15~D8)
	0	A6	A5	A4	A3	A2	A1	A0	(D7~D0)
2nd \$ C00004	0	0	0	0	0	0	0	0	(D15~D8)
	0	0	0	0	0	0	0	0	(D7~D0)
A6~A0: CRAM ADDRESS									
Data: \$ C00000	0	0	0	0	B2	B1	B0	0	(D15~D8)
	G2	G1	G0	0	R2	R1	R0	0	(D7~D0)

D15~D0 are valid when we use word for data set. If the writes are byte wide, write the high byte to \$C00000 and the low byte to \$C00001.

A long word wide access is equivalent to two sequential word wide accesses. Place the first data in D31 - D16, and the second data in D15 - D0.

The data may be written sequentially; the address is incremented by the value of REGISTER #15 after every write, independent of whether the width is byte or word.

Note that A0 is used in the increment, but not in address decoding, resulting in some interesting side effects if writes are attempted at odd addresses.

For word wide reads from the CRAM, use:

1st	\$C00004	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>(D15~D8)</td></tr><tr><td>0</td><td>A6</td><td>A5</td><td>A4</td><td>A3</td><td>A2</td><td>A1</td><td>A0</td><td>(D7~D0)</td></tr></table>	0	0	0	0	0	0	0	0	(D15~D8)	0	A6	A5	A4	A3	A2	A1	A0	(D7~D0)
0	0	0	0	0	0	0	0	(D15~D8)												
0	A6	A5	A4	A3	A2	A1	A0	(D7~D0)												

2nd	\$C00004	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>(D15~D8)</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>(D7~D0)</td></tr></table>	0	0	0	0	0	0	0	0	(D15~D8)	0	0	0	0	0	0	0	0	(D7~D0)
0	0	0	0	0	0	0	0	(D15~D8)												
0	0	0	0	0	0	0	0	(D7~D0)												

A6~A0: CRAM ADDRESS

Data	\$C00000	<table border="1"><tr><td>*</td><td>*</td><td>*</td><td>*</td><td>B2</td><td>B1</td><td>B0</td><td>*</td><td>(D15~D8)</td></tr><tr><td>G2</td><td>G1</td><td>G0</td><td>*</td><td>R2</td><td>R1</td><td>R0</td><td>*</td><td>(D7~D0)</td></tr></table>	*	*	*	*	B2	B1	B0	*	(D15~D8)	G2	G1	G0	*	R2	R1	R0	*	(D7~D0)
*	*	*	*	B2	B1	B0	*	(D15~D8)												
G2	G1	G0	*	R2	R1	R0	*	(D7~D0)												

4. VSRAM ACCESS

The VSRAM contains 80 bytes, addresses 0 to 4FH. For word wide writes to the VSRAM, use:

1st	\$C00004	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>(D15~D8)</td></tr><tr><td>0</td><td>A6</td><td>A5</td><td>A4</td><td>A3</td><td>A2</td><td>A1</td><td>A0</td><td>(D7~D0)</td></tr></table>	0	0	0	0	0	0	0	0	(D15~D8)	0	A6	A5	A4	A3	A2	A1	A0	(D7~D0)
0	0	0	0	0	0	0	0	(D15~D8)												
0	A6	A5	A4	A3	A2	A1	A0	(D7~D0)												

2nd	\$C00004	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>(D15~D8)</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>(D7~D0)</td></tr></table>	0	0	0	0	0	0	0	0	(D15~D8)	0	0	0	0	1	0	0	0	(D7~D0)
0	0	0	0	0	0	0	0	(D15~D8)												
0	0	0	0	1	0	0	0	(D7~D0)												

A6~A0: VSRAM ADDRESS

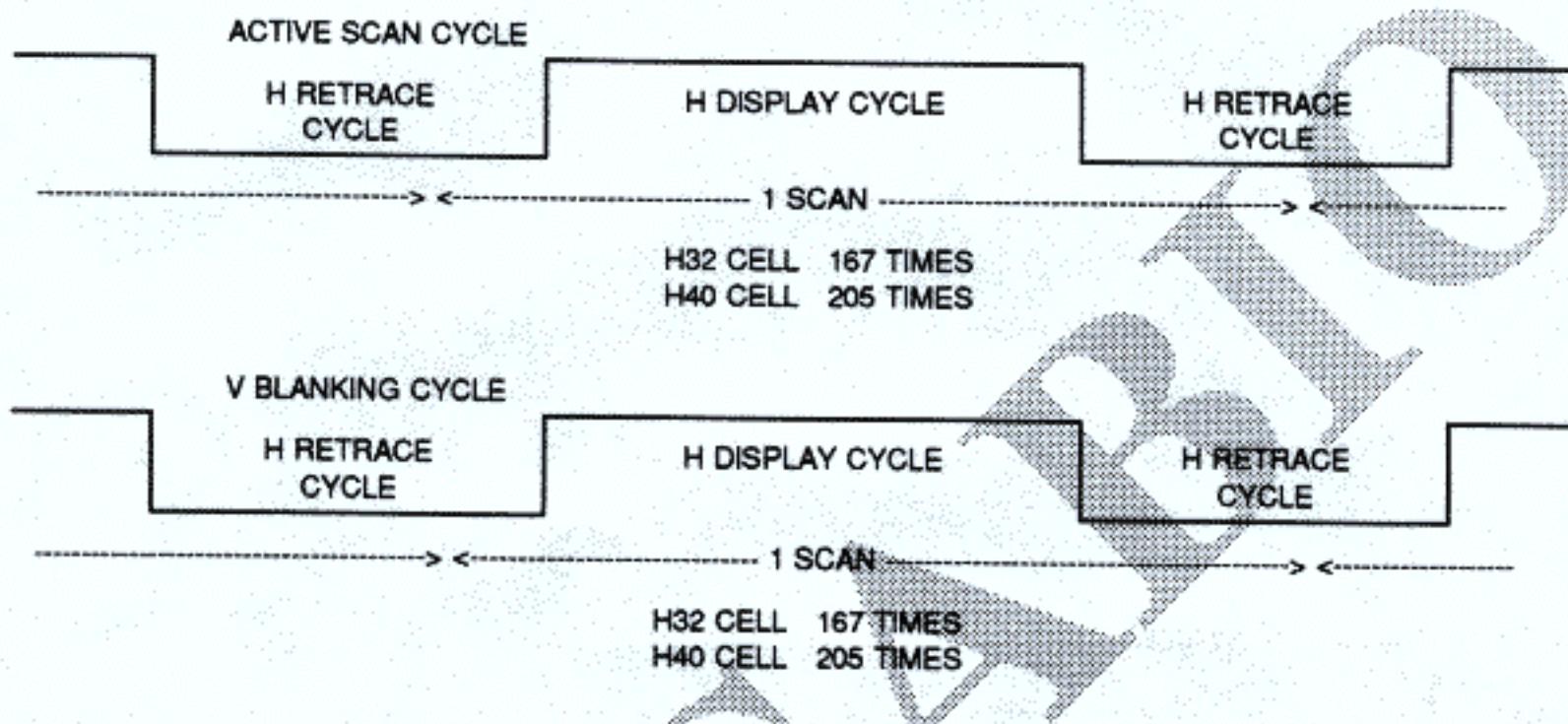
Data	\$C00000	<table border="1"><tr><td></td><td></td><td></td><td></td><td></td><td>VS10</td><td>VS9</td><td>VS8</td><td>(D15~D8)</td></tr><tr><td>VS7</td><td>VS6</td><td>VS5</td><td>VS4</td><td>VS3</td><td>VS2</td><td>VS1</td><td>VS0</td><td>(D7~D0)</td></tr></table>						VS10	VS9	VS8	(D15~D8)	VS7	VS6	VS5	VS4	VS3	VS2	VS1	VS0	(D7~D0)
					VS10	VS9	VS8	(D15~D8)												
VS7	VS6	VS5	VS4	VS3	VS2	VS1	VS0	(D7~D0)												

VS10~VS0: V quantity of scroll

5. ACCESS TIMING

The CPU and VDP access VRAM, CRAM and VSRAM using timesharing. Because the VDP is very busy during the active scan, the CPU accesses are limited. However, during vertical blanking, the CPU may access the VDP continuously.

The number of permitted accesses by the CPU additionally depends on whether the screen is in 32 cell mode or 40 cell mode. Additionally, the access size depends on the RAM type; a VRAM access is byte wide, but CRAM and VSRAM are word wide.



For example, in 32 Cell mode, the CPU may access the VRAM 16 times during horizontal scan in a single line. Each access is a byte write, so this amounts to 8 words. However, CRAM and VSRAM, though sharing the 16 time limit, are word accesses so that 16 words may be written in a single line.

Although there is a four-word FIFO, if writes are done in a tight loop during active scan, the FIFO will fill up and the CPU will eventually end up waiting to write.

The maximum wait times are:

Display Mode	Maximum Waiting Time
H32 cell	Approximately 5.96 μ sec
H40 cell	Approximately 4.77 μ sec

As the CPU has unlimited access to the RAMs during vertical blanking, the wait case never arises.

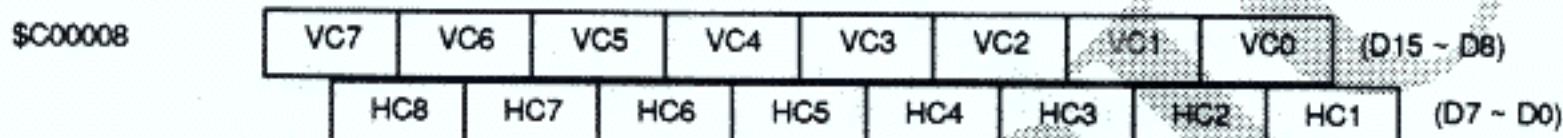
6. HV COUNTER

The HV Counter's function is to give the horizontal and vertical location of the television beam. If the "M3" Bit of Register #0 is set, the HV Counter will then freeze when trigger signal HL goes high, as well as triggering a level 2 interrupt.

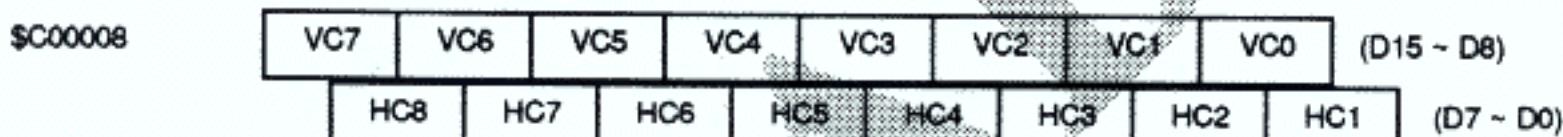
M3	Counter Latch Mode
0	Counter is not latched by trigger signal
1	Counter is latched by trigger signal

M3: Register #0

Non-Interlace Mode



Interlace Mode



V-Counter: VC7 ~ VC0

Display Mode	Counter Data
V 28 Cell	0 ~ D9H
V 30 Cell	0 ~ E9H

H-Counter: HC8 ~ HC1

Display Mode	Counter Data
H 32 Cell	0 ~ 7FH
H 40 Cell	0 ~ 9FH

The counter only has eight bits each for H and V, so Interlace Mode and 40 Cell (320 dot) modes present some problems. During Interlace Mode, the LSB of the vertical position is replaced by the new MSB. The horizontal resolution problem is solved by always dropping the LSB.

Caution:

As the HV Counter's value is not valid during vertical blanking, check to be sure that it is active scan before using the value.

H. DMA TRANSFER

DMA (Direct Memory Access) is a high-speed technique for memory access to the VRAM, CRAM, and VSRAM. During DMA, VRAM, CRAM and VSRAM occur at the fastest possible rate (refer to the section on Access Timing). There are three modes of DMA access, as can be seen below, all of which may be done to VRAM or CRAM or VSRAM. The 68K is stopped during memory to VRAM/CRAM/VSRAM DMA, but the Z80 continues to run as long as it does not attempt access to the 68K memory space.

The DMA is quite fast during VBLANK (about double the tightest possible 68K loop's speed), but during active scan the speed is the same as a 68K loop.

Please note that after this point, VRAM is used as a generic term for VRAM/CRAM/VSRAM.

DMD1	DMD0	DMA Mode	Size
0	SA23	A. Memory to V-RAM	Word to Byte (H) & (L)
1	0	B. VRAM Fill	Byte to Byte
1	1	C. VRAM Copy	Byte to Byte

DMD1, DMD0: Reg. #23 * DMD0 = SA23

Source address are \$000000~\$3FFFF (ROM) and \$FF0000~\$FFFFFF (RAM) for memory to VRAM transfers. In the case of ROM to VRAM transfers, a hardware feature causes occasional failure of DMA unless the following two conditions are observed:

- The destination address write (to address \$C00004) must be a word write.
- The final write must use the work RAM. There are two ways to accomplish this: (1) by copying the DMA program into RAM, or (2) by doing a final "move W RAM address, \$C00004".

1. MEMORY TO VRAM

This function transfers data from 68K memory to VRAM, CRAM or VSRAM. During this DMA all 68K processing stops. The source address is \$000000~\$3FFFF for ROM or \$FF0000~\$FFFFFF for RAM. The DMA reads are word-wide, ~~writes are~~ byte-wide for VRAM and word-wide for CRAM and VSRAM. The destination is specified by:

CD2	CD1	CD0	Memory Type
0	0	1	VRAM
0	1	1	CRAM
1	0	1	VSRAM

Setting of DMA

- A. M1 (Register #1) = 1 : DMA ENABLE.
- B. Increment number set to Register #15 (normally #2).
- C. Transfer word number set to Registers #19, #20.
- D. Source address and DMA mode set into Registers #21, #22 and #23.
- E. Set the destination address.
- F. *VDP gets the CPU bus.
- G. *DMA start.
- H. *VDP releases the CPU bus.
- I. M1 has to be 0 after confirmation of DMA finish : DMA DISENABLE.

DMA starts after Step E.

You must set M1=1 only during DMA; otherwise, we cannot guarantee the operation. Source addresses were increased with +2 and destination address increased with content of Register #15.

Content of Register. Register #1 has another bit.

Register #15	INC7	INC6	INC5	INC4	INC3	INC2	INC1	INC0
--------------	------	------	------	------	------	------	------	------

INC7 ~ INC0 : Number of increment

Register #1	0	DISP	IEO	M1	M2	1	0	0
-------------	---	------	-----	----	----	---	---	---

Register #19	LG7	LG6	LG5	LG4	LG3	LG2	LG1	LG0
--------------	-----	-----	-----	-----	-----	-----	-----	-----

Register #20	LG15	LG14	LG13	LG12	LG11	LG10	LG9	LG8
--------------	------	------	------	------	------	------	-----	-----

Register #21	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1
--------------	-----	-----	-----	-----	-----	-----	-----	-----

Register #22	SA16	SA15	SA14	SA13	SA12	SA11	SA10	SA9
--------------	------	------	------	------	------	------	------	-----

Register #23	0	SA23	SA22	SA21	SA20	SA19	SA18	SA17
--------------	---	------	------	------	------	------	------	------

1st \$C00004	CD1	CD0	DA13	DA12	DA11	DA10	DA9	DA8	(D15 ~ D8)
	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	(D7 ~ D0)
2nd \$C00004	0	0	0	0	0	0	0	0	(D15 ~ D8)
	1	0	0	CD2	0	0	DA15	DA14	(D7 ~ D0)

LG15 ~ LG0: Number of move word
 SA23 ~ SA1: Source address (in 68000)
 DA15 ~ DA0: Destination address (in VDP)
 CD2 ~ CD0: RAM selection

2. VRAM FILL

FILL mode fills with same data from free even VRAM address. FILL for only VRAM.

How to set FILL (DMA):

- A. M1 (Register #1) = 1 : DMA ENABLE
- B. Increment number set to #15 (normally #1)
- C. Fill size set to #19, #20.
- D. DMA mode set to #23.
- E. Destination address and FILL data set.
- F. *DMA start.
- G. M1=0 after confirmation of finishing : DMA DISENABLE

DMA starts after Step E.

M1 should be 1 in the DMA transfer; otherwise, we cannot guarantee the operation.

Destination address is incremented with Register #15. VDP does not ask but open for CPU but CPU cannot access VDP without PSG, HV counter, and status. You can realize end of DMA by DMA bit in status register.

Register Setting. Register #1 has another bit.

Register #15

INC7	INC6	INC5	INC4	INC3	INC2	INC1	INC0
------	------	------	------	------	------	------	------

INC7 ~ INC0: Increment number

Status

*	*	*	*	*	*	*	EMPT	FULL
F	SOVR	C	ODD	VB	HB	DMA	PAL	

DMA: 1: DMA Busy

*: Not care

Register #1

0	DISP	IEO	M1	M2	1	0	0
---	------	-----	----	----	---	---	---

Register #19

LG7	LG6	LG5	LG4	LG3	LG2	LG1	LG0
-----	-----	-----	-----	-----	-----	-----	-----

Register #20

LG15	LG14	LG13	LG12	LG11	LG10	LG9	LG8
------	------	------	------	------	------	-----	-----

Register #23

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

1st \$C00004

0	1	DA13	DA12	DA11	DA10	DA9	DA8	(D15 ~ D8)
DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	(D7 ~ D0)

2nd \$C00004

0	0	0	0	0	0	0	0	(D15 ~ D8)
1	0	0	0	0	0	0	DA15	DA14

\$C00000

FD15	FD14	FD13	FD12	FD11	FD10	FD9	FD8	(D15 ~ D8)
FD7	FD6	FD5	FD4	FD3	FD2	FD1	FD0	(D7 ~ D0)

LG15 ~ LG0: Fill byte number

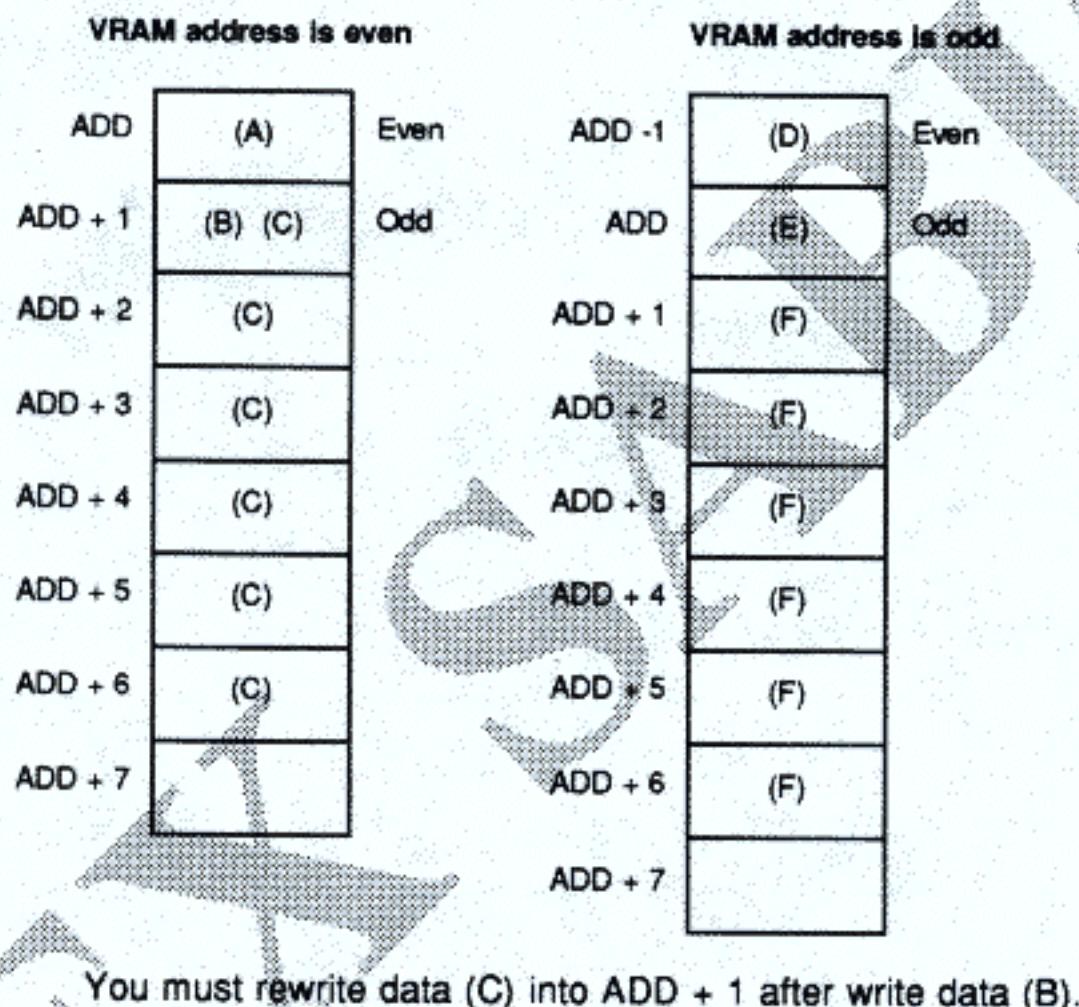
DA15 ~ DA0: Destination address

FD15 ~ FD0: FILL data

When setting first and second by long word, first will be D31-D16 and second will be D15-D0.

Example 1 Fill data are word. Register #15 = 1.

1. V-RAM address is even.
 - A. First, low side of Fill data are written in V-RAM address.
 - B. Second, upper side of Fill data are written in V-RAM+1.
 - C. V-RAM address is added register #15, written upper side Fill data in V-RAM at next each step.
2. V-RAM address is odd.
 - D. First, upper side of FILL data are written in V-RAM address -1.
 - E. Second, low side of FILL data are written in V-RAM.
 - F. Same as (C.)



Example 2 Fill data are word. Register #15 = 2.

VRAM address = even

ADD	(A) lower
ADD + 1	(B) upper
ADD + 2	(C) lower
ADD + 3	upper
ADD + 4	(C) lower
ADD + 5	upper
ADD + 6	(C) lower
ADD + 7	upper

VRAM address = odd

Even	ADD -1	(D) upper
Odd	ADD	(E) lower
	ADD + 1	
	ADD + 2	(F) upper
	ADD + 3	lower
	ADD + 4	(F) upper
	ADD + 5	lower
	ADD + 6	(F) upper
	ADD + 7	lower

Example 3 Fill data are byte.

1. V-RAM address is even
(A) = (B) = (C) = BYTE DATA
2. V-RAM address is odd
(D) = (E) = (F) = BYTE DATA

3. VRAM COPY

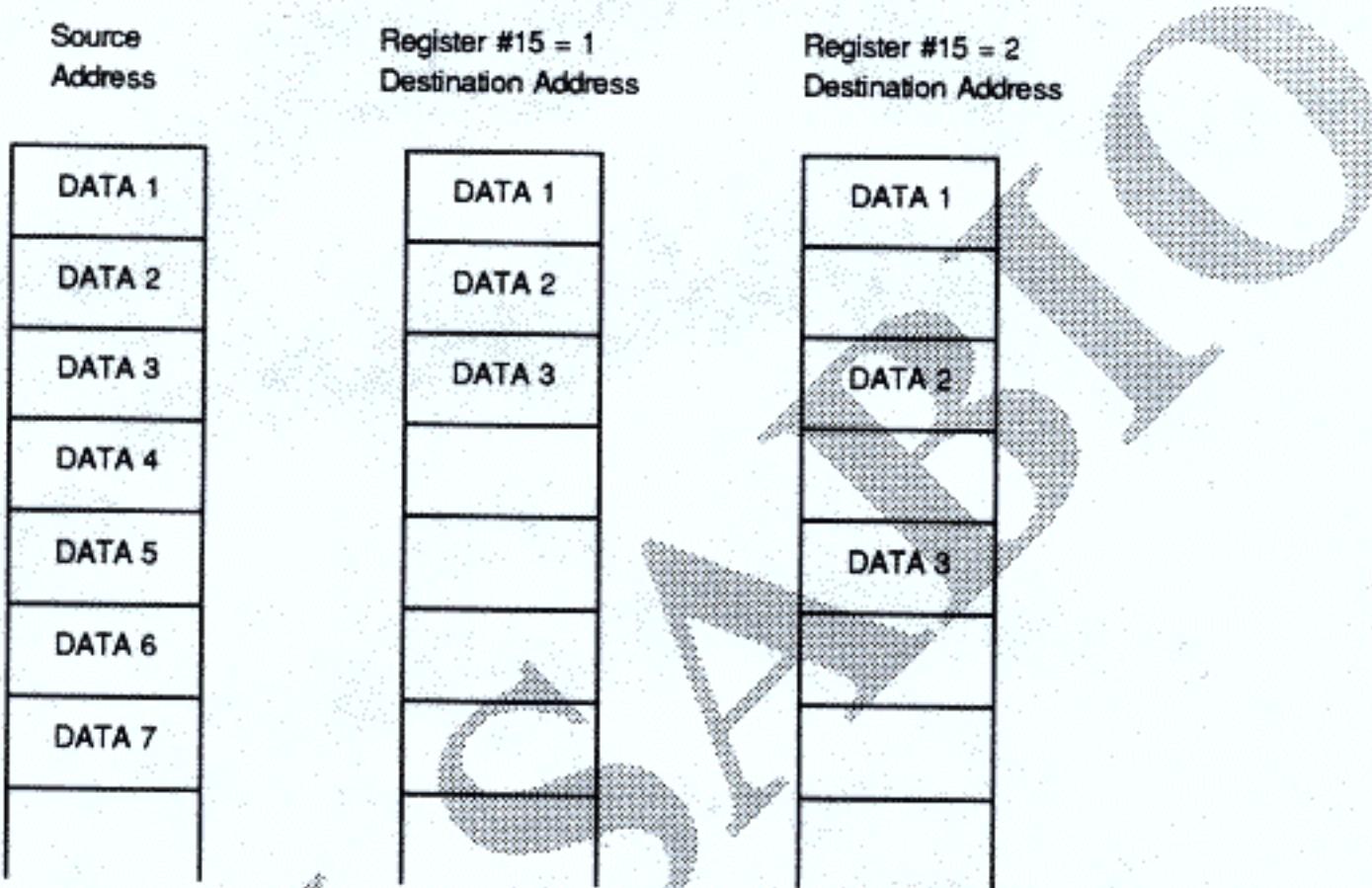
This function copies from source address to destination address by number of Copy byte.

DMA Setting

- A. M1 (Register #1) = 1 : DMA ENABLE.
- B. Number of copy bytes in #19, #20
- C. Source address and DMA mode in #23.
- D. Destination address set.
- E. *DMA transfer.
- F. After confirming DMA finish: M1 = 0 : DMA DISENABLE

DMA starts when (D) above is finished. Apply M1=1 only during DMA transfer. In other cases, if M1 = 1 is set, there is no guarantee that it will function correctly. At the time of DMA transfer, the destination address is incremented by the set value of Register #15. During DMA transfer, although the VDP does not require CPU to make a bus available, no access is possible from CPU to VDP except for PSG, HV counter, Status Read. DMA transfer finish can be recognized by referring to the Status Register's DMA bit.

Example With Transfer byte = 3 at the time of VRAM Copy



Caution

In the case of VRAM Copy, "read from VRAM" and "write to VRAM" are repeated per byte. Therefore, when the Source Area and Transfer Area are overlapped, the transfer may not be performed correctly.

Registers are as follows. Register #1 includes bits set for purposes other than DMA. Therefore, pay careful attention in this regard.

Register #15

INC7	INC6	INC5	INC4	INC3	INC2	INC1	INC0
------	------	------	------	------	------	------	------

INC7 ~ INC0: Increment number

Status

*	*	*	*	*	*	EMPT	FULL
F	SOVR	C	ODD	VB	HB	DMA	PAL

DMA: 1: DMA Busy

Register #1:

0	DISP	IEO	M1	M2	1	0	0
---	------	-----	----	----	---	---	---

Register #19

LG7	LG6	LG5	LG4	LG3	LG2	LG1	LG0
-----	-----	-----	-----	-----	-----	-----	-----

Register #20

LG15	LG14	LG13	LG12	LG11	LG10	LG9	LG8
------	------	------	------	------	------	-----	-----

Register #21

SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0
-----	-----	-----	-----	-----	-----	-----	-----

Register #22

SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8
------	------	------	------	------	------	-----	-----

Register #23

1	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---

1st \$C00004

0	0	DA13	DA12	DA11	DA10	DA9	DA8	(D15 ~ D8)
DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	(D7 ~ D0)

2nd \$C00004

0	0	0	0	0	0	0	0	(D15 ~ D8)
1	1	0	0	0	0	0	DA15	DA14

LG15 ~ LG0: Number of copy byte

SA15 ~ SA0: Source address

DA15 ~ DA0: Destination address

When setting first and second by long word, first will be D31~D16 and second will be D15~D0.

4. DMA TRANSFER CAPACITY

Transfer quantity varies, depending on the Display Mode as follows:

DMA Mode	Display Mode	Screen Scanning	Transfer bytes per line
MEMORY TO VRAM	H32 Cell	During effective screen During V Blank	16 bytes 167 bytes
	H40 Cell	During effective screen During V Blank	18 bytes 205 bytes
VRAM FILL	H32 Cell	During effective screen During V Blank	15 bytes 166 bytes
	H40 Cell	During effective screen During V Blank	17 bytes 204 bytes
VRAM COPY	H32 Cell	During effective screen During V Blank	8 bytes 83 bytes
	H40 Cell	During effective screen During V Blank	9 bytes 102 bytes

In the Memory to VRAM, in the case where CRAM and VSRAM are the destinations, number of words (not bytes) should apply. One line during V Blank allows for data transfer to all the addresses of CRAM and VSRAM.

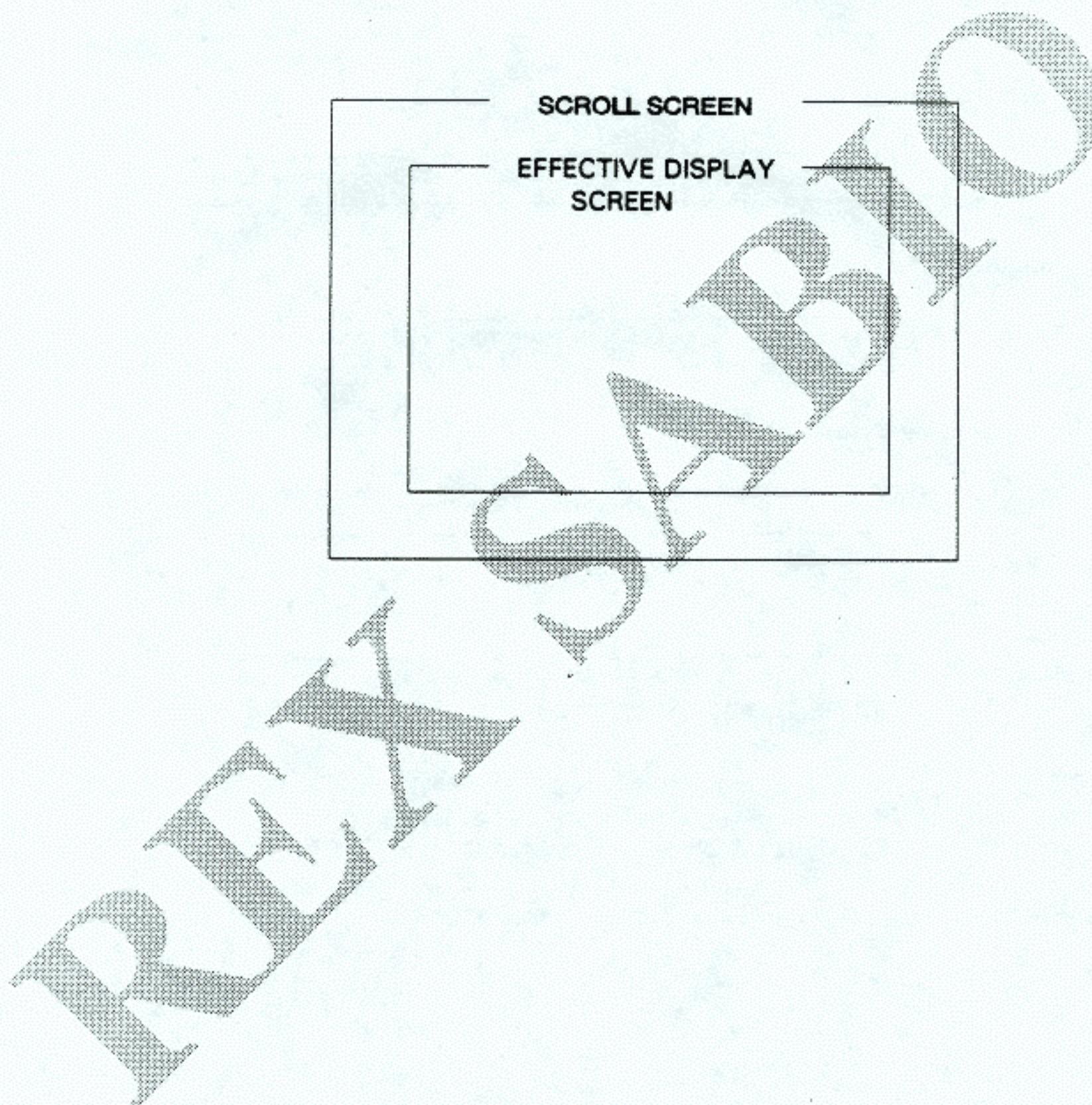
Note that when calculating, the transfer quantity in one screen (1/60 sec) varies depending on the number of lines during V Blank (refer to *Display Mode*) in the case of NTSC (video signal) and PAL systems.

Display Mode	No. of Horizontal Lines
V28 Cell (NTSC)	36
V28 Cell (PAL)	87
V30 Cell (PAL)	71

Where Register #1 DISP=0, i.e., when on-screen display is not made, the Transfer quantity is the same as Transfer Bytes Per Line during Blanking.

I. SCROLLING SCREEN

There are two different scroll screens, A and B, which separately can scroll vertically and horizontally on a basis of a one-dot unit. In the horizontal direction, scrolling overall or based on a one cell unit or one line unit can be selected, and in the vertical direction, scrolling overall or in a two cell unit can be selected. Also, the scroll screen size can be changed on a basis of a 32 cell unit.



For the scrolling screen display, the following register setting and VRAM are required:

Scroll "A" Pattern Name Table Base Address

Register #2	0	0	SA15	SA14	SA13	0	0	0
-------------	---	---	------	------	------	---	---	---

Scroll "B" Pattern Name Table Base Address

Register #4	0	0	0	0	0	SB15	SB14	SB13
-------------	---	---	---	---	---	------	------	------

Mode Set Register #3

Register #11	0	0	0	0	IE2	VSCR	HSCR	LSCR
--------------	---	---	---	---	-----	------	------	------

Mode Set Register #4

Register #12	RS0	0	0	0	S/TE	LSM1	LSM0	RS1
--------------	-----	---	---	---	------	------	------	-----

H Scroll Data Table Base Address

Register #13	0	0	HS15	HS14	HS13	HS12	HS11	HS10
--------------	---	---	------	------	------	------	------	------

Scroll Size

Register #16	0	0	VSZ1	VSZ0	0	0	HSZ1	HSZ0
--------------	---	---	------	------	---	---	------	------

VRAM

Scroll "A" Pattern Name Table

Max 8 Kbyte

Scroll "B" Pattern Name Table

Max 8 Kbyte

H Scroll Data Table

Max 960 byte

VSRAM

V Scroll Data Table

Max 80 byte

1. SCROLLING SCREEN SIZE

The screen size can be set by VSZ1, VSZ0, HSZ1, and HSZ0 (Register #16). The following six kinds can be set for both Scroll Screens A and B.

32*32/32*64/32*128

64*32/64*64

128*32

VSZ1	VSZ0	Function
0	0	V 32 cell
0	1	V 64 cell
1	0	Prohibited
1	1	V 128 cell

HSZ1	HSZ0	Function
0	0	H 32 cell
0	1	H 64 cell
1	0	Prohibited
1	1	H 128 cell

Scroll Screen's Pattern Name Table Address exists in the VRAM and is designated by Registers #2 and #4. Depending VRAM and Scroll Screen correspond to each other differently.

Example

Register #16 = 00H: 32*32 cell

		32 CELL			
0	1	~	30	31	
0	0000	0002	~	003c	003e
1	0040	0042		007c	007e
30	0780	0782	~	07bc	07be
31	07c0	07c2		07fc	07fe

Example

Register #16 = 11H: 64*64 cell

	0	1	64 CELL	62	63
0	0000	0002	~	007c	007e
1	0080	0082		00fc	00fe
64 cell					
62	1f00	1f02		1f7c	1f7e
63	1fc0	1fc2	~	1ffc	1ffe

Example

Register #16 = 03H: 32*128 cell

	0	1	128 CELL	126	127
0	0000	0002	~	00fc	00fe
1	0100	0102		01fc	01fe
32 cell					
30	1e00	1e02		1efc	1efe
31	1f00	1f02	~	1ffc	1ffe

A value shown in a frame indicates an offset from the Pattern Name Table Base Address.

2. HORIZONTAL SCROLLING

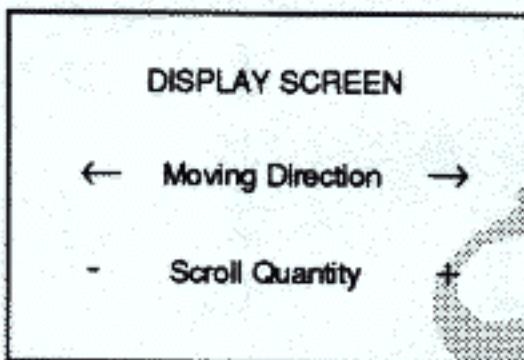
The Display Screen allows for scrolling overall or based on one cell unit, or on a dot by dot basis in one line unit. Either one of the above scrolling can be selected by HSCR and LSCR (Register #11). A setting applies to both Scroll Screens A and B.

HSCR	LSCR	FUNCTION
0	0	Overall Scrolling
0	1	Prohibited
1	0	Scroll in one cell unit
1	1	Scroll in one line unit

HSCR, LCSR: Register #11

The effective scroll quantity is equivalent to 10 bits (000H ~ 3FFFH).

Taking the Display Screen as standard, the scroll direction will be as follows:



Horizontally scrolling quantity setting area:

H Scroll Data Table is in VRAM. From the base address which was set by Register #13, set the scrolling quantity of Screens A and B alternately. Also, the scrolling quantity data setting position varies depending on the following mode (overall, 1 cell or 1 line).

Mode	Setting Position
Overall	Line 0
1 cell	Every 8th line starting from line 0
1 line	All lines

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	A - Scrolling Quantity of Screen A	Overall, Cell, Line
02	B - Scrolling Quantity of Screen B	Overall, Cell, Line
04	A - Scrolling Quantity of Screen A	Line
06	B - Scrolling Quantity of Screen B	Line
08	A - Scrolling Quantity of Screen A	Line
0A	B - Scrolling Quantity of Screen B	Line
10		
1C	A - Scrolling Quantity of Screen A	Line
1E	B - Scrolling Quantity of Screen B	Line
20	A - Scrolling Quantity of Screen A	Cell, Line
22	B - Scrolling Quantity of Screen B	Cell, Line
24		
3FC	A - Scrolling Quantity of Screen A	Line
3FE	B - Scrolling Quantity of Screen B	Line

D15~D10 can be freely utilized for program software.

3. VERTICAL SCROLLING

The Display Screen allows for scrolling overall or every 2 cells in a dot unit. The setting can be done by VSCR (Register #11). A setting applies to both Scroll Screens A and B.

VSCR	Function
0	Overall Scroll
1	2-Cell Unit Scroll

VSCR: Register #11

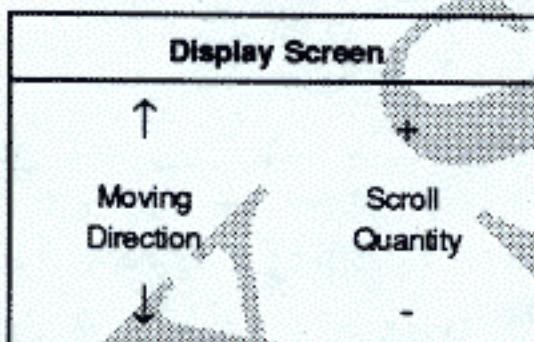
The scrolling quantity is equivalent to 11 bits (000H ~ 7FFH). However, it will be as shown below in the Interlace Mode.

Non-Interlace: The effective scrolling quantity is equivalent to 10 bits.

Interlace 1: Same as above.

Interlace 2: The effective scrolling quantity is equivalent to 11 bits.

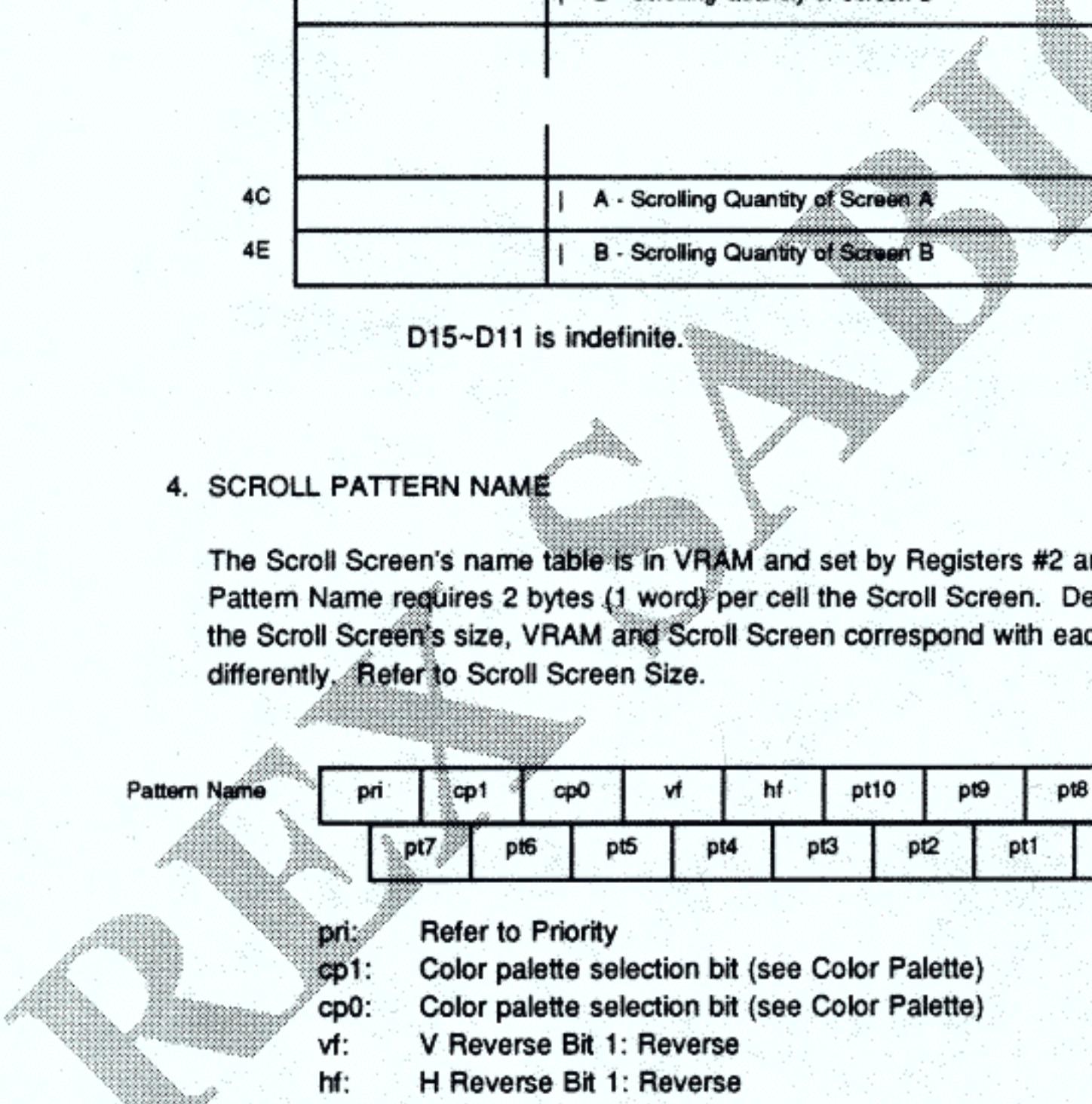
Taking the Display Screen as standard, the scrolling direction will be as follows:



Set the V Scroll quantity by VSRAM.

Alternately set the scroll quantity of Screens A and B. Depending on the Scroll Mode, the Data setting positions differ.

Mode	Setting Position
Overall	Only at the beginning
2 cell	Set to all



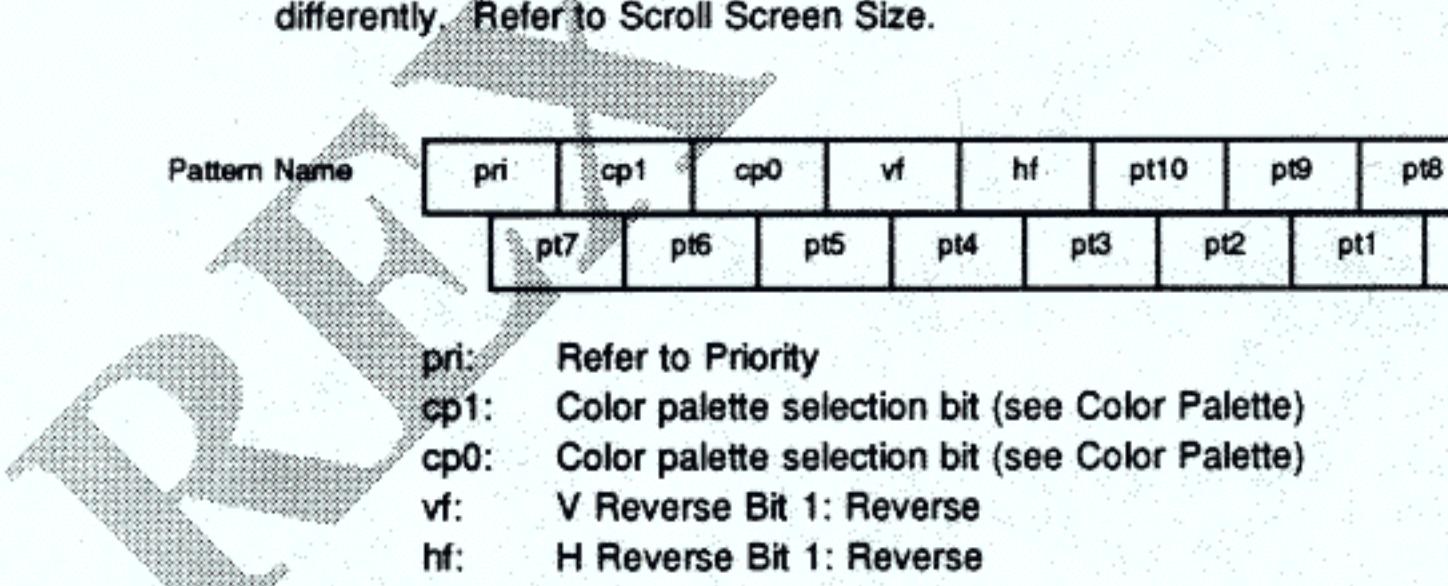
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	A - Scrolling Quantity of Screen A	0, 1 Cell, Overall
02	B - Scrolling Quantity of Screen B	0, 1 Cell, Overall
04	A - Scrolling Quantity of Screen A	2, 3 Cell
06	B - Scrolling Quantity of Screen B	2, 3 Cell
08	A - Scrolling Quantity of Screen A	4, 5 Cell
0A	B - Scrolling Quantity of Screen B	4, 5 Cell
0C	A - Scrolling Quantity of Screen A	6, 7 Cell
0E	B - Scrolling Quantity of Screen B	6, 7 Cell
4C	A - Scrolling Quantity of Screen A	38, 39 Cell
4E	B - Scrolling Quantity of Screen B	38, 39 Cell

D15~D11 is indefinite.

4. SCROLL PATTERN NAME

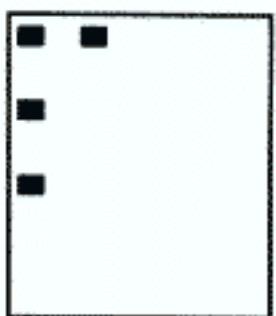
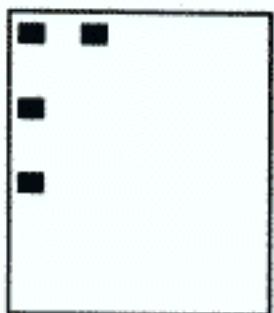
The Scroll Screen's name table is in VRAM and set by Registers #2 and #4. The Pattern Name requires 2 bytes (1 word) per cell the Scroll Screen. Depending on the Scroll Screen's size, VRAM and Scroll Screen correspond with each other differently. Refer to Scroll Screen Size.



Pattern Name	pri	cp1	cp0	vf	hf	pt10	pt9	pt8	(D15 ~ D8)
	pt7	pt6	pt5	pt4	pt3	pt2	pt1	pt0	(D7 ~ D0)

- pri: Refer to Priority
- cp1: Color palette selection bit (see Color Palette)
- cp0: Color palette selection bit (see Color Palette)
- vf: V Reverse Bit 1: Reverse
- hf: H Reverse Bit 1: Reverse
- pt10 ~ pt0: Pattern Generator Number

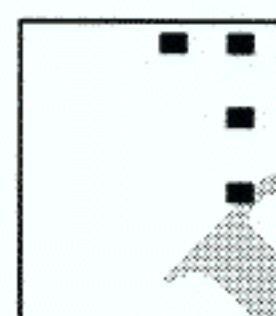
Reverse bits vf and hf allow for H and V reverse on cell unit basis.



vf = 0
hf = 0



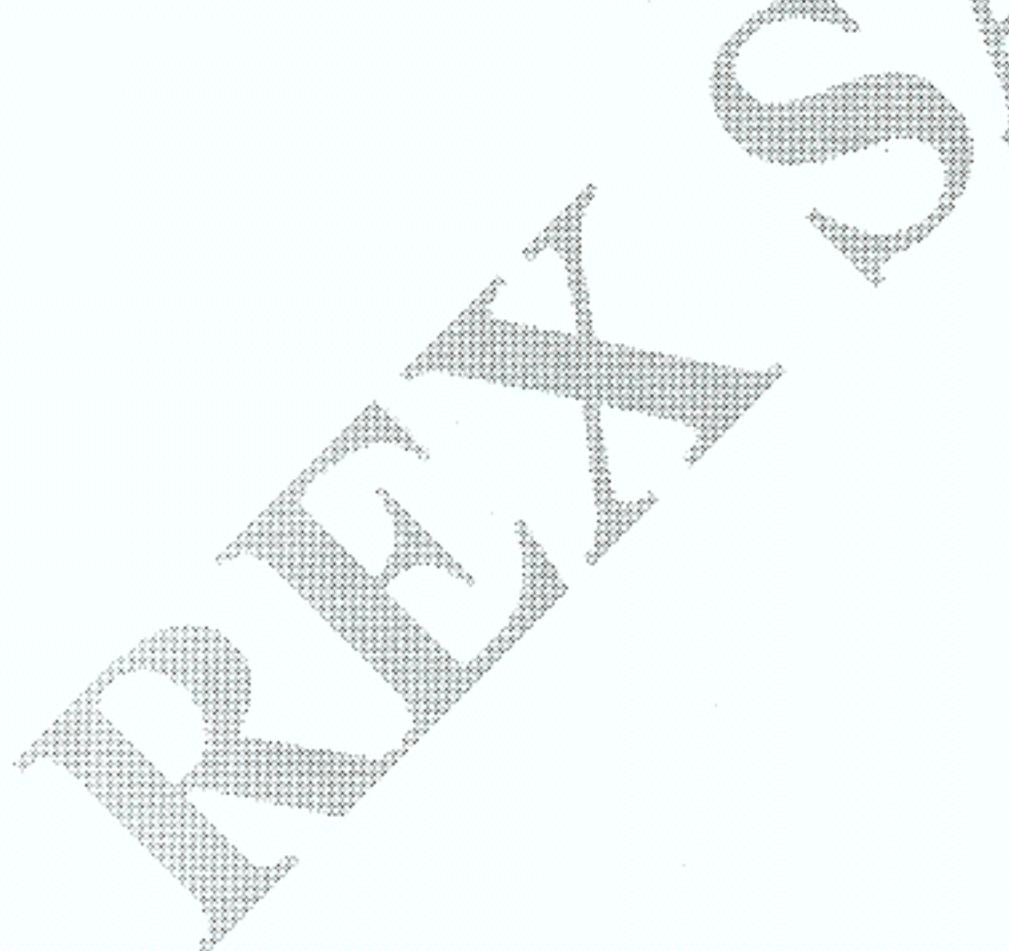
vf = 1
hf = 0



vf = 0
hf = 1



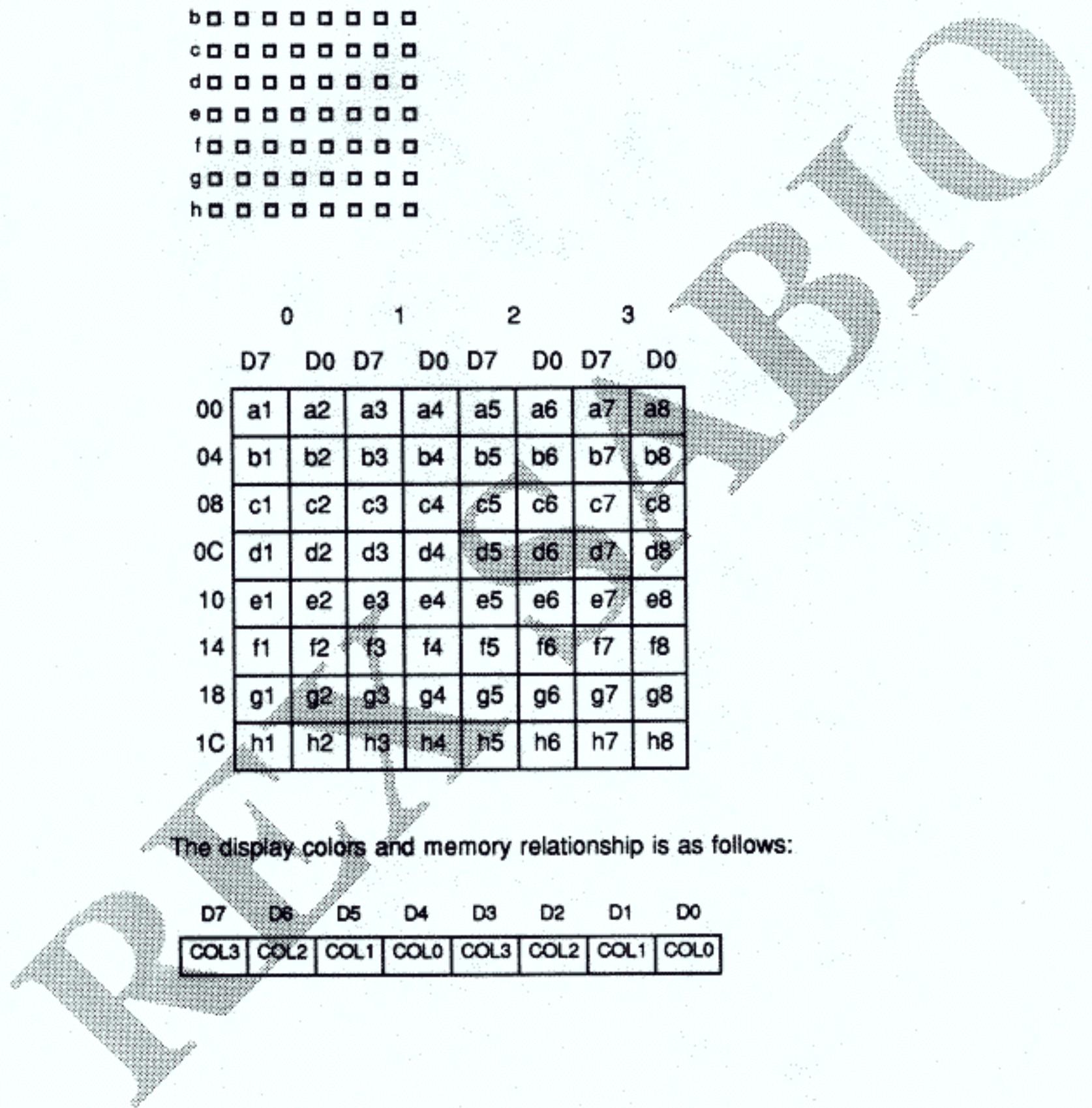
vf = 1
hf = 1



5. PATTERN GENERATOR

Pattern Generator has VRAM 0000H as base address, and a pattern is expressed on an 8 x 8 dot basis. To define a pattern, 32 bytes are required. Starting from 0000H, it proceeds in the sequence of Pattern Generator 0, 1, 2, etc. The relationship between the display pattern and memory is as follows:

1	2	3	4	5	6	7	8
a	□	□	□	□	□	□	□
b	□	□	□	□	□	□	□
c	□	□	□	□	□	□	□
d	□	□	□	□	□	□	□
e	□	□	□	□	□	□	□
f	□	□	□	□	□	□	□
g	□	□	□	□	□	□	□
h	□	□	□	□	□	□	□



	0	1	2	3				
	D7	D0	D7	D0	D7	D0	D7	D0
00	a1	a2	a3	a4	a5	a6	a7	a8
04	b1	b2	b3	b4	b5	b6	b7	b8
08	c1	c2	c3	c4	c5	c6	c7	c8
0C	d1	d2	d3	d4	d5	d6	d7	d8
10	e1	e2	e3	e4	e5	e6	e7	e8
14	f1	f2	f3	f4	f5	f6	f7	f8
18	g1	g2	g3	g4	g5	g6	g7	g8
1C	h1	h2	h3	h4	h5	h6	h7	h8

The display colors and memory relationship is as follows:

D7	D6	D5	D4	D3	D2	D1	D0
COL3	COL2	COL1	COL0	COL3	COL2	COL1	COL0

In Interlace Mode 2, one cell consists of 8 x 16 dots and therefore 64 bytes (16 long words) are required.

1	2	3	4	5	6	7	8
a	□	□	□	□	□	□	□
b	□	□	□	□	□	□	□
c	□	□	□	□	□	□	□
d	□	□	□	□	□	□	□
e	□	□	□	□	□	□	□
f	□	□	□	□	□	□	□
g	□	□	□	□	□	□	□
h	□	□	□	□	□	□	□
i	□	□	□	□	□	□	□
j	□	□	□	□	□	□	□
k	□	□	□	□	□	□	□
l	□	□	□	□	□	□	□
m	□	□	□	□	□	□	□
n	□	□	□	□	□	□	□
o	□	□	□	□	□	□	□
p	□	□	□	□	□	□	□

	0	1	2	3			
	D7	D0 D7	D0 D7	D0 D7	D0		
00	a1	a2	a3	a4	a5	a6	a7
04	b1	b2	b3	b4	b5	b6	b7
08	c1	c2	c3	c4	c5	c6	c7
0C	d1	d2	d3	d4	d5	d6	d7
10	e1	e2	e3	e4	e5	e6	e7
14	f1	f2	f3	f4	f5	f6	f7
18	g1	g2	g3	g4	g5	g6	g7
1C	h1	h2	h3	h4	h5	h6	h7
20	i1	i2	i3	i4	i5	i6	i7
24	j1	j2	j3	j4	j5	j6	j7
28	k1	k2	k3	k4	k5	k6	k7
2C	l1	l2	l3	l4	l5	l6	l7
30	m1	m2	m3	m4	m5	m6	m7
34	n1	n2	n3	n4	n5	n6	n7
38	o1	o2	o3	o4	o5	o6	o7
3C	p1	p2	p3	p4	p5	p6	p7

J. WINDOW

For Window display, the following register setting and VRAM areas are required.

Window Pattern Name Table and Base Address

Register #3

0	0	WD15	WD14	WD13	WD12	WD11	0
---	---	------	------	------	------	------	---

Mode Set Register Number 4

Register #12

RS0	0	0	0	S/TE	LSM1	LSM0	RS1
-----	---	---	---	------	------	------	-----

Window H Position

Register #17

RIGT	0	0	WHP5	WHP4	WHP3	WHP2	WHP1
------	---	---	------	------	------	------	------

Window V Position

Register #18

DOWN	0	0	WVP4	WVP3	WVP2	WVP1	WVPO
------	---	---	------	------	------	------	------

VRAM: Window Pattern Name Table Maximum 4 Kbytes.

1. DISPLAY POSITION

The Window Display Position is designated by Registers #17 and #18. Screen display can be divided on a unit basis of H2 cells and V1 cell. The dividing position varies depending on resolution.

	0	1	2	3	4	5		34	35	36	37	38	39
0													
1													
2													
25													
26													
27													

H 40 cells / V 28 cells mode

Register #17

RIGHT	0	0	WHP5	WHP4	WHP3	WHP2	WHP1
-------	---	---	------	------	------	------	------

Register #18

DOWN	0	0	WVP4	WVP3	WVP2	WVP1	WVP0
------	---	---	------	------	------	------	------

- RIGHT: 0 Displays Window from the left to H dividing position
 1 Displays Window from the H dividing position to the right end.
 DOWN: 0 Displays Window from the top end to the V dividing position.
 1 Displays Window from the V dividing position to the bottom end.

WHP5 ~ WHP1: H dividing position
 WVP4 ~ WVP0: V dividing position

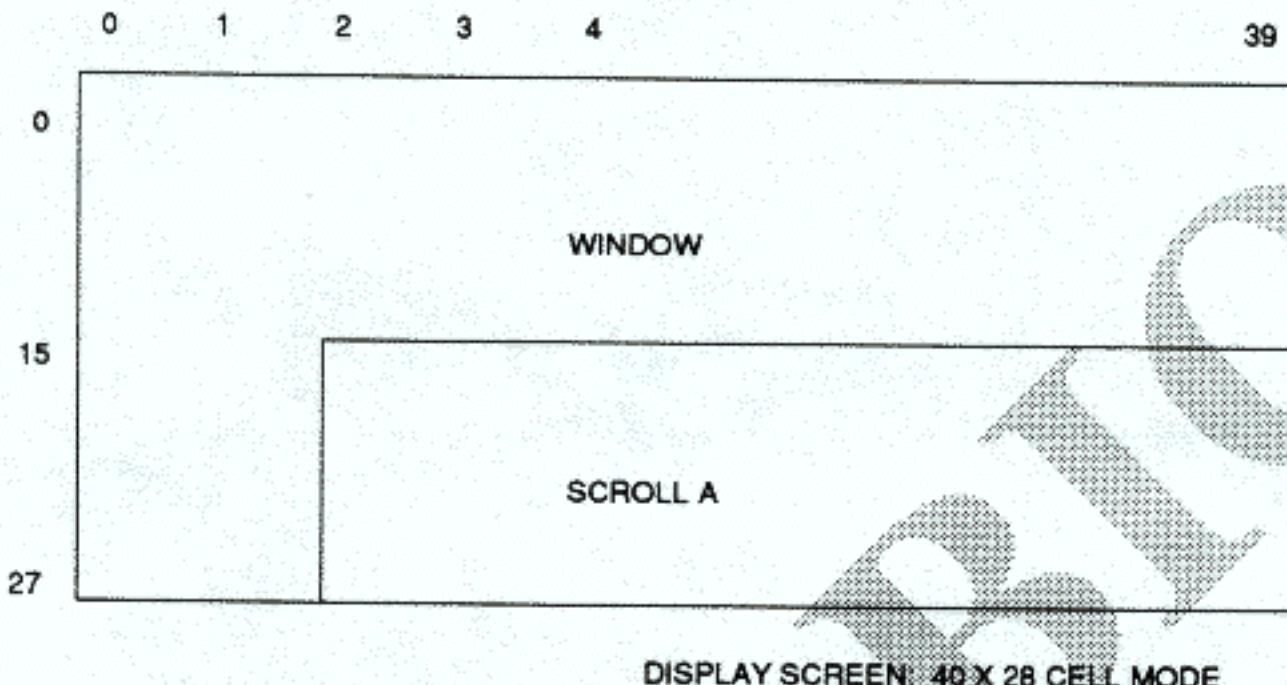
H Resolution	Dividing Position (WHP)
32 cell	0 ~ 16 (0 ~ 32 cell)
40 cell	0 ~ 20 (0 ~ 40 cell)

V Resolution	Dividing Position (WVP)
28 cell	0 ~ 28
30 cell	0 ~ 30

Setting Example

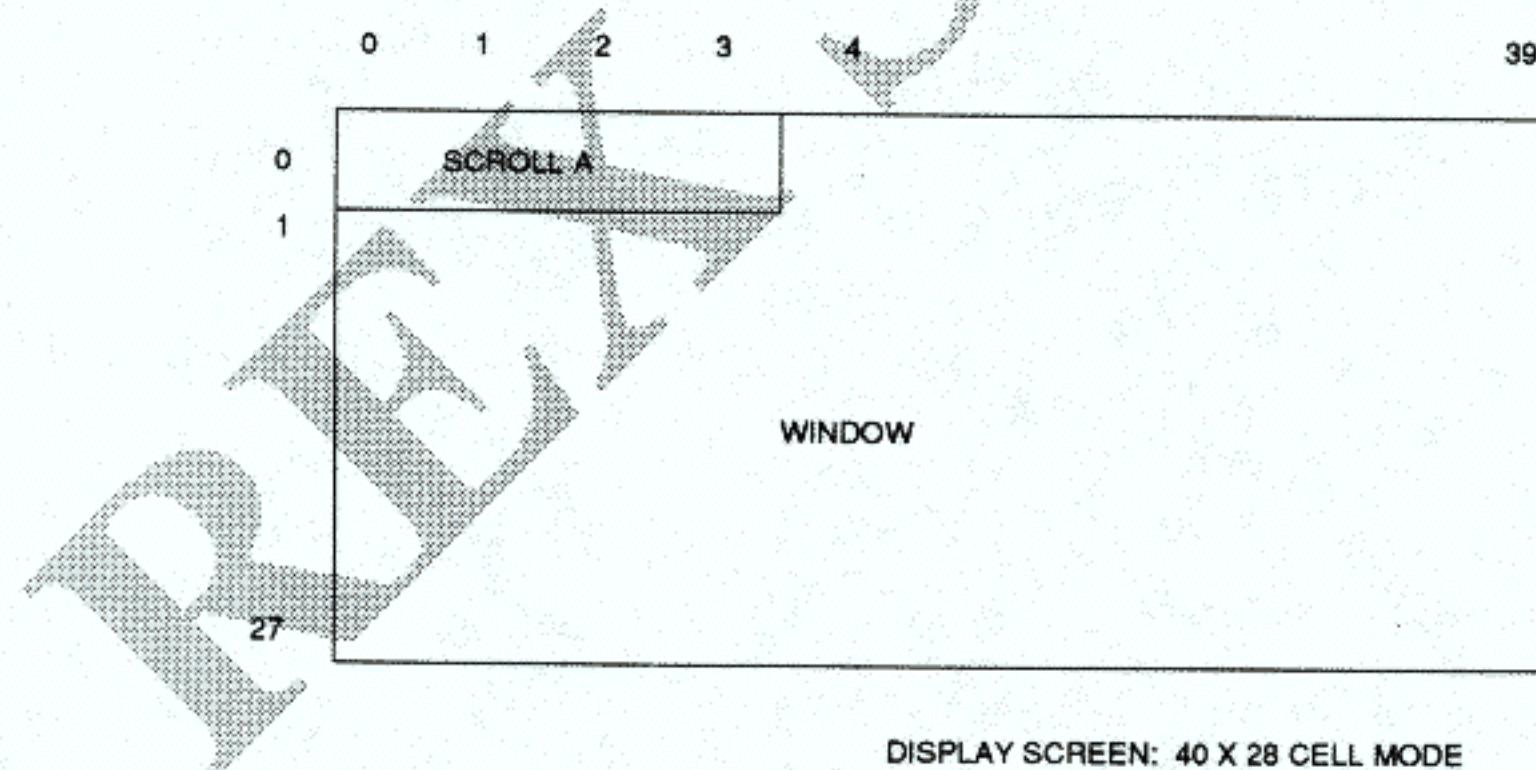
Register #17: 000H+01H
Register #18: 000H+10H

Window from the left end to the second cell
Window from the top end to the 16th cell



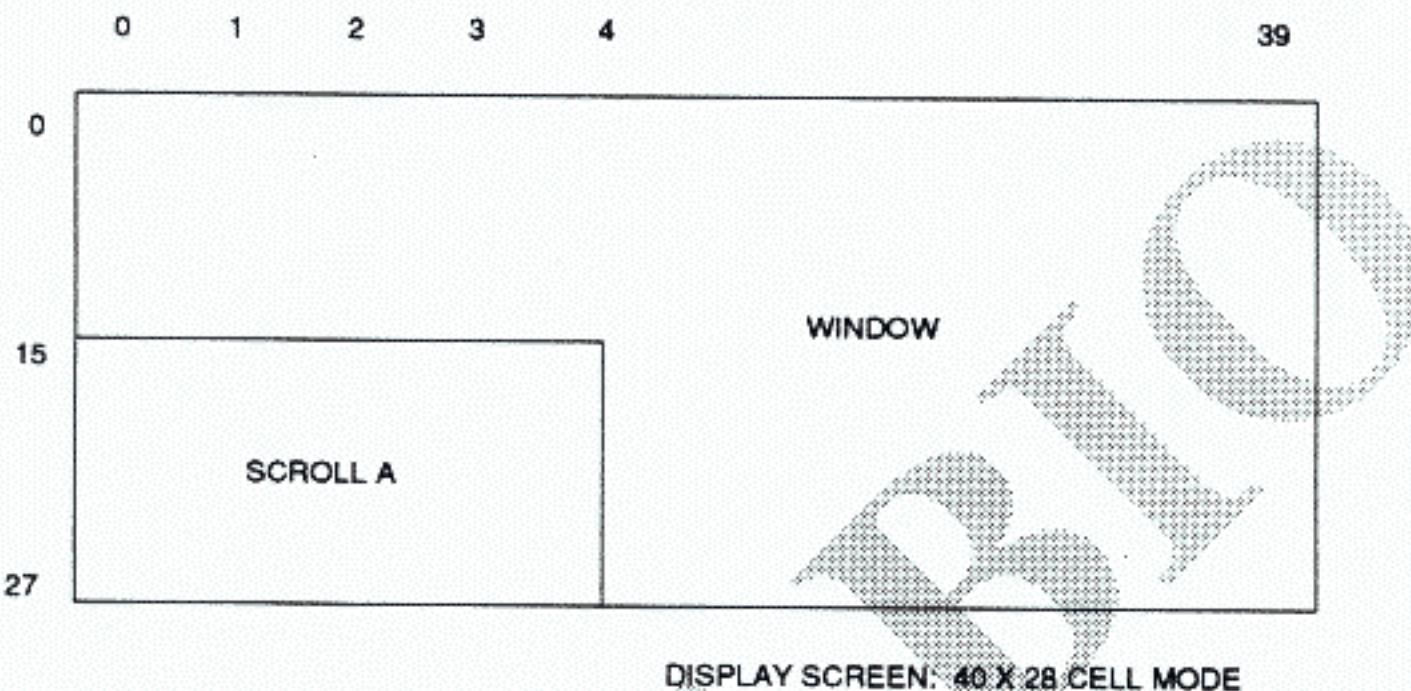
Register #17: 80H+02H
Register #18: 80H+01H

Window from the left end 4th cell to the right end
Window from the 2nd cell to the bottom end



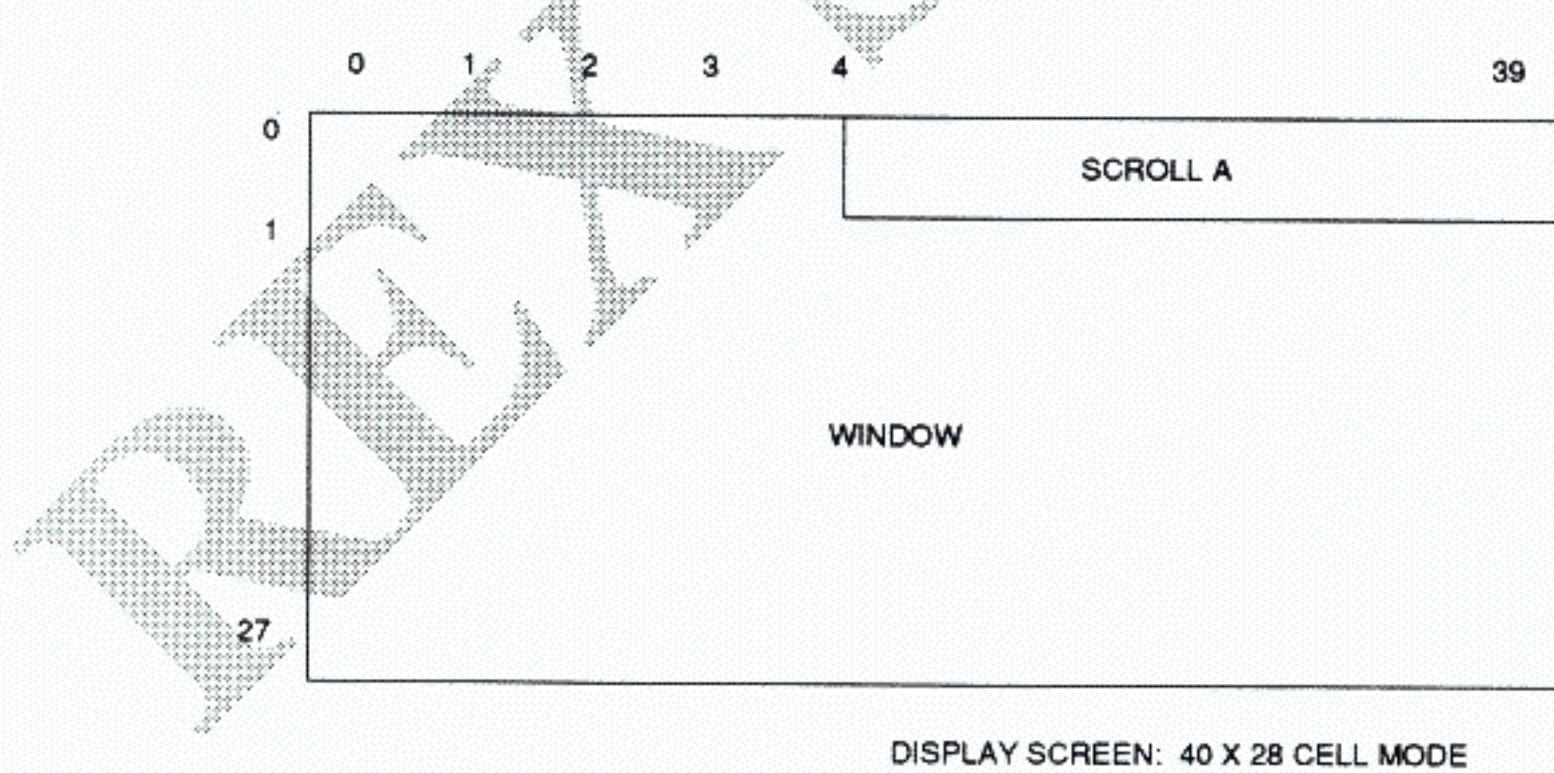
Register #17: 80H+01H
Register #18: 00H+10H

Window from the 4th cell to the right end
Window from the top end to the 16th cell



Register #17: 00H+02H
Register #18: 80H+01H

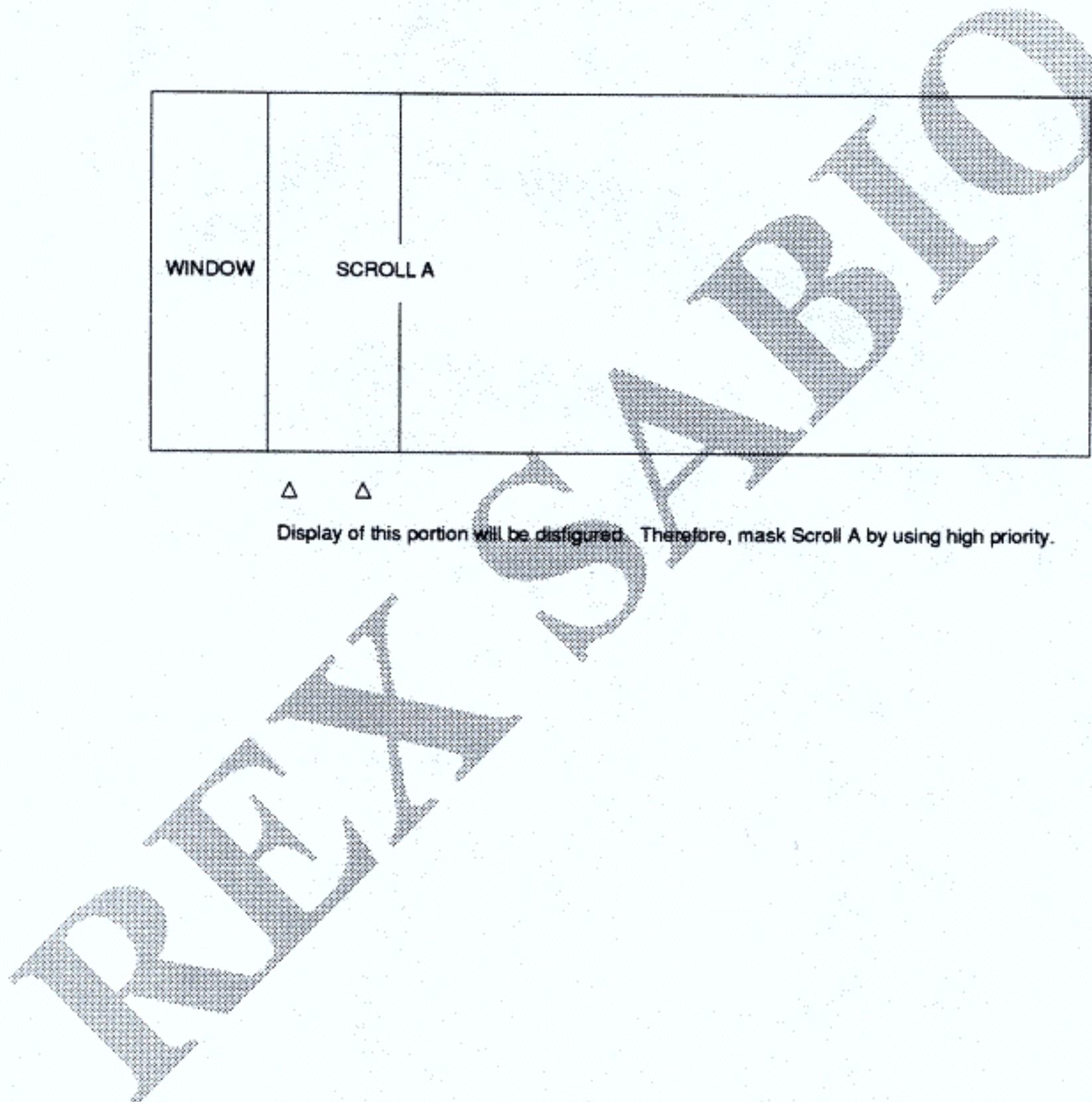
Window to the 4th cell from the left
Window from the 2nd cell to the bottom end



2. WINDOW PRIORITY

Window Priority is handled in the same way as in Scroll A. Scroll A is not displayed in the area where Window is displayed. Also, only when Window is set to the left and Scroll A is moved in H direction, the character corresponding to two cells on the right side of the boundary between Window and Scroll A will be disfigured.

There will be no malfunctioning when Window is set to the left side and Scroll A is moved only in the V direction; also when Window is set to the right side.



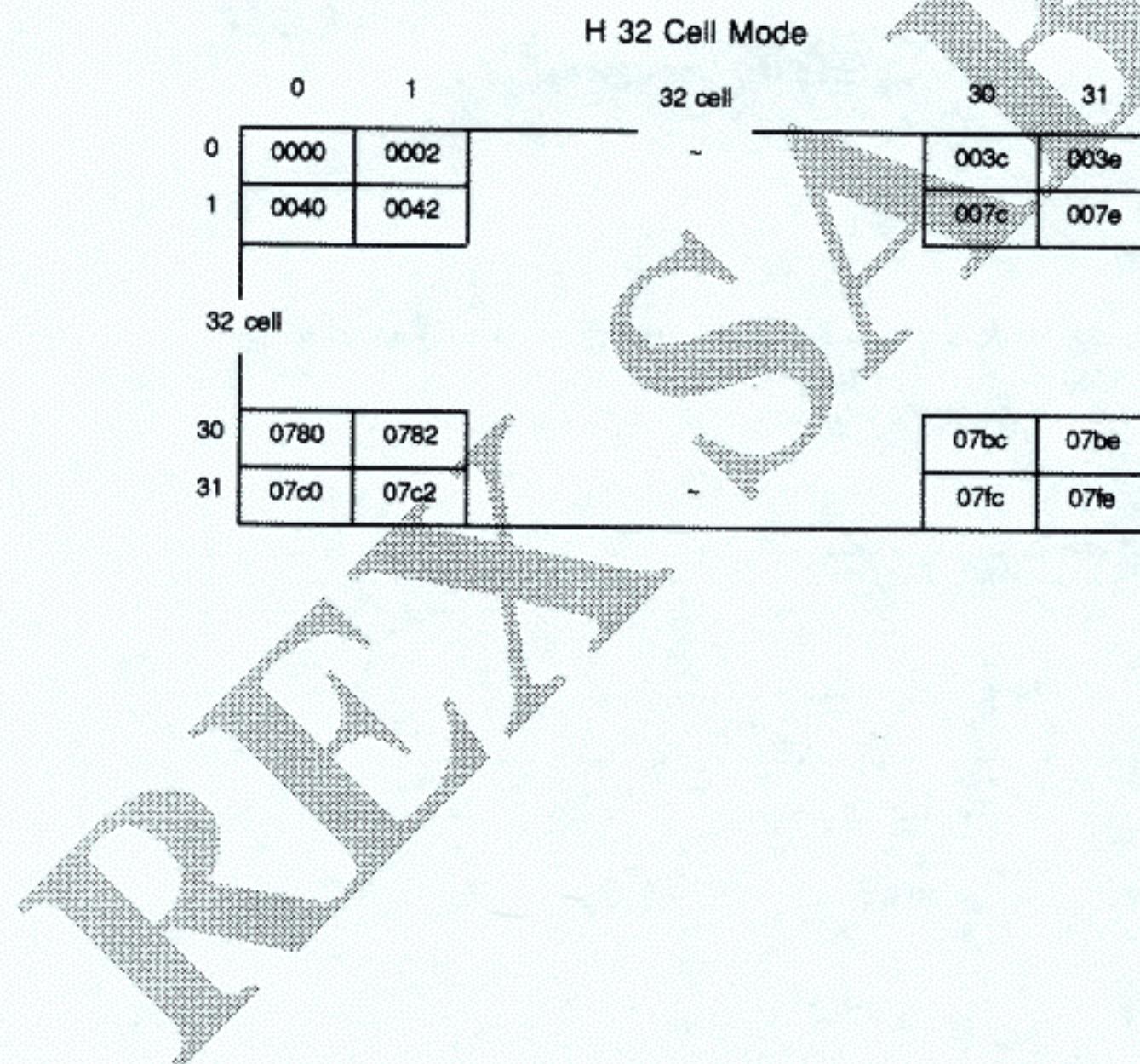
3. WINDOW PATTERN NAME

Window Pattern Name Table is on VRAM, and the base address is designated by Register #13. The Pattern Name, the same as in Scroll Screen, requires 2 bytes (1 word) per cell.

Pattern Name	pri	cp1	cp0	vf	hf	pt10	pt9	pt8	(D15 ~ D8)
	pt7	pt6	pt5	pt4	pt3	pt2	pt1	pt0	(D7 ~ D0)

pri: Refer to Priority
 cp1: Color palette selection bit
 cp0: Color palette selection bit
 vf: V reverse Bit 1: Reverse
 hf: H Reverse Bit 1: Reverse
 pt10~pt0: Pattern Generator Number

Pattern Name and VRAM relation varies depending on H 32 cell/40 cell mode. Pay careful attention to this point.



H 40 Cell Mode							
0	1	39		40	62		63
0	0000	0002	~	004e	0050	~	007c
1	0080	0082		00dc	00e0		00fc
32 cell							
30	0f00	0f02		0f4e	0f50		0f7c
31	0fc0	0fc2	~	0fde	0fe0	~	0ffc
							0fe

40~63 are not displayed

Values shown are offset from the Base Address

In the H40 Cell Mode, there exists the area for H64 cells. However, there will be no display from the 41st cell in the H direction.

In the V28 Cell Mode, there will be no display from the V29th cell, and in the 30th Cell Mode, there will be no display from the 31st cell.

K. SPRITE

For Sprite Display, the following register setting and VRAM area are required.

Sprite Attribute Table and Base Address

Register #5	0	AT15	AT14	AT13	AT12	AT11	AT10	AT9
-------------	---	------	------	------	------	------	------	-----

Mode Setting Register #4

Register #12	RS0	0	0	0	S/TE	LSM1	LSM0	RS1
--------------	-----	---	---	---	------	------	------	-----

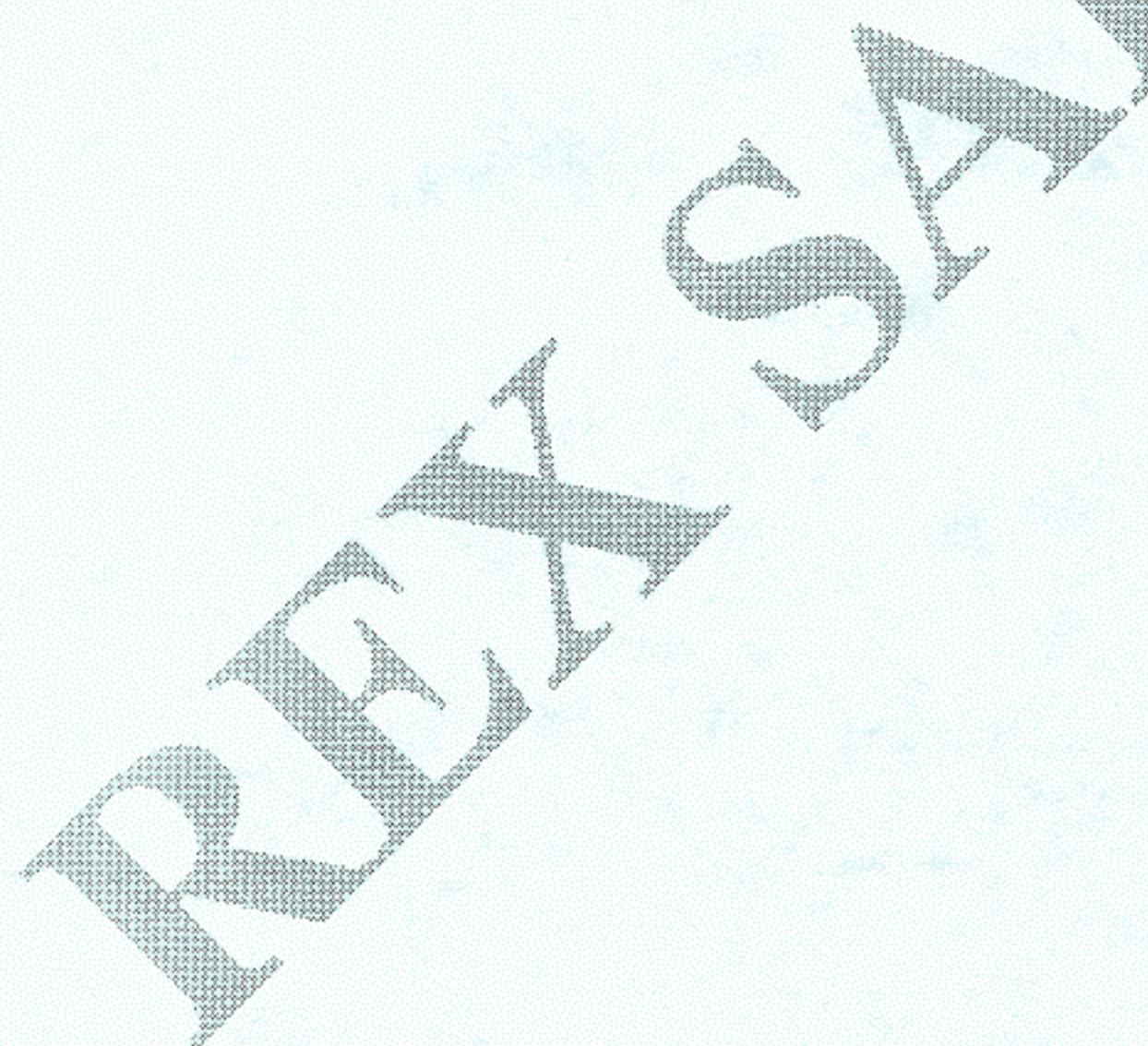
VRAM: Sprite Attribute Table Maximum 640 Bytes

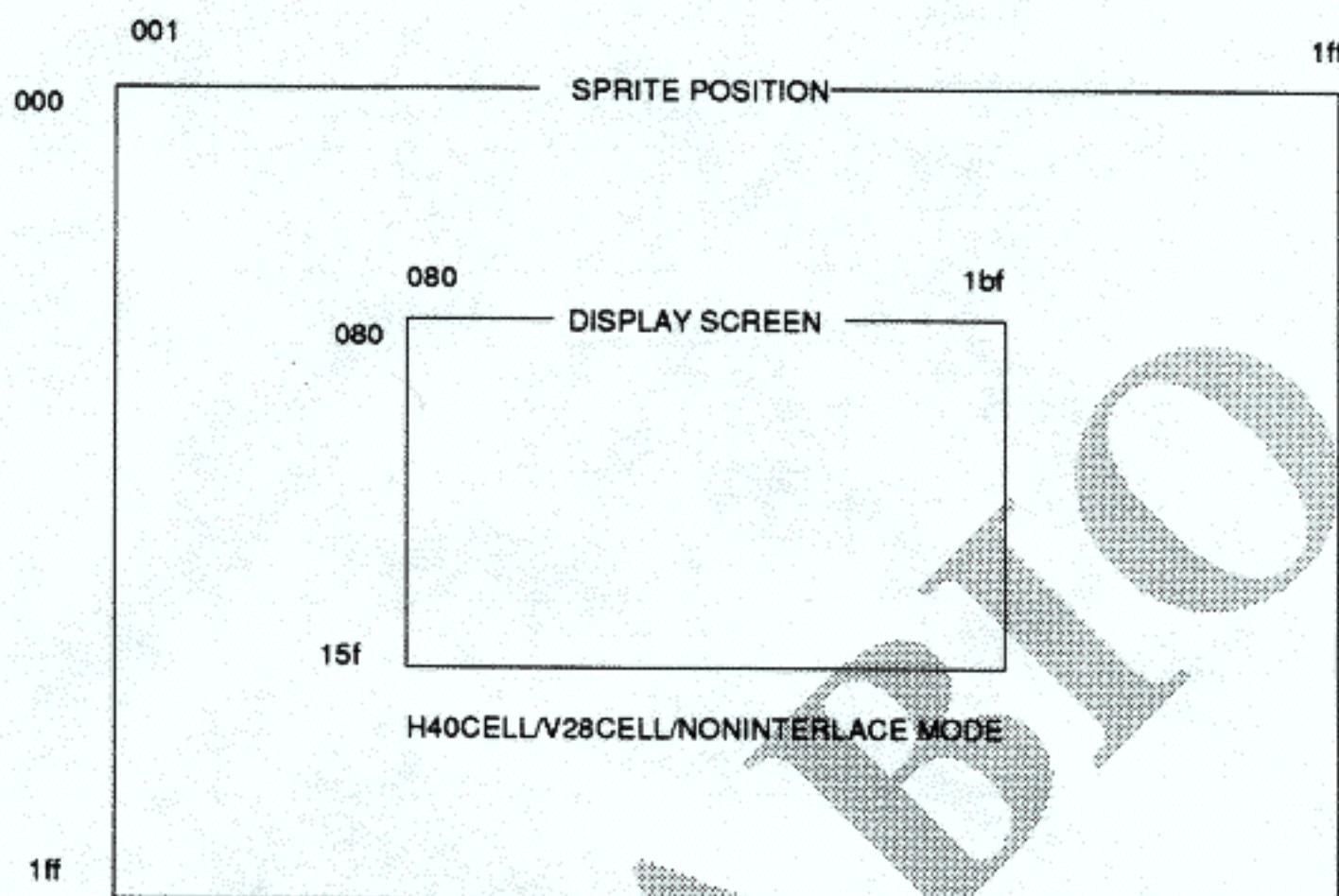
1. DISPLAY POSITION

Sprite Position and Display Screen are as follows: When Sprite H position is 0, this is in a special mode. Therefore, pay careful attention to this point.

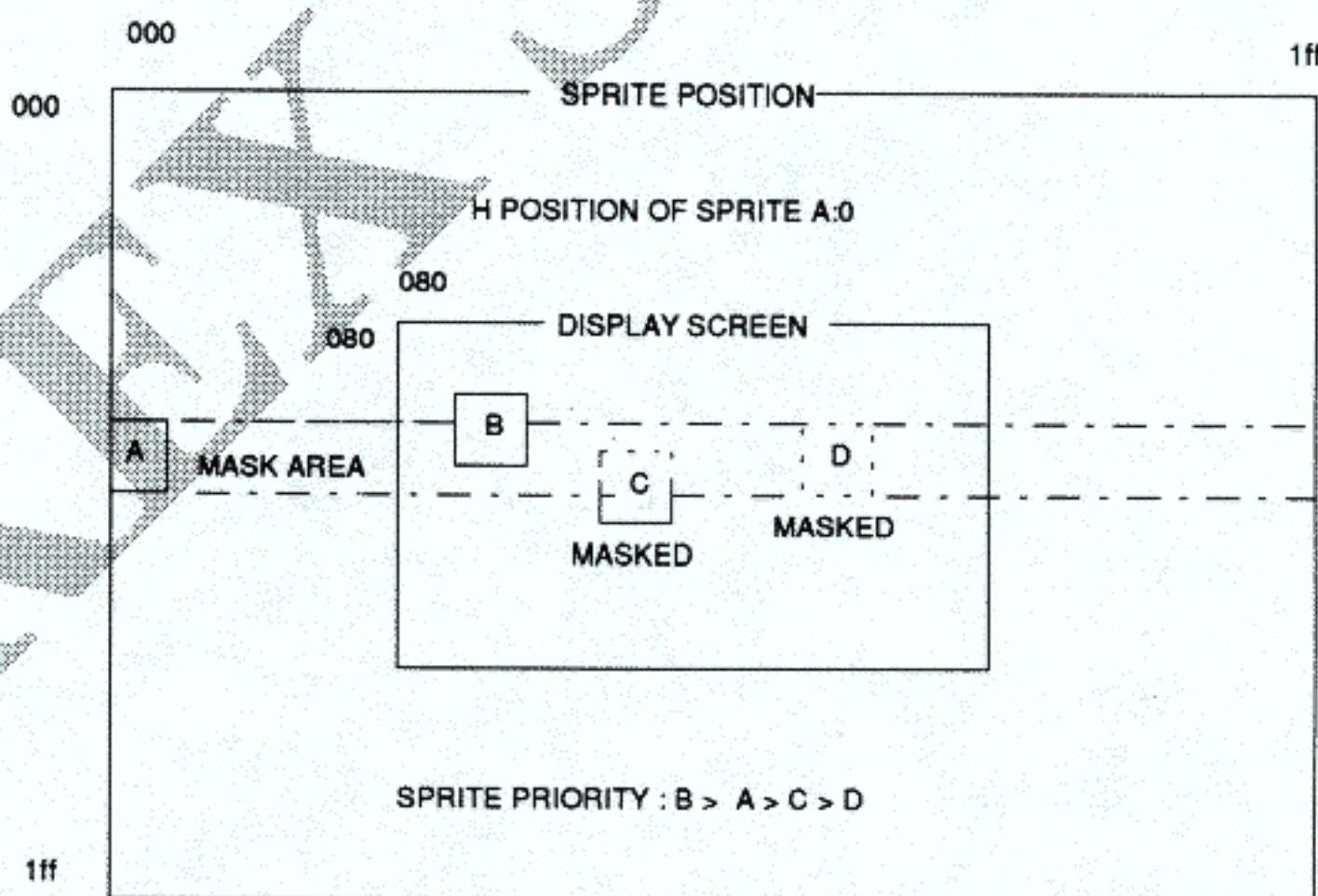
Resolution	H Position	Display Area
H32 Cell	001~1FFH	080 ~ 17FH
H40 Cell		080 ~ 1BFH

Resolution		V Position	Display Area
V28 Cell	Non-interface	000 ~ 1FFH	080 ~ 15FH
	Interface 1		
	Interface 2	000 ~ 3FFH	100 ~ 2BFH
V30 Cell	Non-interface	000 ~ 1FFH	080 ~ 16FH
	Interface 1		
	Interface 2	000 ~ 3FFH	100 ~ 2DFH





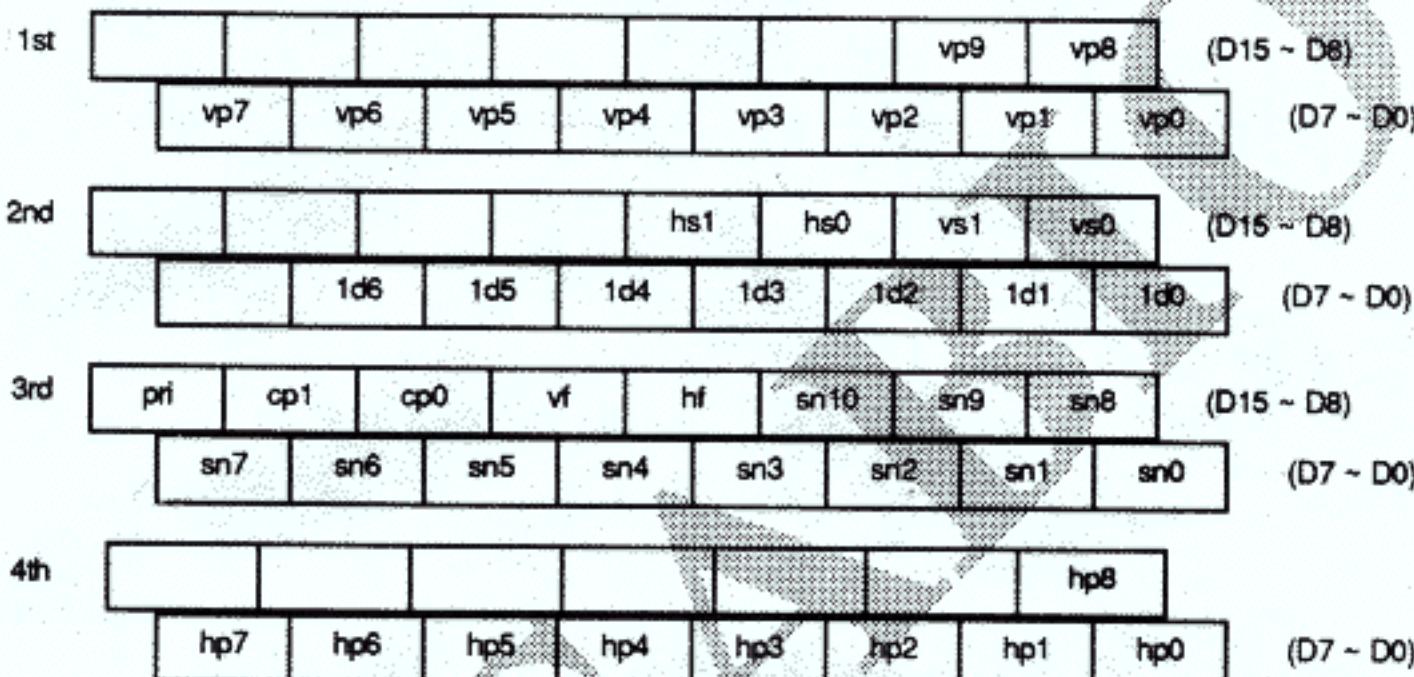
When 0 is set to the SPRITE H POSITION, a low priority sprite on the same line will not be displayed.



2. SPRITE ATTRIBUTE

Sprite Attribute Table is on VRAM and the Base Address is designated by Register #5. Attribute requires 8 bytes (4 words) per Sprite, and indicates Display Position, Priority, Sprite Generator Number, and Attribute.

Starting from the beginning of the Attribute Table, numbers are given in the sequence of Sprite 0, Sprite 1, Sprite 2, Sprite 3, etc. Priority between Sprites is not determined by the sequence of Sprite number, but by each Sprite's Link Data, and thus becomes programmable.



Blank portions can be utilized freely for software.

vp9 ~ vp0: V position

hp8 ~ hp0: H position

hs1, hs0: Sprite's H Size

vs1, vs0: Sprite's V Size

1d6 ~ 1d0: Link Data

pri: Priority Bit (see *Priority*)

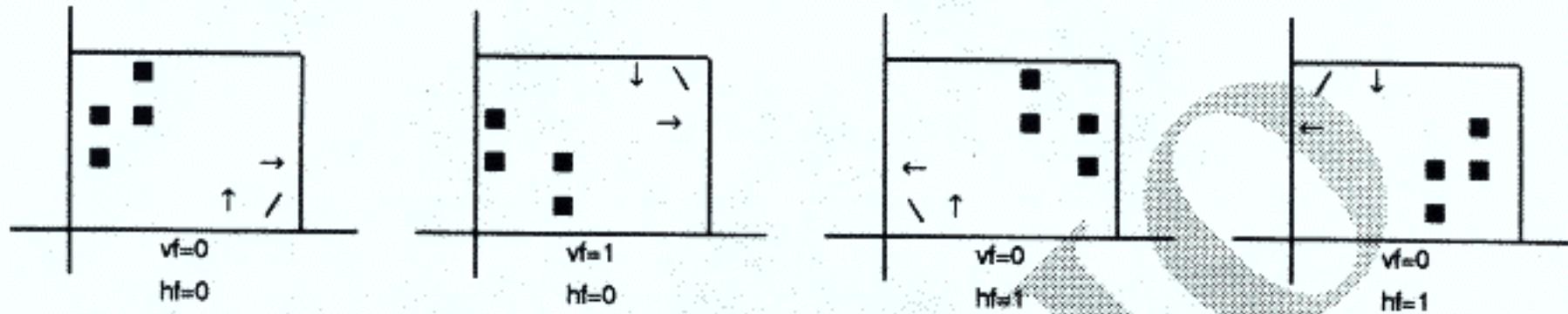
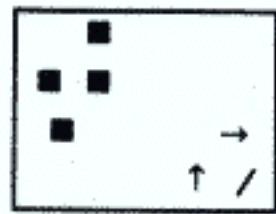
cp1, cp0: Color palette selection bit (see *Color Palette*)

vf: V Reverse Bit 1: Reverse

hf: H Reverse Bit 1: Reverse

sn10 ~ sn0: Sprite Pattern Generator Number

By using Reverse Bit vf, hf, V and H Reverse per Sprite is possible.



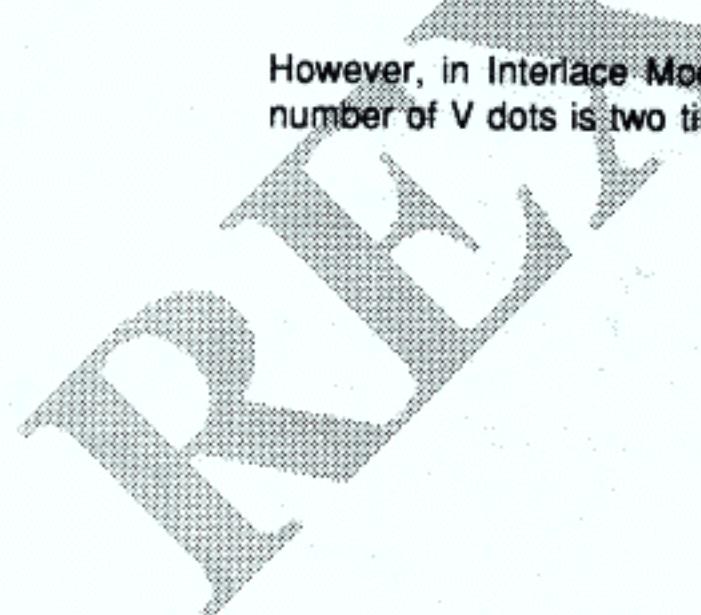
3. SPRITE SIZE

Per Sprite dot number can be set on a cell unit basis, by using vs1, vs0, hs1, and hs0.

V		
vs1	vs0	Number of cell
0	0	1 (8 dots)
0	1	2 (16 dots)
1	0	3 (24 dots)
1	1	4 (32 dots)

H		
hs1	hs0	Number of cell
0	0	1 (8 dots)
0	1	2 (16 dots)
1	0	3 (24 dots)
1	1	4 (32 dots)

However, in Interlace Mode 2, one cell is comprised of 8 x 16 dots, therefore, the number of V dots is two times (as compared to Interlace Mode 1).



4. SPRITE's DISPLAY CAPACITY

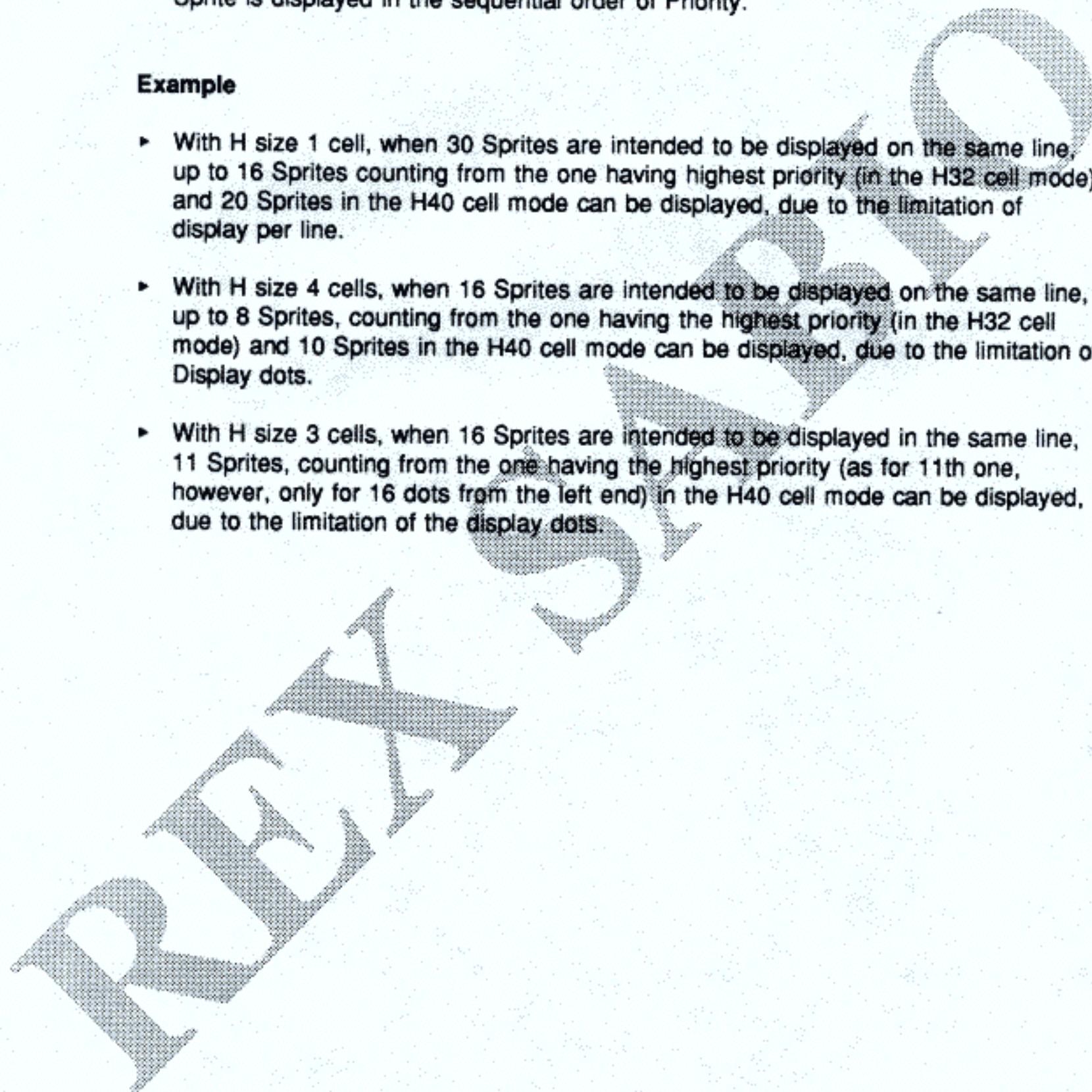
The number of Sprite's maximum display varies depending on H resolution setting.

Resolution	No. of Display	No. of Display per Line	Display Dot per Line
H32 cell	Max. 64 Sprites	Max. 16 Sprites	Max. 256 dots
H40 cell	Max. 80 Sprites	Max. 20 Sprites	Max. 320 dots

Sprite is displayed in the sequential order of Priority.

Example

- With H size 1 cell, when 30 Sprites are intended to be displayed on the same line, up to 16 Sprites counting from the one having highest priority (in the H32 cell mode) and 20 Sprites in the H40 cell mode can be displayed, due to the limitation of display per line.
- With H size 4 cells, when 16 Sprites are intended to be displayed on the same line, up to 8 Sprites, counting from the one having the highest priority (in the H32 cell mode) and 10 Sprites in the H40 cell mode can be displayed, due to the limitation of Display dots.
- With H size 3 cells, when 16 Sprites are intended to be displayed in the same line, 11 Sprites, counting from the one having the highest priority (as for 11th one, however, only for 16 dots from the left end) in the H40 cell mode can be displayed, due to the limitation of the display dots.



5. PRIORITY BETWEEN SPRITES

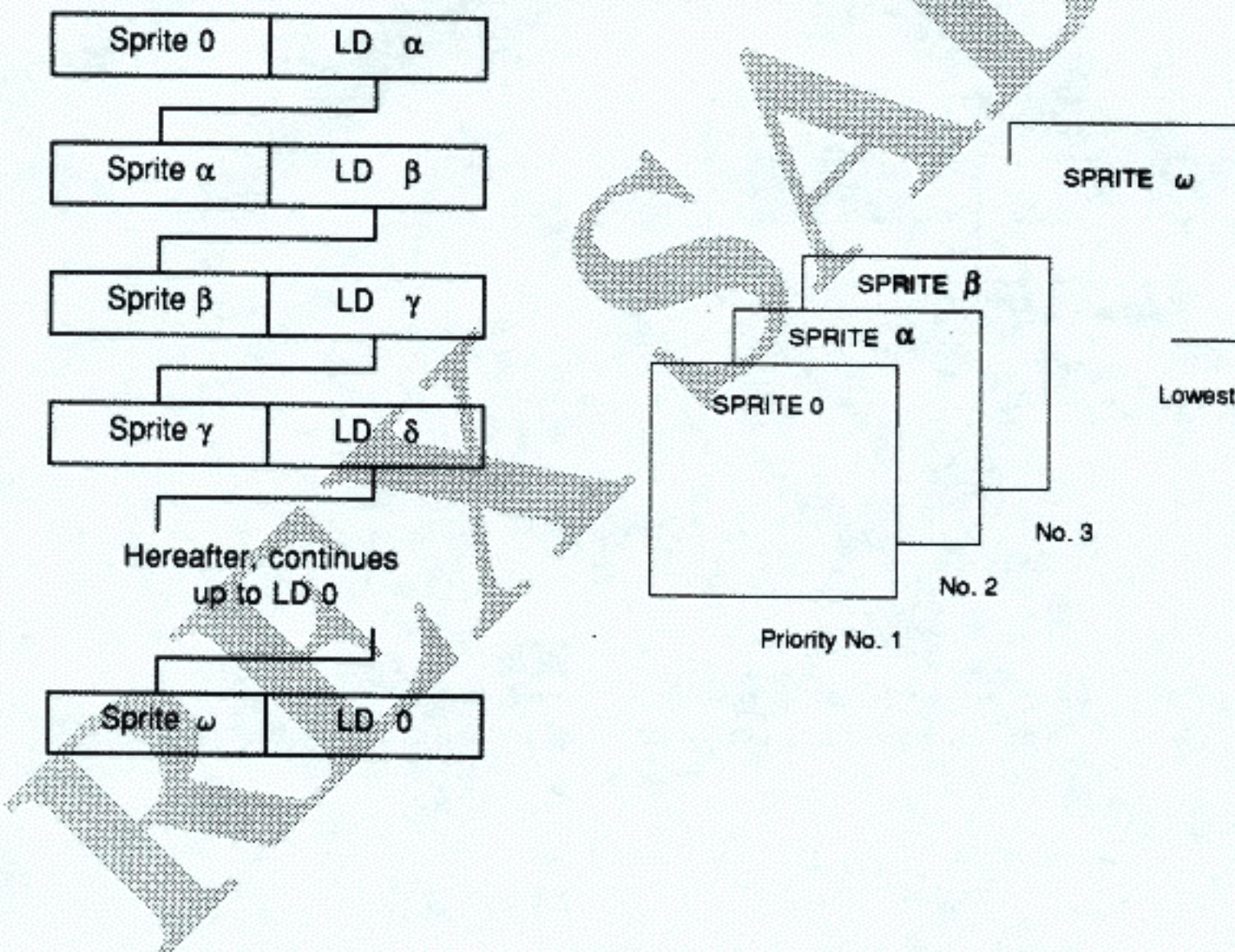
Priority between Sprites is designated by each Sprite's Link Data.

With Sprite 0 being Priority No. 1, the Sprite No. 2 written in the Link Data there will be Priority No. 2. Priority No. 2 Sprite's Link Data shows Priority No. 3 Sprite. Priority No. 3 Sprite's Link Data shows Priority No. 4 Sprite.

In this way, Priority is sequentially designated by each Sprite's Link Data and thus it is in List form. The value that can be set in the Link Data is 0~ (number of maximum Display on one screen minus one). Be sure to set 0 to the lowest priority Sprite's Link Data.

When 0 is given to the Sprite Link Data, List ends at that Sprite and the Priority will become the lowest. Even in the case that the number of Sprites linked to List is less than the maximum display quantity (64 or 80), the remaining Sprites not linked to Sprite will not be displayed.

When value other than those specified is set to Link Data, or 0 is not set to the lowest Priority Sprite Link Data, ordinary functioning is not guaranteed.



Setting Example

	Link Data
Sprite 0	2
Sprite 1	10
Sprite 2	1
Sprite 3	4
Sprite 4	5
Sprite 5	15
Sprite 6	—
Sprite 7	0
Sprite 8	—
Sprite 9	—
Sprite 10	11
Sprite 11	13
Sprite 12	—
Sprite 13	3
Sprite 14	—
Sprite 15	7
Sprite 16	—

Sprite 0
Sprite 2
Sprite 1
Sprite 10
Sprite 11
Sprite 13
Sprite 3
Sprite 4
Sprite 5
Sprite 15
Sprite 7

The 11 Sprites shown in the Display Priority are displayed on the screen. Sprites No. 6, 8, 9, 12, 14, and 16 onward are not displayed because they are not linked with Link Data List.

6. SPRITE PATTERN GENERATOR

The Sprite Pattern Generator with VRAM 0000H as Base Address expresses one pattern on a basis of 8x8 dots. 32 bytes are required to define one pattern. Every 32 bytes, one pattern is expressed in the sequence of Pattern Generator 0, 1, 2, etc. The relationship of Display Pattern and Memory is the same as in Pattern Generator. Also, Sprite Size and Pattern Generator relationship is as follows:

V1 cell H1 cell	V1 cell H2 cell	V1 cell H3 cell	V1 cell H4 cell
0	0 1	0 1 2	0 1 2 3

V2 cell H1 cell	V2 cell H2 cell	V2 cell H3 cell	V2 cell H4 cell
0	0 2	0 2 4	0 2 4 6
1	1 3	1 3 5	1 3 5 7

V3 cell H1 cell	V3 cell H2 cell	V3 cell H3 cell	V3 cell H4 cell
0	0 3	0 3 6	0 3 6 9
1	1 4	1 4 7	1 4 7 A
2	2 5	2 5 8	2 5 8 B

V4 cell H1 cell	V4 cell H2 cell	V4 cell H3 cell	V4 cell H4 cell
0	0 4	0 4 8	0 4 8 C
1	1 5	1 5 9	1 5 9 D
2	2 6	2 6 A	2 6 A E
3	3 7	3 7 B	3 7 B F

$$\frac{4 \text{ bits}}{\text{pix}} \times \frac{8 \text{ pix}}{\text{char}} \times \frac{1 \text{ byte}}{8 \text{ bits}} = \frac{4 \text{ bytes}}{8 \text{ bits}} \text{ across char}$$

$$4 \text{ bytes} \times 8 \text{ rows} = 32 \text{ bytes/char}$$

L. PRIORITY

Priority between Sprite, Scroll A, and Scroll B can be designated.

Priority can be designated by each Pattern Name and Attribute Priority bit. It will be set for the Scroll Screen on a cell unit basis and for each Sprite. By combining each priority bit, Priority will be as follows, however, the Background Priority is always the lowest.

S pri	A pri	B pri	Priority
0	0	0	S>A>B>G
1	0	0	S>A>B>G
0	1	0	A>S>B>G
1	1	0	S>A>B>G
0	0	1	B>S>A>G
1	0	1	S>B>A>G
0	1	1	A>B>S>G
1	1	1	S>A>B>G

S: Sprite

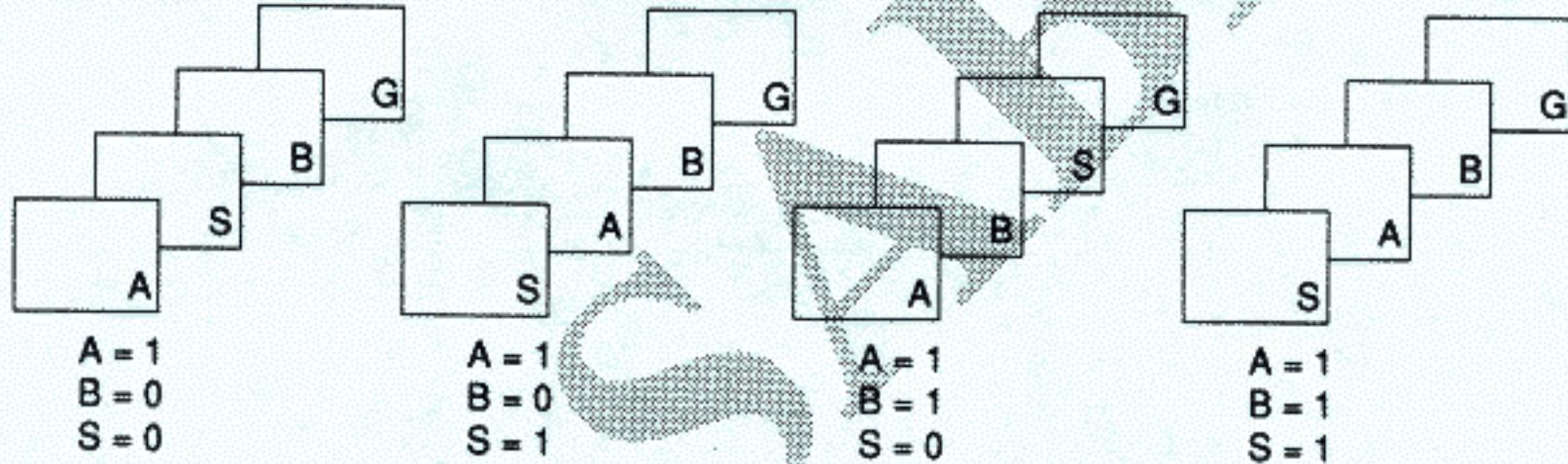
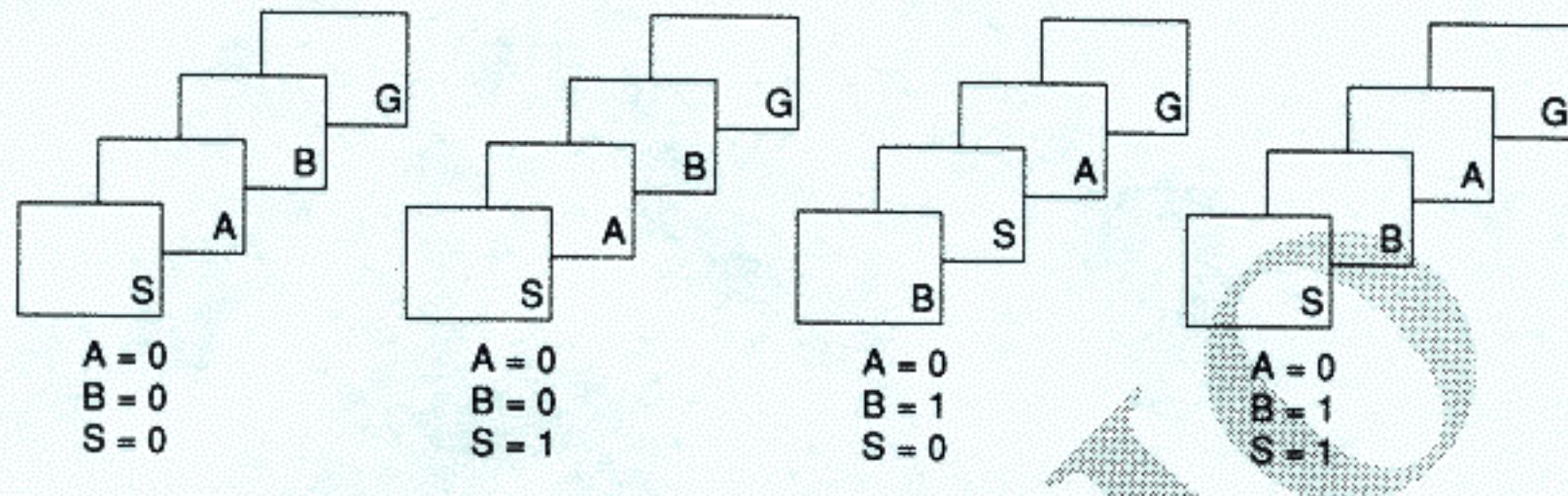
A: Scroll A

B: Scroll B

G: Background

Also, by combining S/Ten (Register #12) and the above priority, Shadow-Highlight effect function can be utilized.

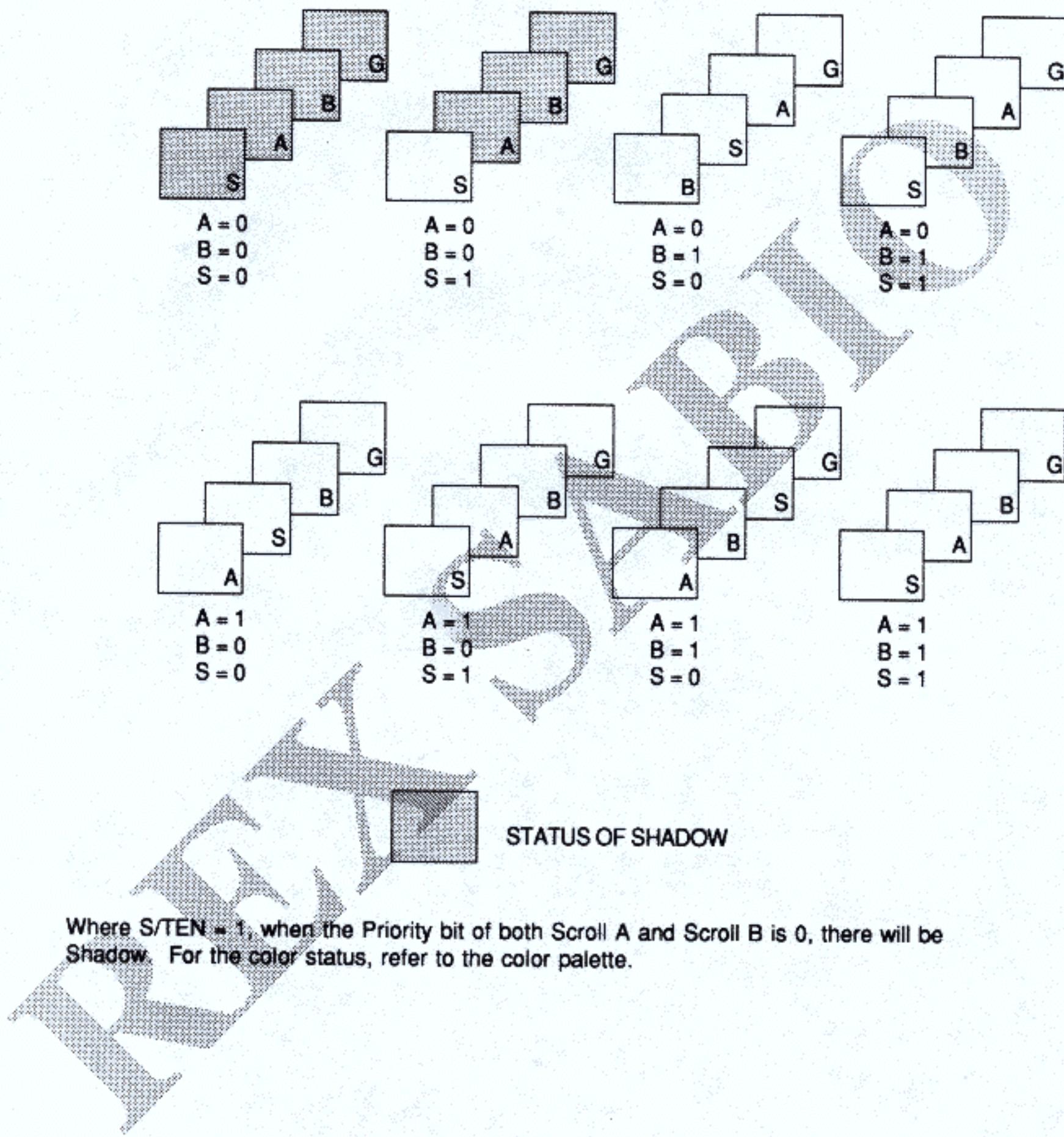




The above shows Priority situation of Sprite, Scroll A, Scroll B and Background. The dot to which Color Code 0 is designated is transparent. Therefore, either one of Scroll Screen A, Scroll Screen B, or Background (the priority of which is one step lower than the transparent one) will appear.

S/TEN - 1

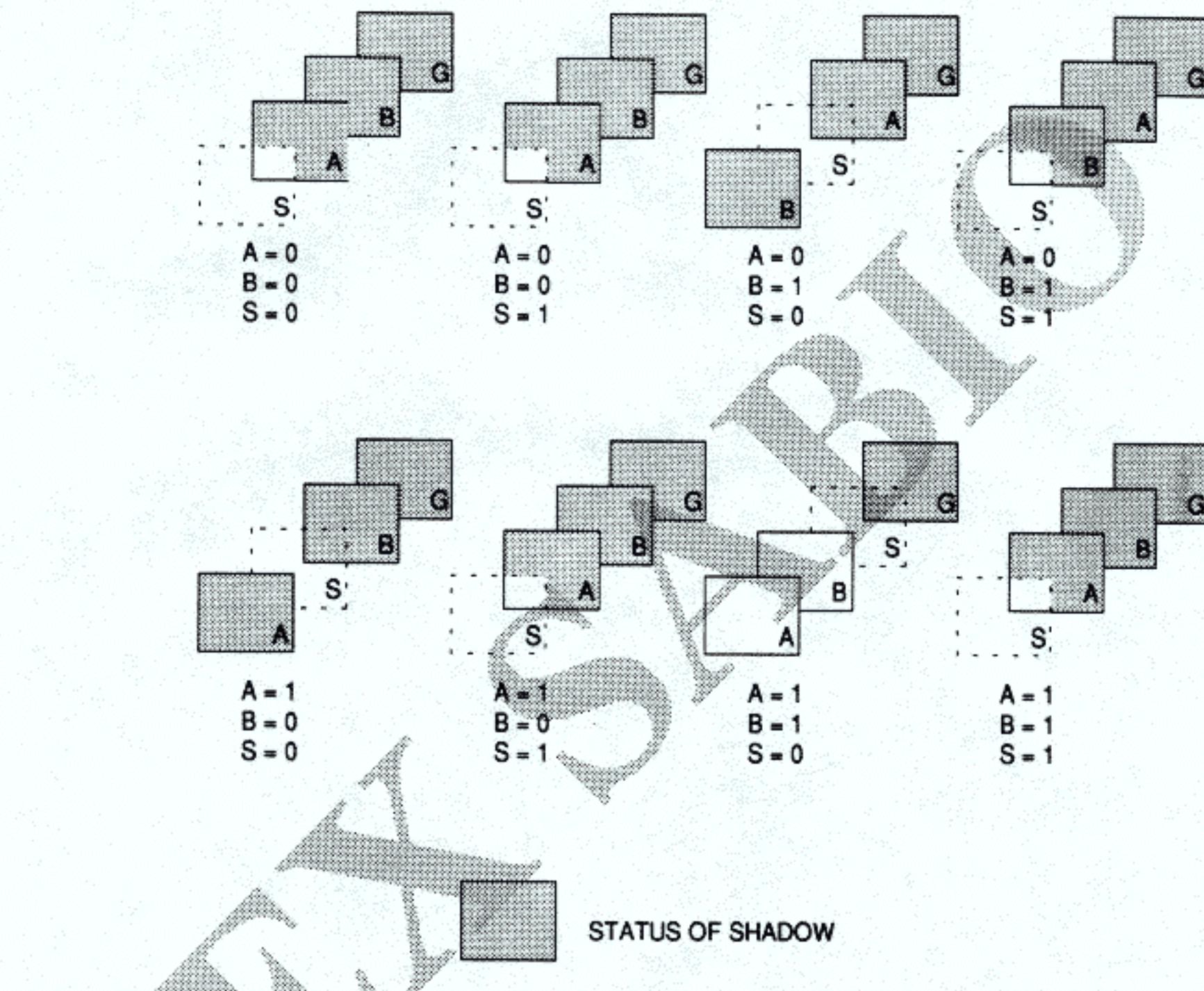
Sprite Color Palette 0 ~ 3 , Color Code 0 ~ 15
Color Palette 3 , Color Code 0 ~ 13



Where S/TEN = 1, when the Priority bit of both Scroll A and Scroll B is 0, there will be Shadow. For the color status, refer to the color palette.

S/TEN = 1

Sprite Color Palette 3 , Color Code 15

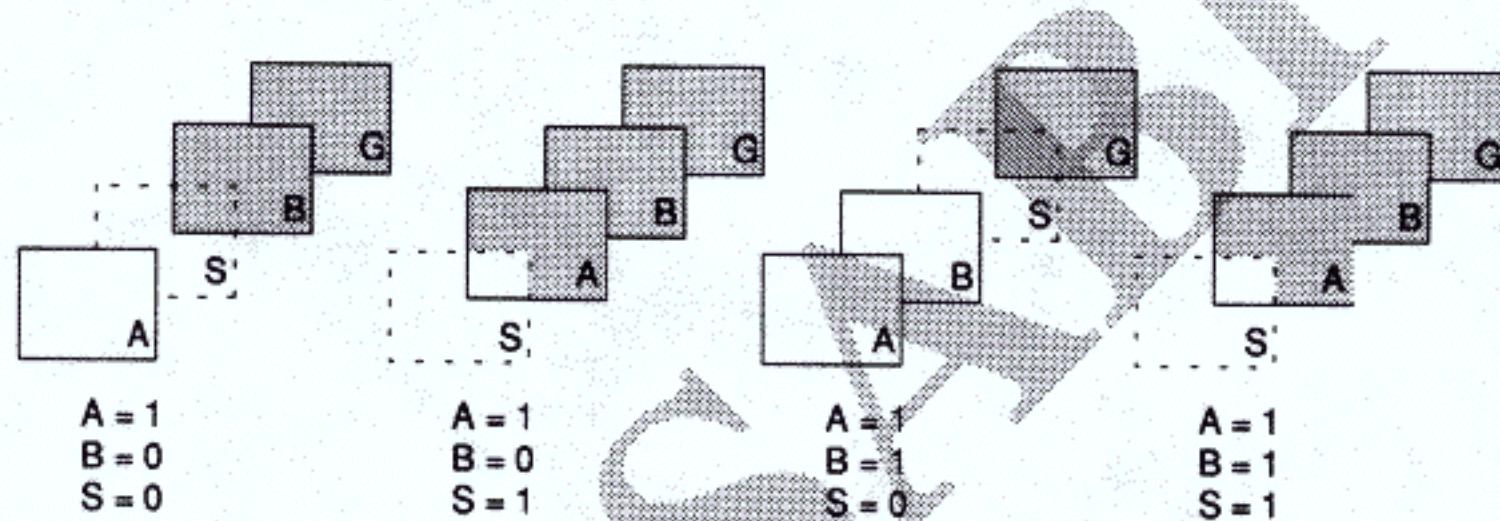
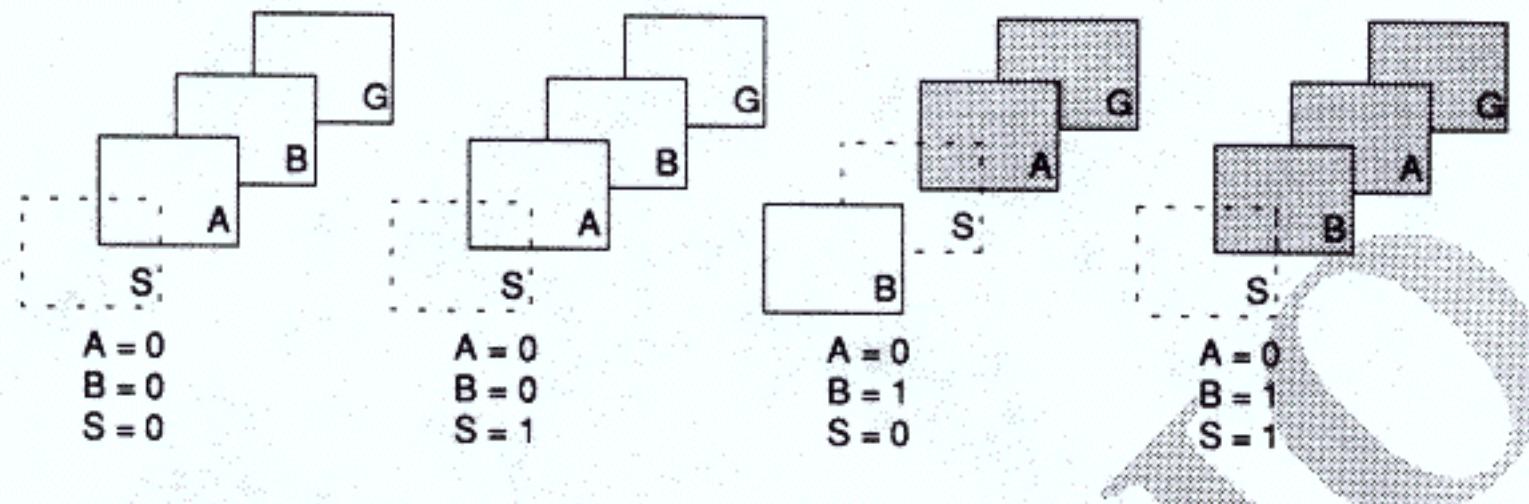


The dots for the Sprite Color Code 15 work as a Shadow operator on the screen, the Priority of which is lower than the Sprite.

Since Sprite dot works as an operator, this will not be displayed.

S/TEN = 1

Sprite Color Palette 3 , Color Code 14



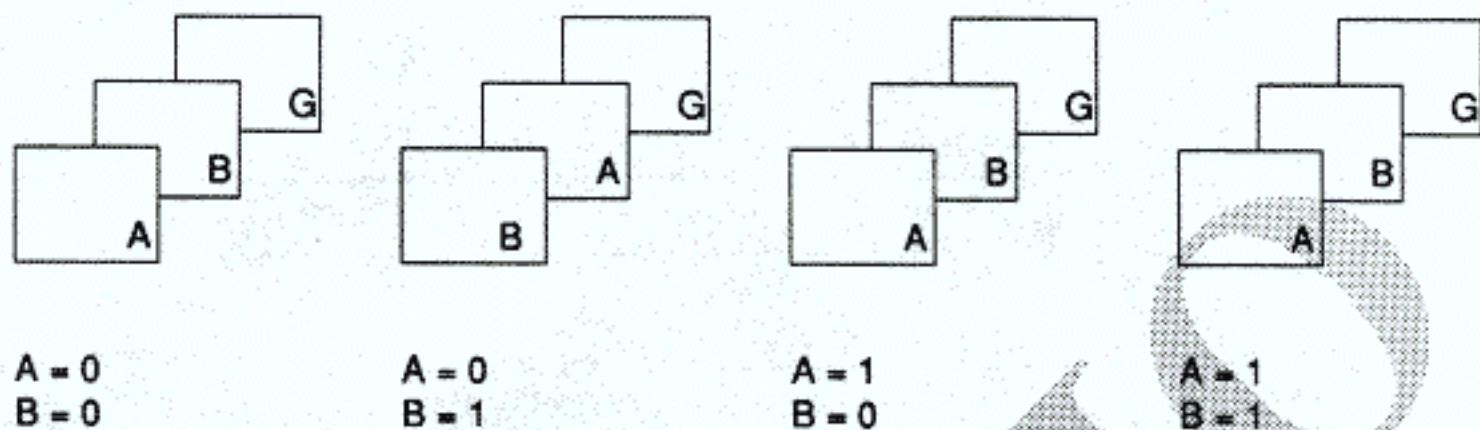
STATUS OF HIGHLIGHT

The dots of Sprite Color Code 15 work as an operator on the screen, the priority of which is lower than Sprite.

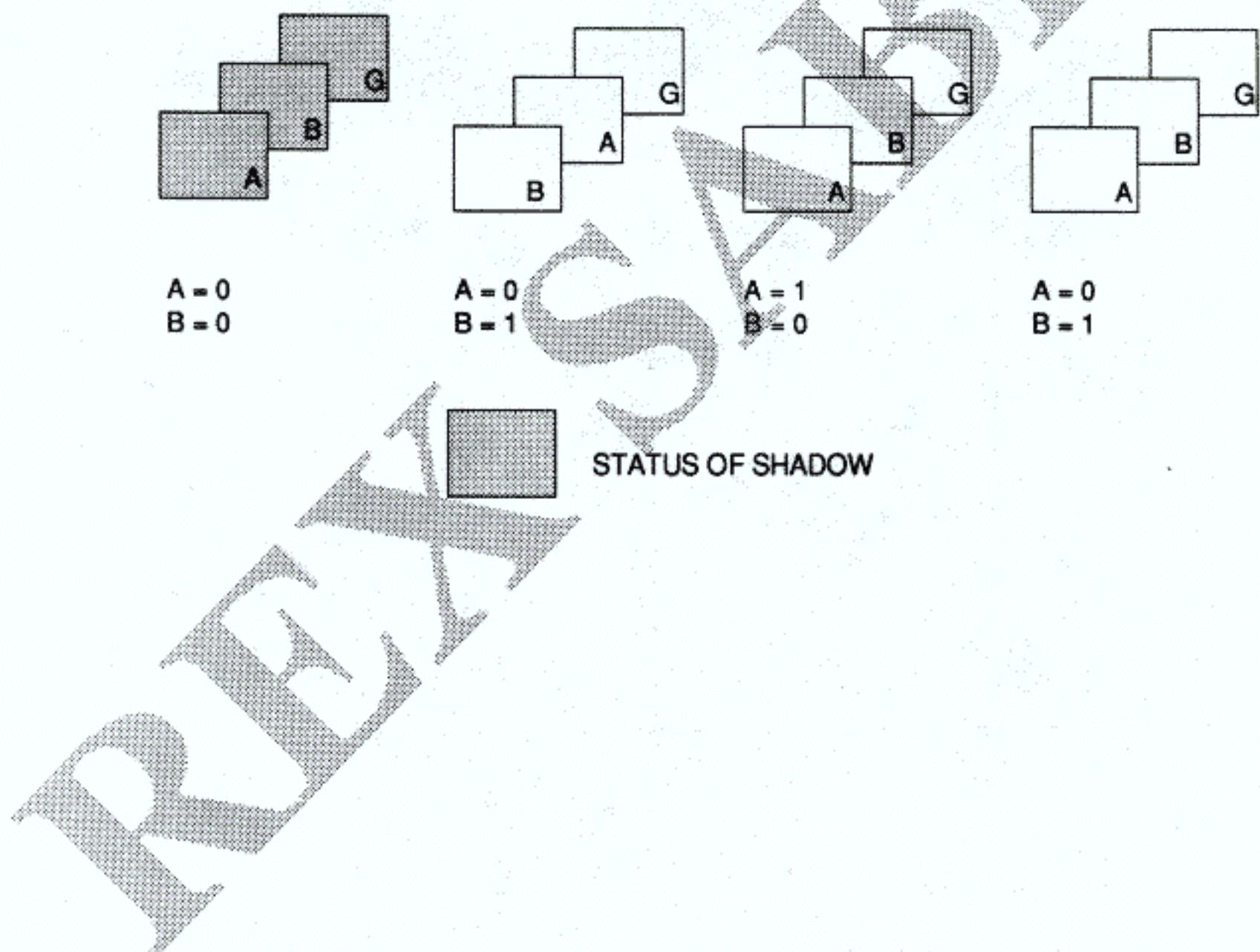
Since Sprite dots work as an operator, this will not be displayed.

When SPRITE is not related to PRIORITY, the following PRIORITY applies:

S/TEN = 0

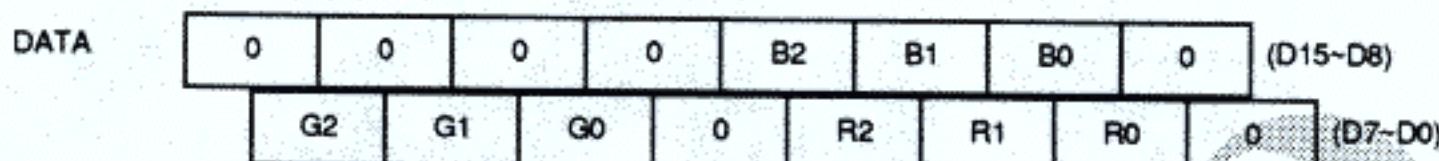


S/TEN = 1



M. COLOR PALETTE

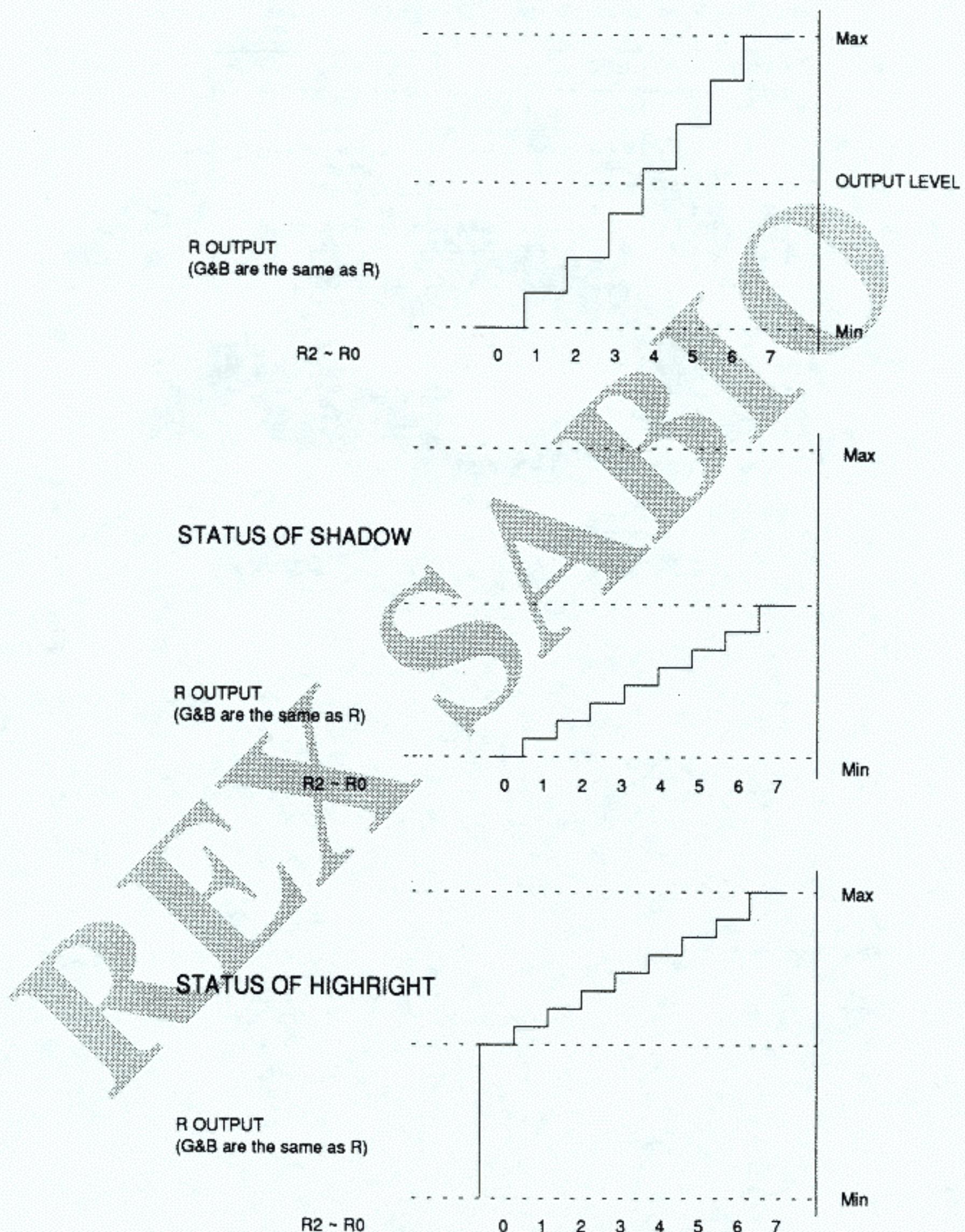
One dot is comprised of 4 bits and can designate the 0-15 colors. Also, 0-3 color palette can be designated by Scroll screen on a cell basis and by each Sprite. CRAM data are as follows. Since each of R, G and B has 3 bits, colors can be freely selected out of 512 colors.



The relationships between CRAM address, palette, and color code are as follows. However, in the case of each palette's color code 0, the color for the Scroll A, Scroll B, Window, and Sprite is See Through irrespective of RGB designation.

ADDRESS	BLUE	GREEN	RED	PALETTE	CODE	REMARKS
00H				0	0	
02H				0	1	
04H				0	2	
:				0	:	
1AH				0	13	
1CH				0	14	
1EH				0	15	
20H				1	0	
:				1	:	
3EH				1	15	Same as Palette 0
40H				2	0	
:				2	:	
5EH				2	15	Same as Palette 0
60H				3	0	
62H				3	1	
7AH				3	:	
7CH				3	13	
7EH				3	14	
				3	15	For 14 and 15, refer to Priority

RGB bit and display are as follows:



N. INTERLACE MODE

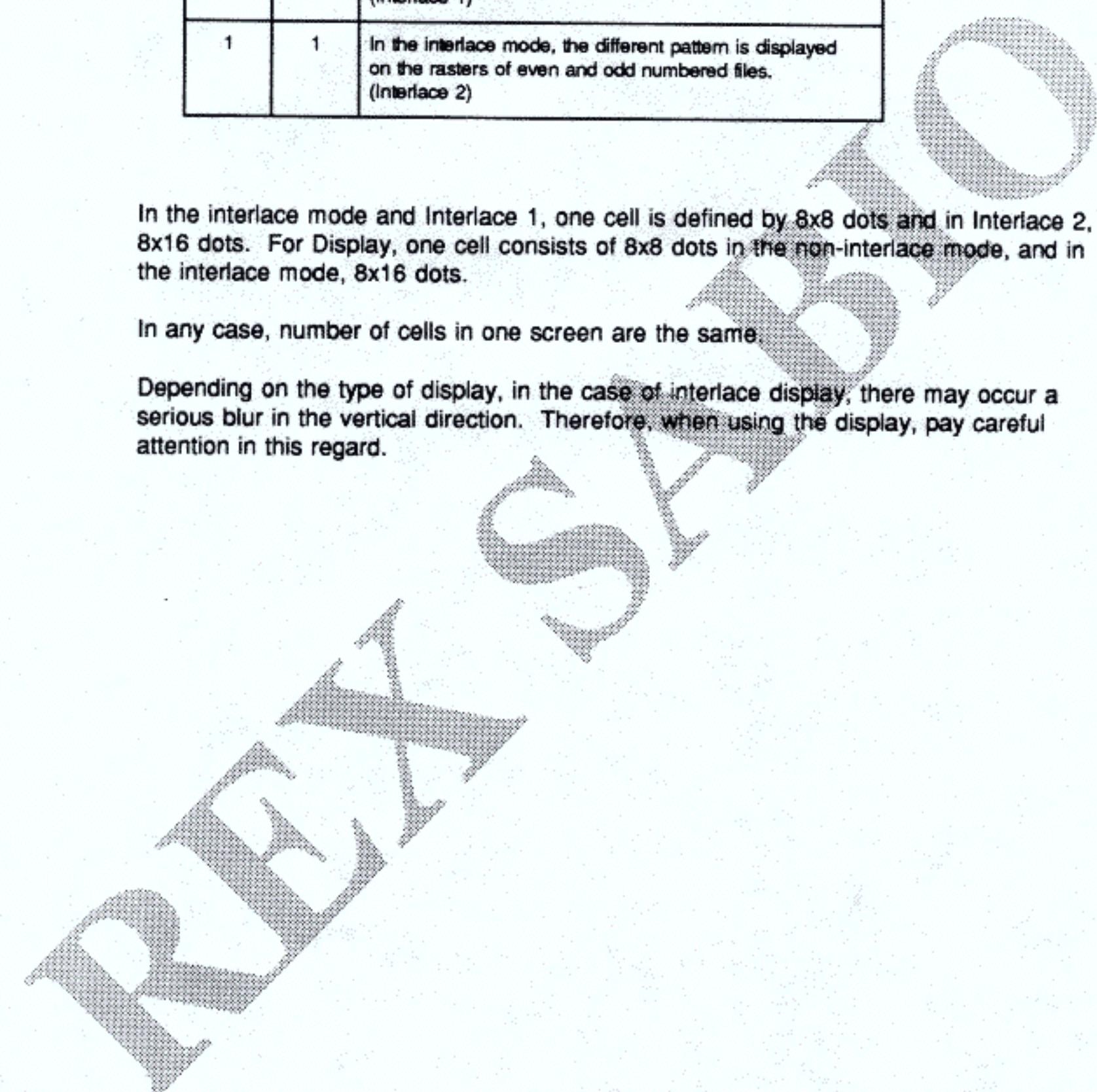
Raster Scan Mode can be changed by setting LSM0 and LSM1 (RGB #12).

LSM1	LSM0	Raster Scan Mode
0	0	Non-interlace mode
0	1	In the non-interlace mode, the same pattern is displayed on the rasters of even and odd numbered files. (Interlace 1)
1	1	In the interlace mode, the different pattern is displayed on the rasters of even and odd numbered files. (Interlace 2)

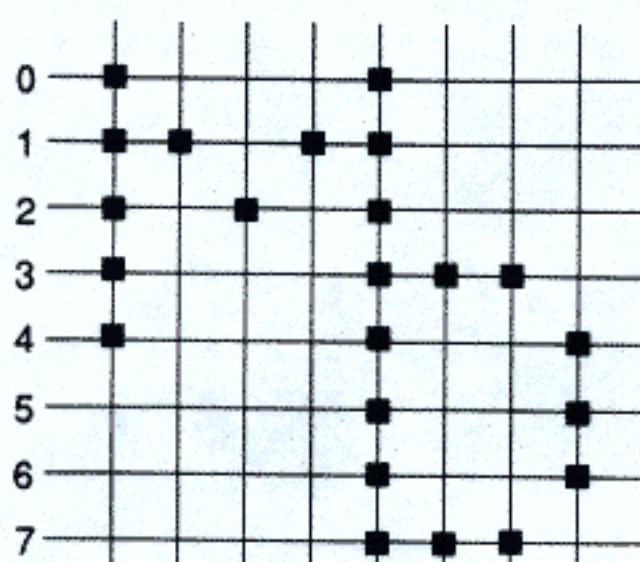
In the interlace mode and Interlace 1, one cell is defined by 8x8 dots and in Interlace 2, 8x16 dots. For Display, one cell consists of 8x8 dots in the non-interlace mode, and in the interlace mode, 8x16 dots.

In any case, number of cells in one screen are the same.

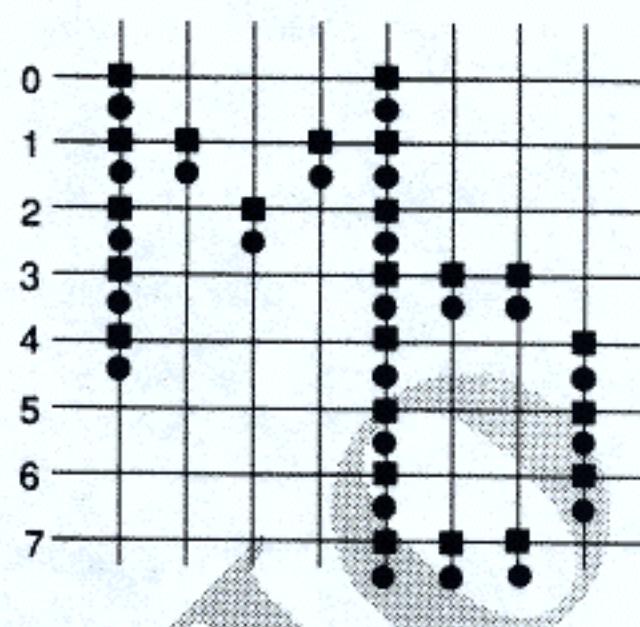
Depending on the type of display, in the case of interlace display, there may occur a serious blur in the vertical direction. Therefore, when using the display, pay careful attention in this regard.



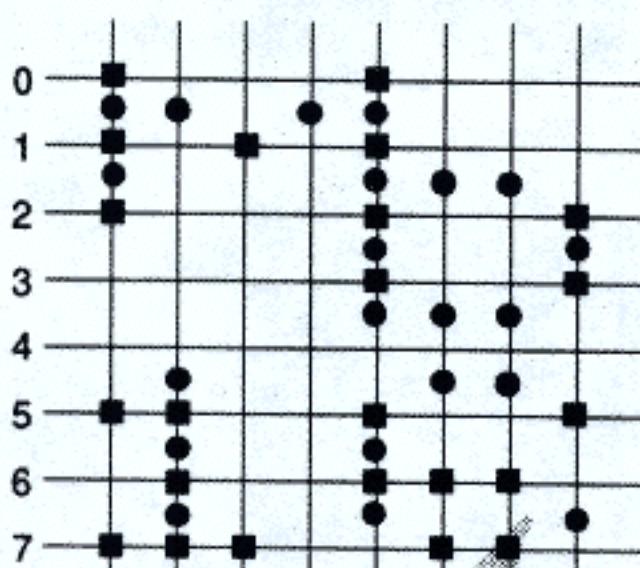
NON-INTERLACE



INTERLACE 1



INTERLACE 2



FIELD NO. 1

FIELD NO. 2

REF

S

III. BACKWARD COMPATIBILITY MODE

In the case of Backward Compatibility Mode, the Mega Drive differs from the original Mark III and Master System in the following points:

A. MARK III (MS-Japan)

OS-ROM is not incorporated

ROM cartridge/card selections are made by hardware in the same manner as in the case of Mark III. Start-up slot number is not written in 0C000H. Start-up SEGA logo is not displayed.

FM sound source is not incorporated

FM sound is incorporated in MS-Japan (standard) and Mark III (optional) (OPLL), however, Mega Drive has no option for that, although connection is possible.

Consider the Mega Drive's Japanese specifications as that of Mark III with MS-Japan's Joystick Port, or as MS-Japan without FM sound source and OS-ROM.

B. MASTER SYSTEM

OS-ROM is not incorporated

0C000H-0DFFFH RAM is not clear on power-up. RAM 0C000 has no meaningful value. Start-up Sega logo is not displayed.

FM sound source is not incorporated

FM sound source is incorporated in MS by option (OPLL). However, Mega Drive has no option, although connection is possible. Please regard the Mega Drive overseas version as a Master System without an Operating System ROM.

C. RAM BOARD

In the Mega Drive's Mark III and Master System backward compatibility mode, the RAM board for development (for which D-RAM was used) cannot be used due to the problem of Refresh. The other boards for development (which utilize S-RAM) can be used without any problem.

IV. SYSTEM I/O

Mega Drive System I/O area assignment starts from \$A00000, with the Z80 sub-CPU's memory area.

A. VERSION NUMBER

Indicates the Mega Drive's hardware version.

\$A10001

MODE	VMOD	DISK	RSV	VER3	VER2	VER1	VER0
------	------	------	-----	------	------	------	------

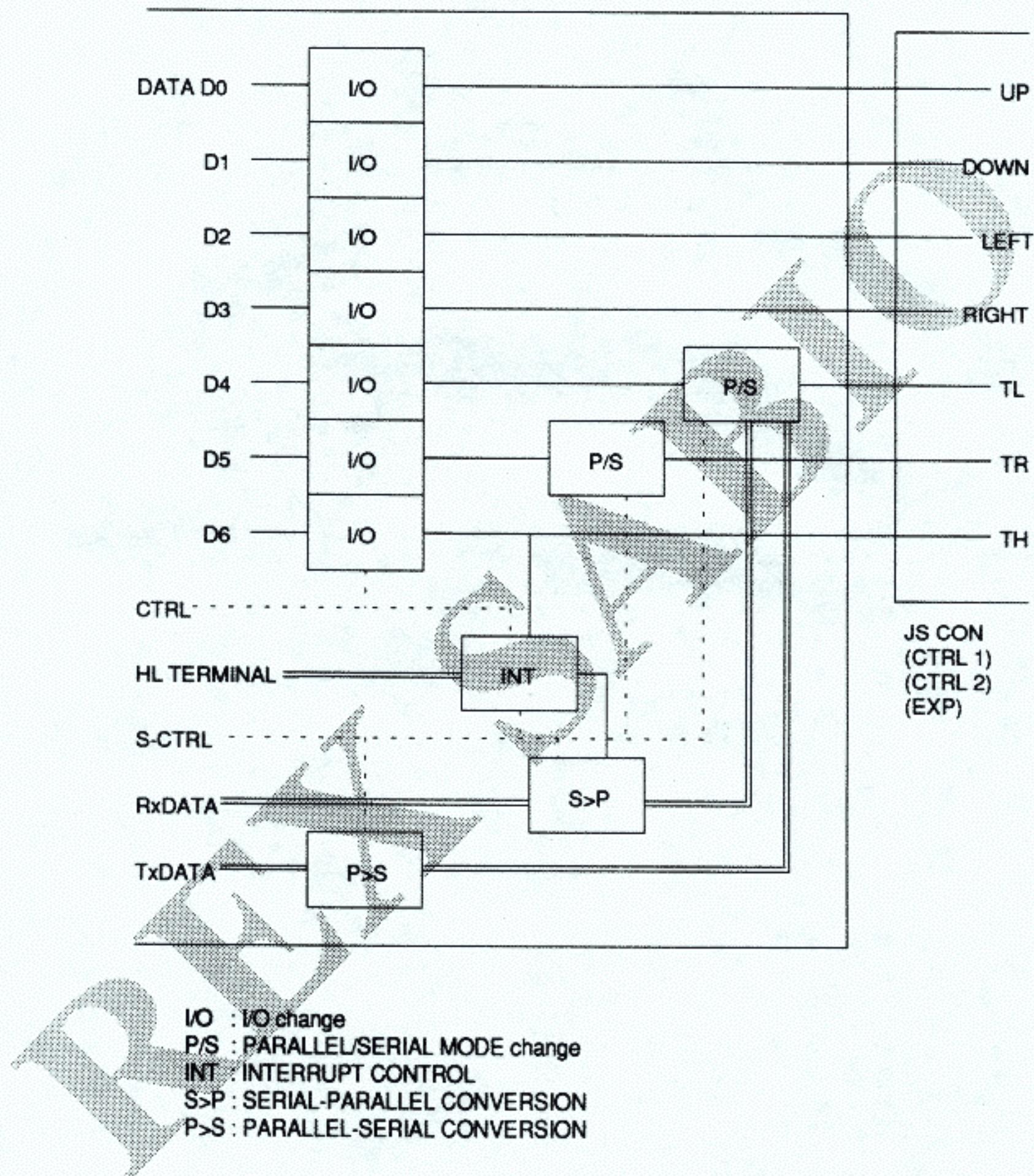
Mode (R)	0: Domestic Model 1: Overseas Model
VMOD (R)	0: NTSC CPU clock 7.67 MHz 1: PAL CPU clock 7.60 MHz
Disk (R)	0: FDD unit connected 1: FDD unit not connected
RSV (R)	Currently not used
VER 3~0 (R)	Mega Drive version is indicated by \$0~\$F. The present hardware version is indicated by \$0.

B. I/O PORT

The Mega Drive has the three general purpose I/O ports: Ctrl 1, Ctrl 2, and Exp. Although each port differs from the others in physical shape, it functions in the same manner. Each port has the following five registers for control.

Data	(Parallel data)	R/W
Ctrl	(Parallel control)	R/W
S-Ctrl	(Serial control)	R/W
TxDATA	(Tx data)	R/W
RxDATA	(Rx data)	R

The relationship between REGISTERS is as follows:



Mapping is as follows:

\$A10003	:	Data 1	(Ctrl 1)
\$A10005	:	Data 2	(Ctrl 2)
\$A10007	:	Data 3	(Exp)
\$A10009	:	Ctrl 1	
\$A1000B	:	Ctrl 2	
\$A1000D	:	Ctrl 3	
\$A1000F	:	TxDATA 1	
\$A10011	:	RxDATA 1	
\$A10013	:	S-Ctrl 1	
\$A10015	:	TxDATA 2	
\$A10017	:	RxDATA 2	
\$A10019	:	S-Ctrl 2	
\$A1001B	:	TxDATA 3	
\$A1001D	:	RxDATA 3	
\$A1001F	:	S-Ctrl 3	

Both Byte and Word access are possible. However, in the case of Word access, only the lower byte is meaningful.

Data shows the status of each port. The I/O direction of each bit is set by Ctrl and S-Ctrl.

DATA	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	PD7 (RW)	PD6 (RW)	PD5 (RW)	PD4 (RW)	PD3 (RW)	PD2 (RW)	PD1 (RW)	PD0 (RW)
	TH	TR	TL					
					RIGHT	LEFT	DOWN	UP

Ctrl designates the I/O direction of each port and the Interrupt Control of TH.

CTRL	INT	PC6	PC5	PC4	PC3	PC2	PC1	PC0
------	-----	-----	-----	-----	-----	-----	-----	-----

INT	(RW)	0: TH-INT Prohibited 1: TH-INT Allowed
PC6	(RW)	0: PD6 Input Mode 1: PD6 Output Mode
PC5	(RW)	0: PD5 Input Mode 1: PD5 Output Mode
PC4	(RW)	0: PD4 Input Mode 1: PD4 Output Mode
PC3	(RW)	0: PD3 Input Mode 1: PD3 Output Mode
PC2	(RW)	0: PD2 Input Mode 1: PD2 Output Mode
PC1	(RW)	0: PD1 Input Mode 1: PD1 Output Mode
PC0	(RW)	0: PD0 Input Mode 1: PD0 Output Mode

S-Ctrl is for the status, etc. of each port's mode change, baud rate, and serial.

Ctrl designates the I/O direction of each port and the Interrupt Control of TH.

S-CTRL	BPS1	BPS0	SIN	SOUT	RINT	RERR	RRDY	TFUL
--------	------	------	-----	------	------	------	------	------

SIN	(RW)	0: TR - Parallel Mode 1: TR - Serial In
SOUT	(RW)	0: TL - Parallel Mode 1: TL - Serial Out
RINT	(RW)	0: Rxd Ready - Interrupt Prohibited 1: Rxd Ready - Interrupt Allowed
RERR	(R)	0: 1: RxdError
RRDY	(R)	0: 1: RxdReady
TFUL	(R)	0: 1: TxdFull

BPS1	BPS0	bps
0	0	4800
0	1	2400
1	0	1200
1	1	300

C. MEMORY MODE

The Mega Drive is able to generate internally the Refresh signal for the D-RAM development cartridge. When using the development cartridge, set to D-RAM mode. In the case of a production cartridge, set to ROM mode.

Only D8 of address \$A11000 is effective and for Write only.

\$A11000 D8 (W) 0: ROM mode
1: D-RAM mode

Access to \$A11000 can be based on byte.

D. Z80 CONTROL

1. Z80 BusReq

When accessing the Z80 memory from the 68000, first stop the Z80 by using BusReq. At the time of power on reset, the 68000 has access to the Z80 bus.

\$A11100 D8 (W) 0: BusReq cancel
1: BusReq request
(R) 0: CPU function stop, accessible
1: Functioning

Access to Z80 area in the following manner.

- Write \$0100 in \$A11100 by using a Word access.
- Check to see that D8 of \$A11100 becomes 0.
- Access to Z80 area.
- Write \$0000 in \$A11100 by using a Word access.

Access to \$A11100 can also be based on byte.

2. Z80 Reset

The 68000 may also reset the Z80. The Z80 is automatically reset during the Mega Drive hardware's power on reset sequence.

\$A11200 D8 (W) 0: Reset request
1: Reset cancel

Access to \$A11100 can also be based on byte.

E. Z80 AREA

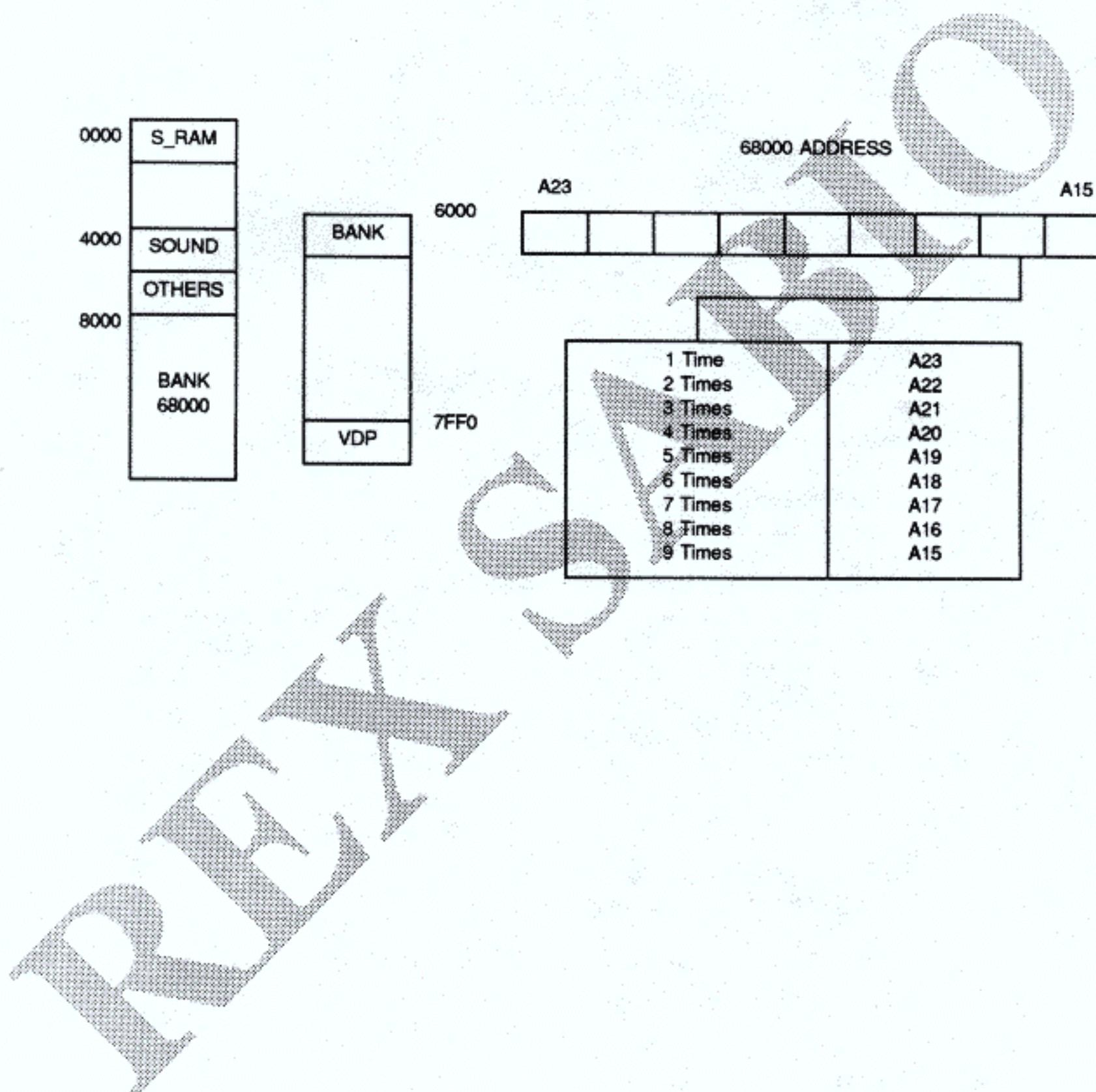
Mapping is performed starting from \$A00000 for Z80, a sub-CPU. As viewed from 68000, the memory map will be as follows:

\$A00000	Sound RAM
\$A02000	Prohibited
\$A04000	
\$A04004	Sound Chip
\$A06000	Prohibited
\$A06002	Bank Register
\$A08000	Prohibited
	Prohibited

1. **Sound RAM**
This is for the Z80 program. Access from 68000 by byte.
2. **Sound Chip**
This is the mapping area for FM sound source (YM2612). When accessing from 68000, use byte, due to timing problem.
3. **Bank Register**
Access to the 68000 side Memory Area from Z80 will be based on a 32 K byte unit. At this time, this register sets which bank is to be accessed. Registering from 68000 can be set, however, do not access to Z80 Bank Memory Area by 68000.

Setting Method

When accessing to the 68000 side addresses from Z80 side, all the addresses can be classified into Banks. Bank can be set by writing 9 times in 0 bit of 6000 (Z80 address). The 9 bits correspond to 68000 address 15-23 as shown below:



V. VRAM MAPPING

In VRAM, there are various tables and pattern generators as stated below. Among those, the base address of Pattern Generator Table and Sprite Generator Table are 0000H and fixed. However, the other base addresses can be freely assigned in VRAM by setting VDP Register. Also, Area can be overlapped, therefore, Table can be commonly used by Scroll screen and Window, for example.

- Scroll A Pattern Name Table, maximum 8 K byte
Base address designated by Register #2.
- Scroll B Pattern Name Table, maximum 8 K byte
Base address designated by Register #4.
- Window Pattern Name Table, varies by H resolution
Base address designated by Register #3.
- H Scroll Data Table, 1 K byte
Base address designated by Register #13.
- Sprite Attribute Table, varies by H resolution
Base address designated by Register #5.
- Pattern Generator Table
Base address is 0000H (fixed).
- Sprite Generator Table
Base address is 0000H (fixed).

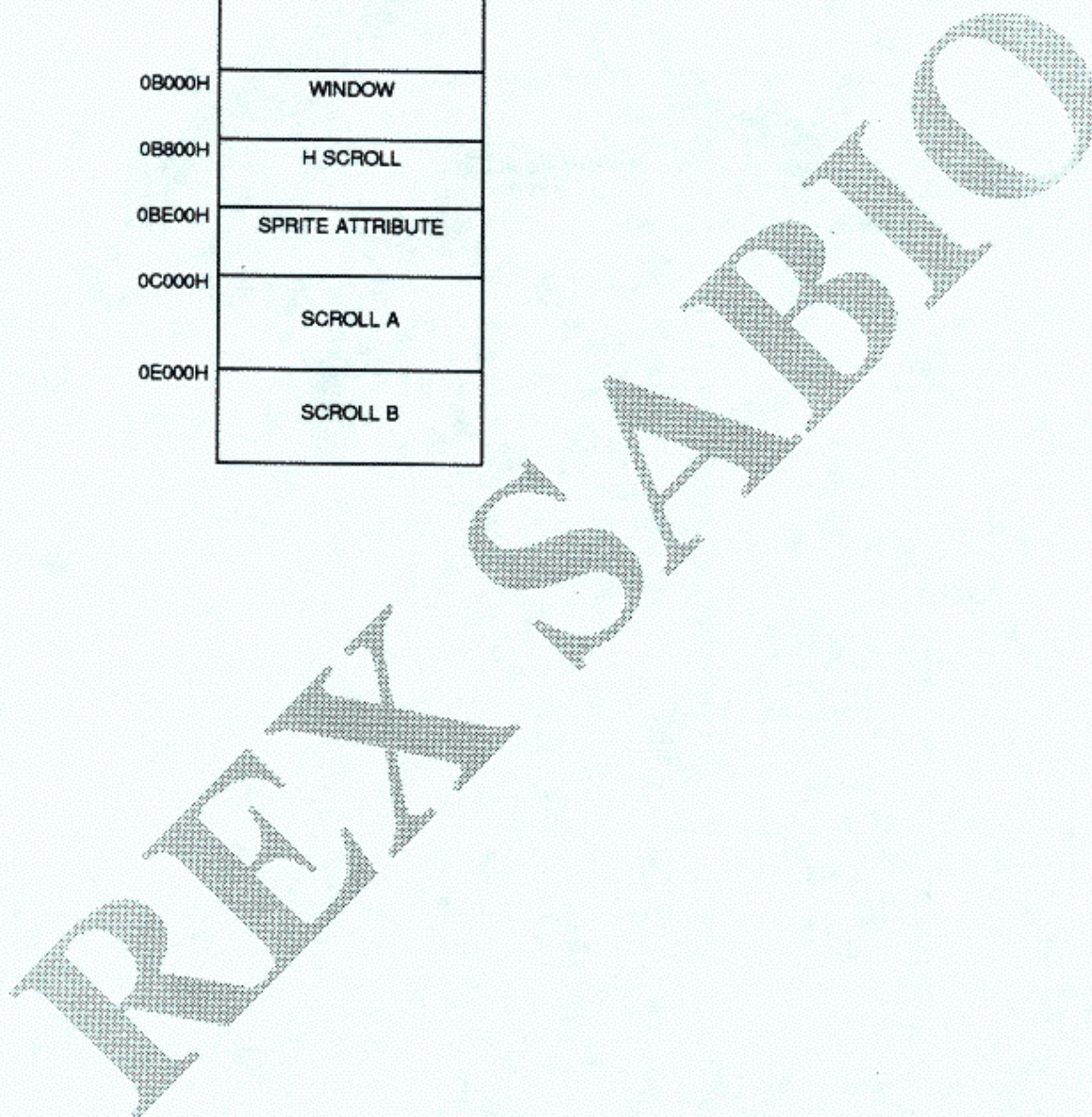
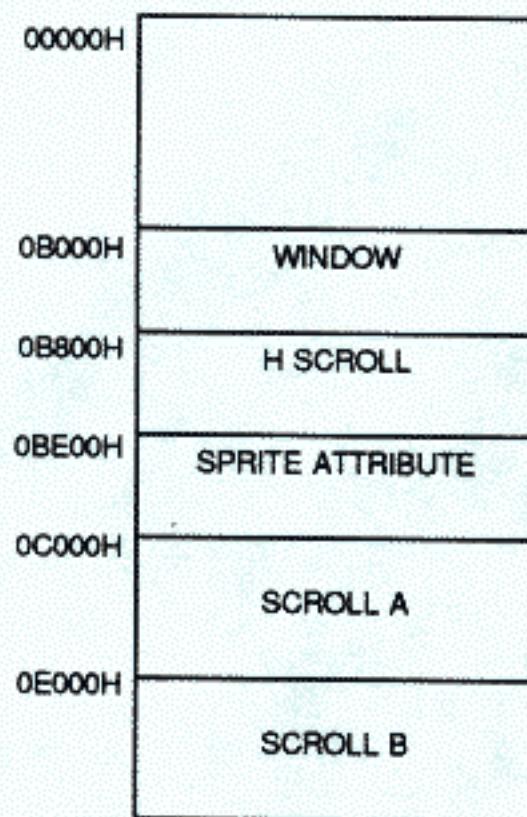
There are 1 K bytes for H Scroll Table, however, as for display 896 bytes in V28 Cell Mode and 960 bytes in V30 Cell Mode. There are 2 K bytes for Window Pattern Name Table in H32 Cell Mode, and 4 K byte area in H40 Cell Mode. For details, refer to Window. There are 512 bytes for Sprite Attribute Table in H32 Cell and 1 K byte area in H40 Cell Mode. However, as for display, there are 640 bytes in H40 Cell Mode.

Setting examples

H32 Cell Mode

- Scroll A Pattern Name Table
8 K bytes from 0C000H: Register #2 = \$30
- Scroll B Pattern Name Table
8 K bytes from 0E000H: Register #4 = \$07
- Window Pattern Name Table
2 K bytes from 0B000H: Register #3 = \$2C
- H Scroll Data Table
1 K byte from 0B800H: Register #13 = \$2E
- Sprite Attribute Table
512 bytes from 0BE00H: Register #5 = \$5F

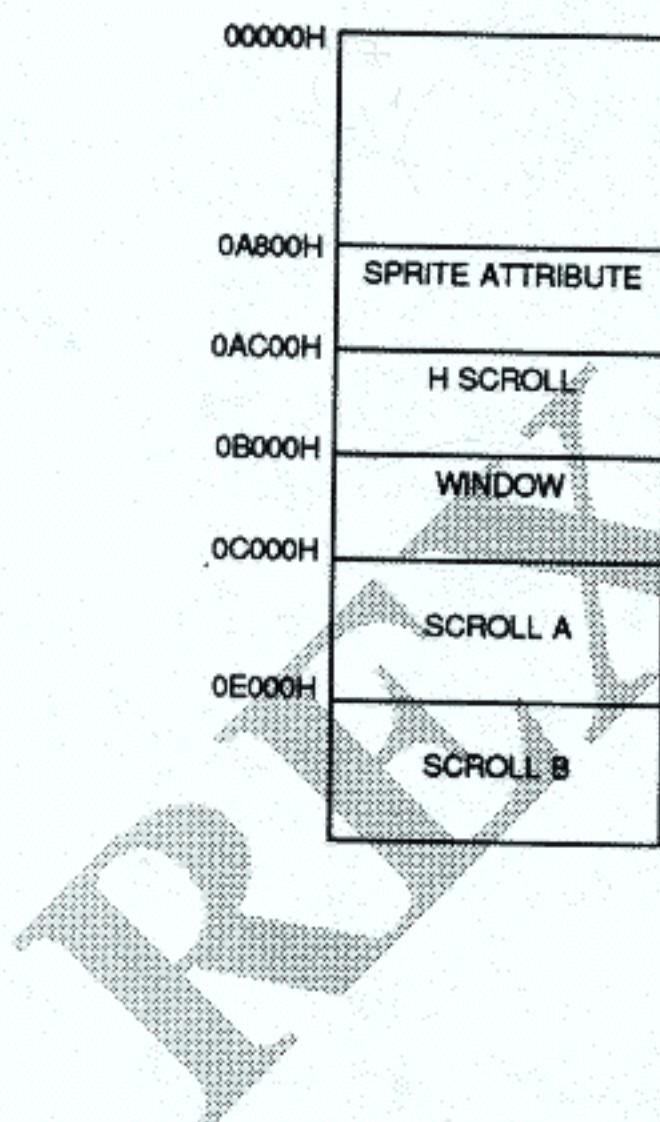
Unoccupied area is used as Pattern Generator and Sprite Generator.



H40 Cell Mode

- Scroll A Pattern Name Table
8 K bytes from 0C000H: Register #2 = \$30
- Scroll B Pattern Name Table
8 K bytes from 0E000H: Register #4 = \$07
- Window Pattern Name Table
4 K bytes from 0B000H: Register #3 = \$2C
- H Scroll Data Table
2 K bytes from 0AC00H: Register #13 = \$2B
- Sprite Attribute Table
1 K byte from 0A800H: Register #5 = \$54

Unoccupied area is used as Pattern Generator and Sprite Generator.



Precautions for M5 Software Programming

When programming the M5 software, pay attention to the following:

1. The program of DMA (RAM, ROM→VRAM, CRAM, VSRAM) should be resident in RAM or it should be as in LIST1 for example. However, in either one of the above two cases, a long word access is not possible as regards the last VRAM address set.
2. ID should be as in the next page.
3. Put LIST2 at your program's start. This is the U.S. security software.

LIST 1

DMA-RAM:

```
lea           ; vdp_cmd = $c00000
              ; An = Address Register
; Set source Address to VDP Register
; Set Data Length to VDP Register
move.1 xx,ram0      ; xx: Destination Address
                     ; ram0: Work RAM
move.w ram0, (An)
move.w ram0+2, (An) ; Pay careful attention to the sequential order of
                     ; 1st word and 2nd word.
                     ; Destination Address should be set by Word and
                     ; not by Long Word.
```

LIST 2

```
move.b $a10001,d0    ; Get version number
andi.b #$0f,d0
beq.b ?0             ; If not version #0
move.1 #'SEGA' ,$a14000 ; Output ASCII
?0:
```


10. When no external RAM is mounted, fill the address by a space code; when it is mounted, do the following:

1B0H: dc.b 'RA',%1x1yz000,%00100000
x 1 for Backup and 0 if not Backup
yz 10 if even address only, 11 if odd address only
1B4H: dc.l RAM start address, RAM end address

11. If corresponding to modem, fill it by space code; if not, do the following

1BCH: dc.b 'MO','xxxx','yy.z'
xxxx Firm name, the same as in 2
yy Modem number
z Version

13. Data on the countries in which the product can be released

Japan : J
USA : U
Europe : E

Be sure to input a space code in the unoccupied 1-7, 9-13 space.

How to Obtain a Check Sum

The Check Sum obtaining program is shown as follows. The program starts with 0FF8000H, RAM space.

First, fill game capacity by -1 (0FFH) and then load all of the programs. Next, load the Check Sum program and run the program from 0FF8000H.

After a while, stop running the program. At this time, the lower Word of Data Register 0 (d0) is the Check Sum value. Note that Break in Memory should be cancelled in advance.

Also, when burn-ins to ROM, first fill the game capacity by -1 (0FFH).

```
end_addr equ $1a4
org -$8000
start:
    move.1 (a0),d1
    addq.1 #$1,d1
    movea.1 #$200,a0
    sub.1 a0,d1
    asr.1 #1,d1
    move d1,d2
    subq.w #$1,d2
    swap d1
    moveq #$0,d0
•?12:
    add (a0)+,d0
    dbra d2,?12
    dbra d1,?12
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
?1e:
    nop
    nop
    bra.b ?1e
```

Memory Mapping for Emulation

For the 68000 Emulation

All address should be disabled initially: 0 to 0FFFFFF

Required areas should then be enabled as follows:

1. Program and Data are in 0 to 007FFFF
2. S-RAM is for Z80 in 0A00000 to 0A01FFF
3. FM sound chip interface is in 0A04000 to 0A04FFF
4. I/O and Z80 control port are in 0A10000 to 0A11FFF
5. VDP and sound control port are in 0C00000 to 0C00FFF
6. Scratch RAM is in 0FF0000 to 0FFFFFF

RAM Card (No. 171-5642-02)

This board has two memory areas:

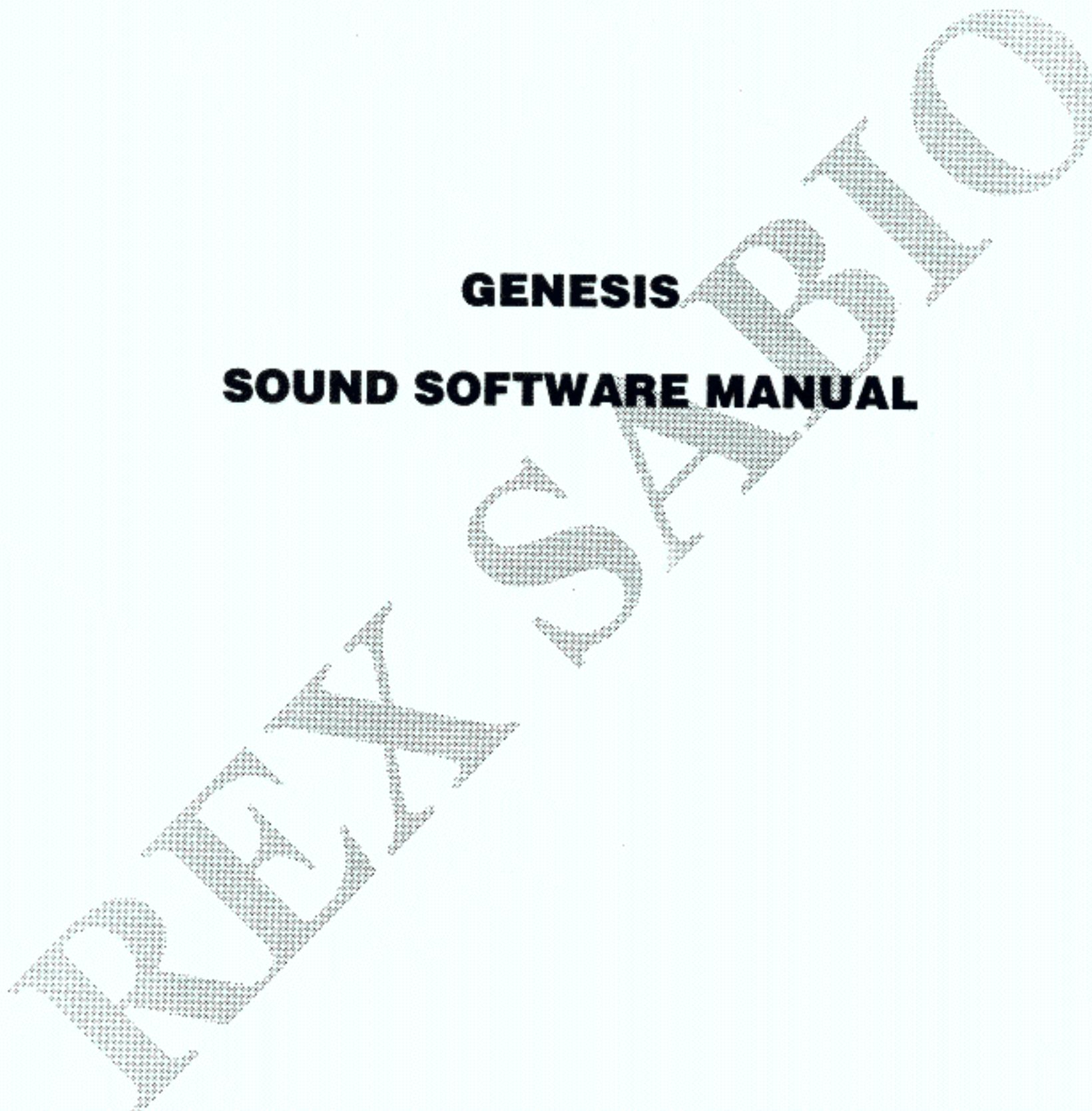
Main Memory	(D-RAM) \$000000 - \$0FFFFFF
Backup Memory	(S-RAM) \$200000 - \$203FFF

1. Initialize
Write 0100H into \$0A11000
Write 1 into \$0A130F0
(Green LED light up)
2. Write Protect
Write 3 into \$0A130F0
(Red LED light up)
3. Read/Write
Write 1 into \$0A130F0
(Red LED turns off)
4. Note: Emulator access to these ports should be enabled before the writes, then disabled after words.

Mega Drive Registers Fixed Bits (40 Cell and NTSC Mode)

RO	0	0	0		0	1	0	0
1	0				0	1	0	0
2	0	0				0	0	0
3	0	0						0
4	0	0	0	0	0			
5	0							
6	0	0	0	0	0	0	0	0
7	0	0						
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10								
11	0	0	0	0	0			
12	1	-	-	-				1
13	0	0						
14	0	0	0	0	0	0	0	0
15								
16	0	0			0	0		
17	0	0						
18	0	0						
19								
20								
21								
22								
23								

* DMA cannot be performed emulated ROM or RAM on most ICs.



GENESIS
SOUND SOFTWARE MANUAL

INDEX

I. Z80 MAPPING

- A. Z80 Memory Map
- B. Interrupt

II. 68K CONTROL OF Z-80

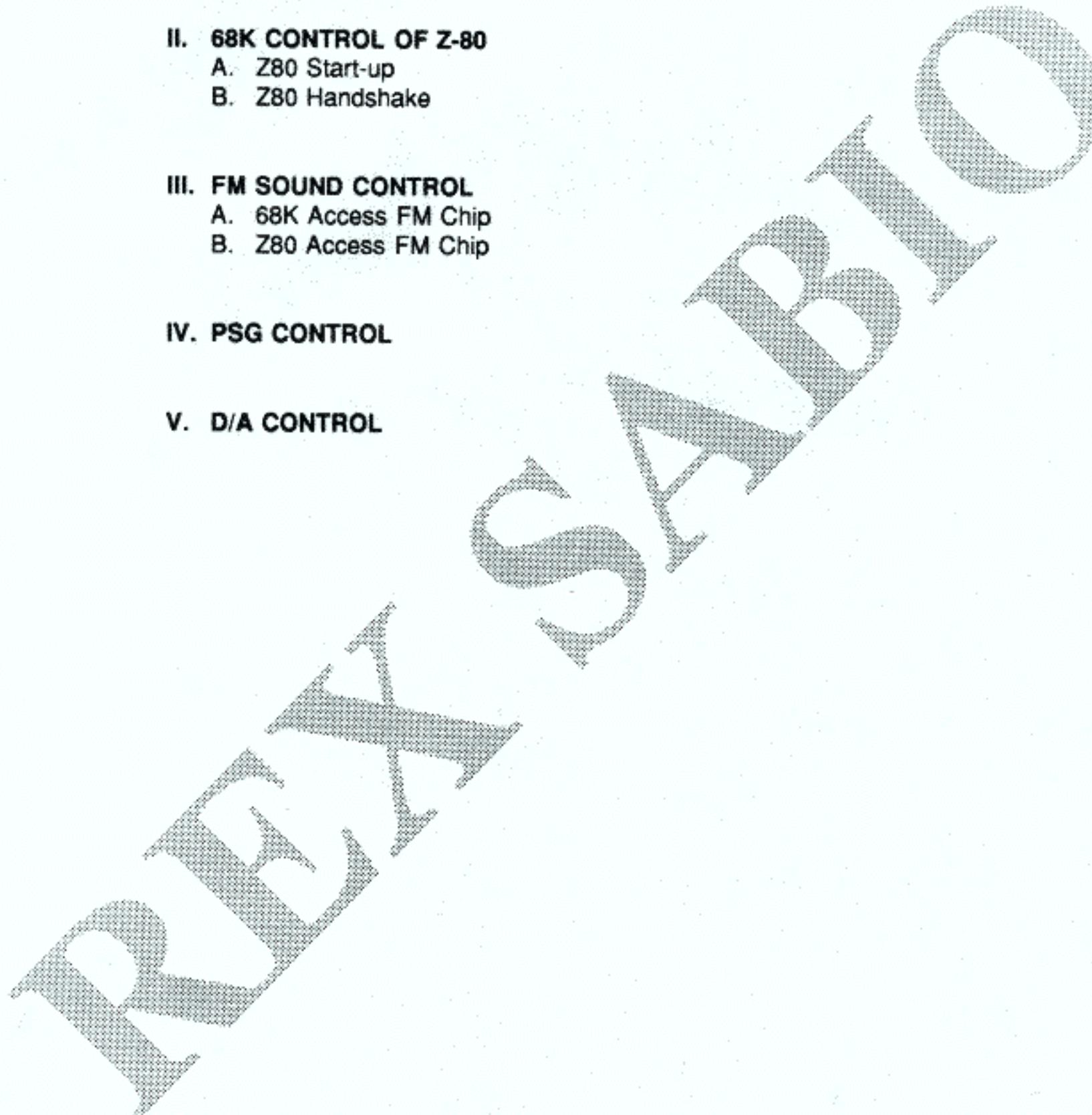
- A. Z80 Start-up
- B. Z80 Handshake

III. FM SOUND CONTROL

- A. 68K Access FM Chip
- B. Z80 Access FM Chip

IV. PSG CONTROL

V. D/A CONTROL



This manual explains memory mapping and way of accessing especially. FM sound generation and PSG are explained in another manual.

SUPPLEMENTARY INDEX

I. FM AUDIO – YM2612

- A. Overview
- B. A little bit about operators
- C. Register Overview
- D. Envelope Specification
- E. Part 1 Memory Map
- F. Test Program

II. PROGRAMMABLE SOUND GENERATOR (PSG)

- A. Tone Generator Frequency
- B. Noise Generator Control
- C. Attenuators

This supplementary index was provided to allow direct access to the content on this PDF. It was not included in the original document.

I. Z80 MAPPING

A. Z80 Map

We show the memory at right.
I/O is contained in memory map.

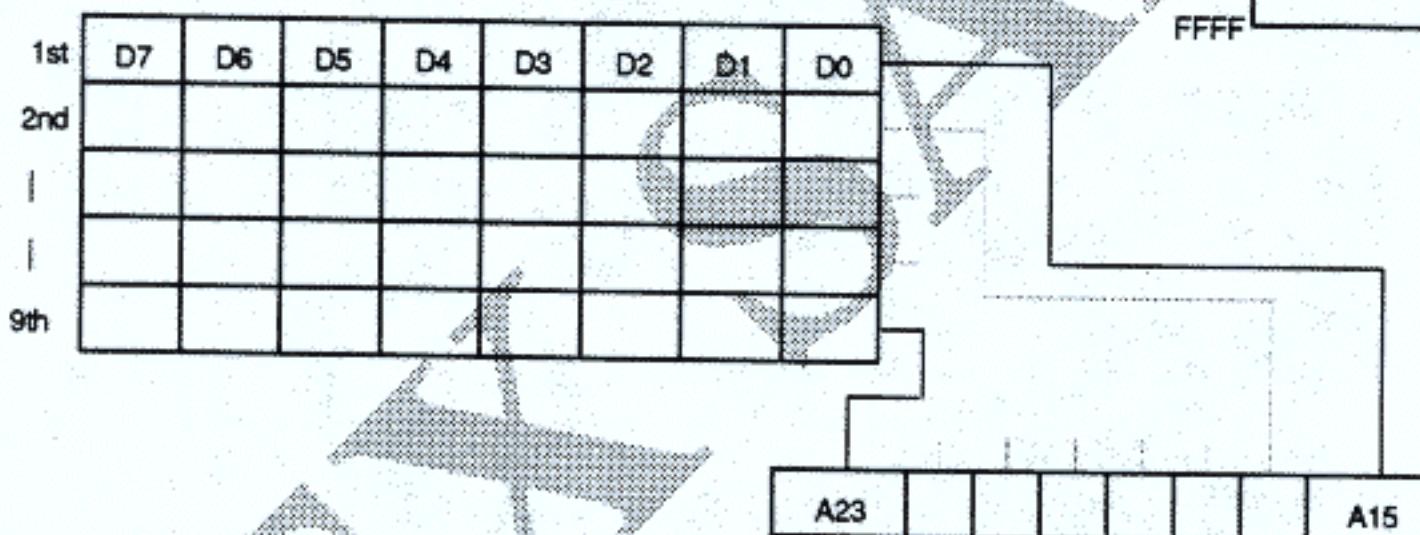
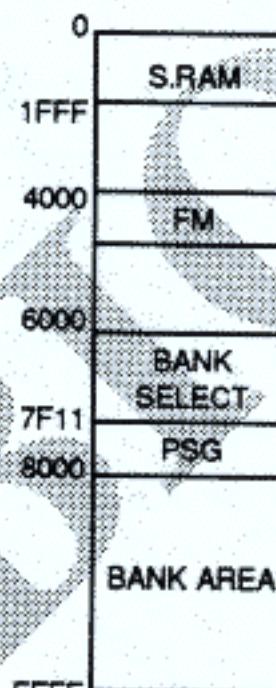
1. Program Area

Program, data and scratch
are in 0 to 1FFFH, in S-RAM.

2. BANK

From 8000H-FFFFH is window of
68K memory. Z-80 can access
all of 68K memory by BANK
switching. BANK select data
create 68K address from A15
to A23. You must write these
9 bits one at a time into 6000H
serially, byte units, 7F11 using
the LSB.

Z80 ADDRESS



3. I/O

4000H FM1 register select (Channel 1-3)

4001H FM1 DATA

4002H FM2 register select (Channel 4-6)

4003H FM2 DATA

PSG address is in 7F11H.

B. Interrupt

Z-80 gets the only VIDEO vertical interrupt. This interrupt is generated 16ms period and 64ms length.

II. 68K CONTROL OF Z-80

A. Z80 Start-Up

Z-80 Operation Sequence:

1. BUS REQ ON
2. BUS RESET OFF
3. 68K copies program into Z-80 S-RAM
4. BUS RESET ON
5. BUS REQ OFF
6. BUS RESET OFF

BUS REQUEST

- BUS REQ ON
DATA 100H (WORD) → \$A11100
- BUS REQ OFF
DATA 0H (WORD) → \$A11100

RESET Z-80

- RESET ON
DATA 0H (WORD) → \$A11200
- RESET OFF
DATA 100H (WORD) → \$A11200

This period requires 26ms.
Also FM sound source is cleared at the same time.

CONFIRMATION OF BUS STATUS

This information is in \$A11100, bit 0.

- 0 - Z-80 is using
- 1 - 68K can access

B. Z80 Handshake

If you access the HANDSHAKE area (A00000 - A07FFF) you must use BUS REQ. 68K has to access the Z-80 S-RAM by byte.

III. FM SOUND CONTROL

A. 68K Accesses the FM Source

68K needs BUS REQ when accessing the FM source, because this memory is controlled by Z-80.

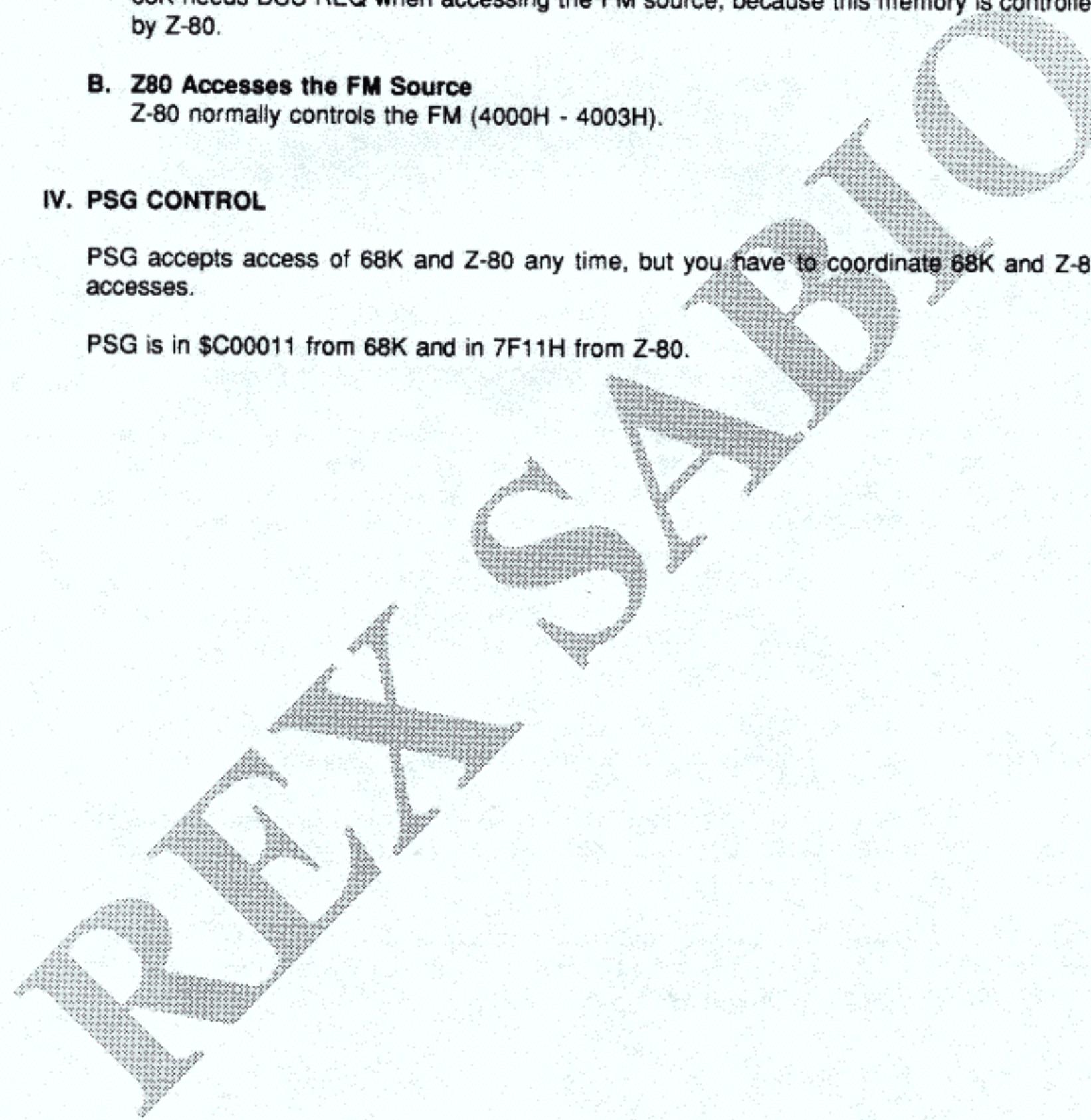
B. Z80 Accesses the FM Source

Z-80 normally controls the FM (4000H - 4003H).

IV. PSG CONTROL

PSG accepts access of 68K and Z-80 any time, but you have to coordinate 68K and Z-80 accesses.

PSG is in \$C00011 from 68K and in 7F11H from Z-80.



OVERVIEW

The Yamaha 2612 Frequency Modulation (FM) sound synthesis IC resembles the Yamaha 2151 (used in SEGA's coin-operated machines) and the chips used in Yamaha's synthesizers.

Its capabilities include:

- 6 channels of FM sound
- An 8-bit Digitized Audio channel (as replacement for one of the FM channels)
- Stereo output capability
- One LFO (low frequency oscillator) to distort the FM sounds
- 2 timers, for use by software.

To define these terms more carefully, an FM channel is capable of expressing, with a high degree of realism, a single note in almost any instrument's voice. Chords are generally created by using multiple FM channels.

The standard FM channels each have a single overall frequency and data for how to turn this frequency into the complex final waveform (the voice). This conversion process uses four dedicated channel components called "operators," each possessing a frequency (a variant of the overall frequency), an envelope, and the capability to modulate its input using the frequency and envelope. The operator frequencies are offsets of integral multiples of the overall frequency.

There are two sets of three FM channels, named channels 1 to 3 and 4 to 6, respectively. Channels 3 and 6, the last in each set, have the capability to use a totally separate frequency for each operator rather than offsets of integral multiples. This works well (we believe) for percussion instruments, which have harmonics at odd multiples such as 1.4 or 1.7 of the fundamental.

The 8-bit Digitized Audio Channel (DAC) exists as a replacement of FM channel 6, meaning that turning on the DAC turns off FM channel 6. Unfortunately, all timing must be done by software — meaning that unless the software has been very cleverly constructed, it is impossible to use any of the FM channels at the same time as the DAC.

Stereo output capability means that any of the sounds, FM or DAC, may be directed to the left, the right, or both outputs. The stereo is output only through the headphone jack.

The LFO, or Low Frequency Oscillator, allows for amplitude and/or frequency distortions of the FM sounds. Each channel elects the degree to which it will be distorted by the LFO, if at all. This could be used, for example, in a guitar solo.

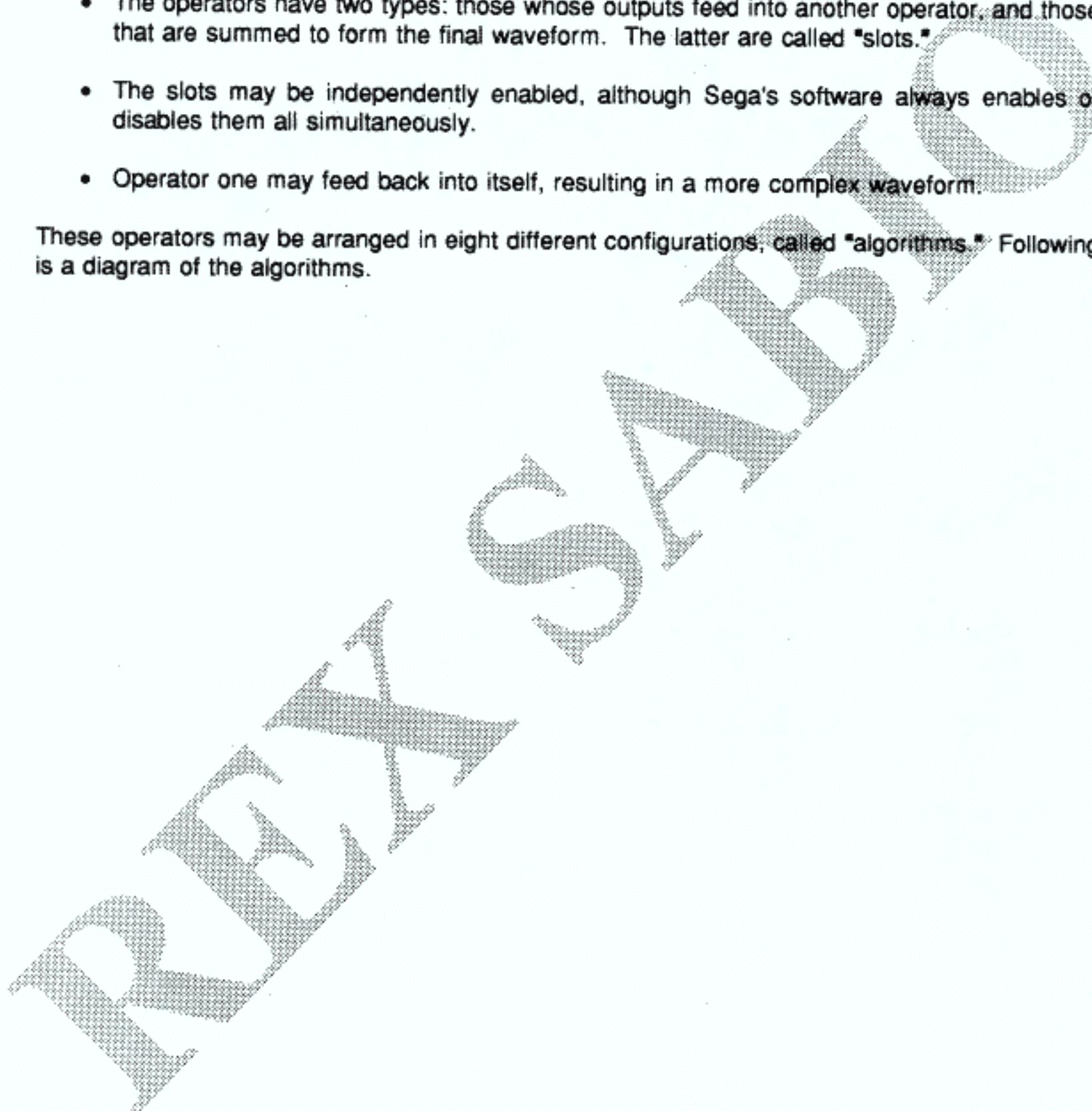
Finally, the system has two software timers which may be used as an alternative to the Z80 VBLANK interrupt. Unfortunately, these two timers do not cause interrupts — they must be read by the software to determine if they have finished counting.

A LITTLE BIT ABOUT OPERATORS

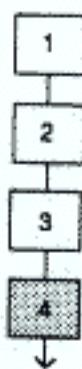
There are four dedicated operators assigned to every channel, with the following properties:

- An operator has an input, a frequency and an envelope (with which to modify the input), and an output.
- The operators have two types: those whose outputs feed into another operator, and those that are summed to form the final waveform. The latter are called "slots."
- The slots may be independently enabled, although Sega's software always enables or disables them all simultaneously.
- Operator one may feed back into itself, resulting in a more complex waveform.

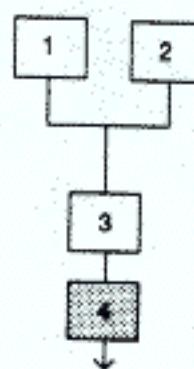
These operators may be arranged in eight different configurations, called "algorithms." Following is a diagram of the algorithms.



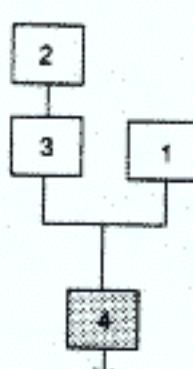
ALGORITHM #0



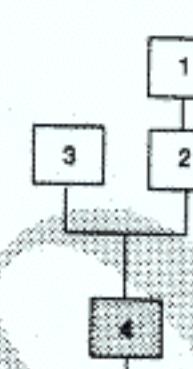
#1



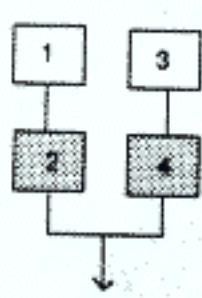
#2



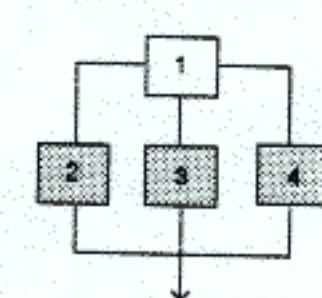
#3



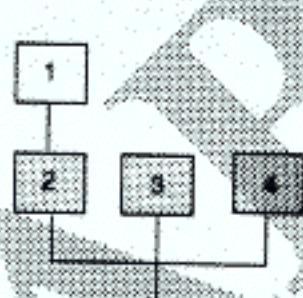
#4



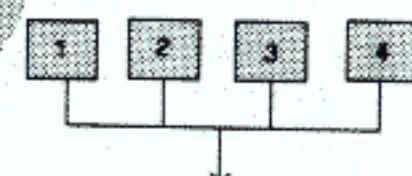
#5



#6



#7



SLOTS ARE INDICATED BY SHADING

Algorithm 0 - distortion guitar, "high hat chopper" (?) bass

Algorithm 1 - harp, PSG (Programmable Sound Generator) sound

Algorithm 2 - Bass, electric guitar, brass, piano, woods

Algorithm 3 - strings, folk guitar, chimes

Algorithm 4 - flute, bells, chorus, bass drum, snare drum, tom-tom

Algorithm 5 - brass, organ

Algorithm 6 - xylophone, tom-tom, organ, vibraphone, snare drum, base drum

Algorithm 7 - pipe organ

REGISTER OVERVIEW

The system is controlled by means of a large number of registers. General system registers are:

- timer values and status, software use
- LFO enable and frequency, to distort the FM channels
- DAC enable and amplitude
- output enables for each of the six FM channels
- number of frequencies to be used in FM channels 3 and 6.

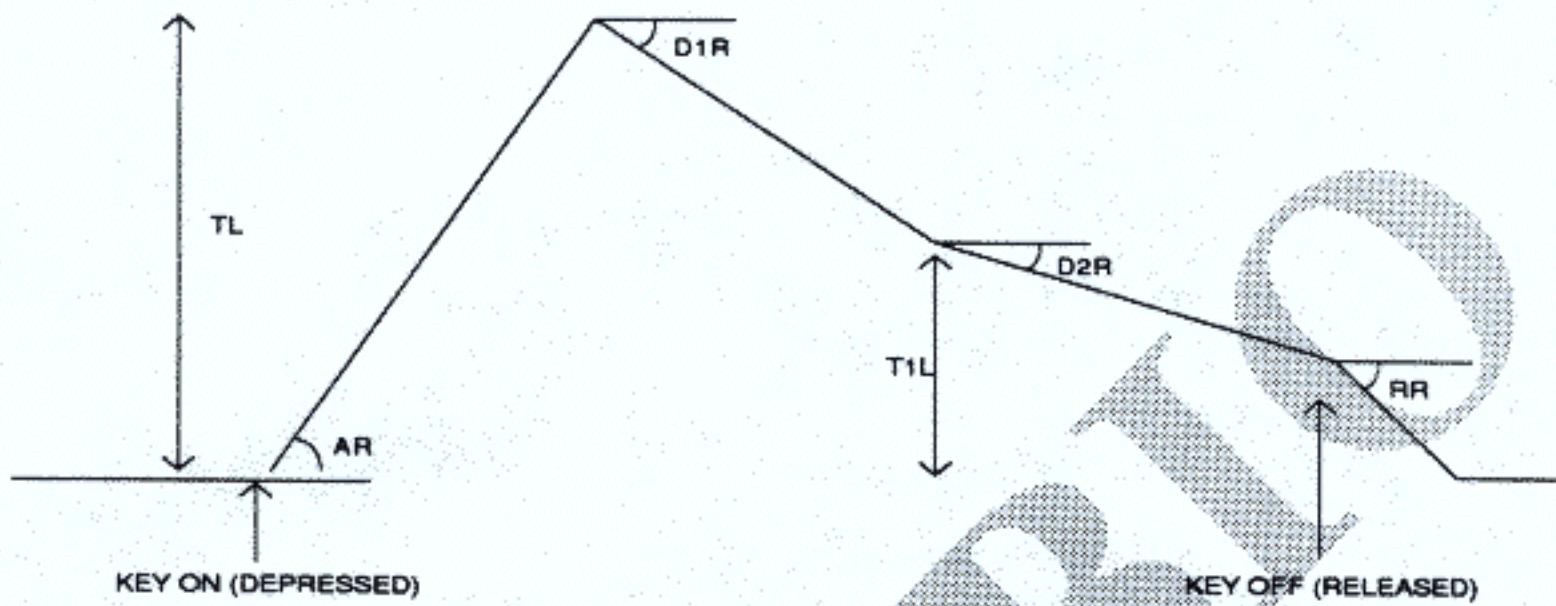
Usually, an FM channel has only one overall frequency, but if so elected, FM channels 3 and 6 use four separate frequencies, one for each operator.

The remainder of the registers apply to a single FM channel, or to an operator in that channel. Registers that refer to the channel as a whole are:

- frequency number (in the standard case)
- algorithm number
- extent of self-feedback in operator 1
- output type, to L, R, or both speakers. This can only be heard if headphones are used.
- the extent to which the channel is distorted by the LFO.

Registers that refer to each operator make up the remainder. The four operators' connections are determined by the algorithm used, but the envelope is always specified individually for each operator. In the case of FM channels 3 and 6, the frequency may be specified individually for each operator.

ENVELOPE SPECIFICATION



The sound starts when the key is depressed, a process called "key on." The sound has an attack, a strong primary decay, followed by a slow secondary decay. The sound continues this secondary decay until the key is released, a process called "key off." The sound then begins a rapid final decay, representing, for example, a piano note, after the key has been released and the damper has come down on the strings.

The envelope is represented by the above amplitudes and angles, and a few supplementary registers. Used in the above diagram are:

- TL** — Total level, the highest amplitude of waveform.
- AR** — Attack rate, the angle of initial amplitude increase. This can be made very steep if desired. The problem with slow attack rates is that if the notes are short, the release (called "key off") occurs before the note has reached a reasonable level.
- D1R** — The angle of initial amplitude decrease.
- T1L** — The amplitude at which the slower amplitude decrease starts.
- D2R** — The angle of secondary amplitude decrease. This will continue indefinitely unless "key off" occurs.
- RR** — The final angle of amplitude decrease, after "key off."

Additional registers are:

- RS** — Rate scaling, the degree to which envelopes become shorter as frequencies become higher. For example, high notes on a piano fade much more quickly than low notes.

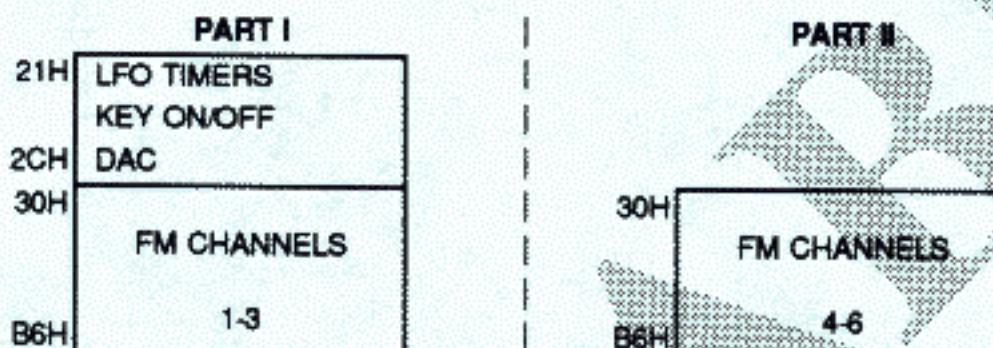
AM — Amplitude Modulation enable, whether or not this operator will allow itself to be modified by the LFO. Changing the amplitude of the slots changes the loudness of the note; changing the amplitude of the other operators changes its flavor.

SSG-EG — A proprietary register whose usage is unknown. It should be set to zero.

The FM-2612 may be accessed from either the 68000 or the Z-80. In both cases, however, the bus is only 8 bits wide.

The FM-2612 is accessed through memory locations 4000H - 4003H in the Z80 case, or A04000H - A04003H in the 68000 case. These will be referred to as 4000 to 4003.

The internal registers of the FM-2612 are divided as follows:

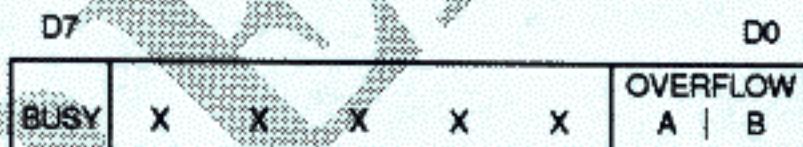


To write to Part I, write the 8-bit address to 4000 and the data to 4001. To write to Part II, write the 8-bit address to 4002 and the data to 4003.

Caution: Before writing, read from any address to determine if the YM-2612 I/O is still busy from the last write. Delay until Bit 7 returns to 0.

Caution: In the case of registers that are "ganged together" to form a longer number — for example the 10-bit Timer A value or the 14-bit frequencies — write the high register first.

READ DATA: Reading from any of the four locations.



BUSY — 1 if busy, 0 if ready for new data.

OVERFLOW — 1 if the timer has counted up and overflowed. See Register 27H.

PART I MEMORY MAP

22H	X	X	X	X	LFO EN	LFO FREQ
24H	TIMER A					
25H	X	X	X	X	X	TIMER A
26H	TIMER B					
27H	CH3	RESET	ENABLE	LOAD		
28H	MODE	B	A	B	A	B
28H	OPERATOR		X	CHANNEL		
2AH	DAC					
2BH	DAC EN	X	X	X	X	X

30H+	X	DT1	MUL	
40H+	X	TL		
50H+	RS	X	AR	
60H+	AM	X	X	
70H+	X	X	X	
80H+	D1L		RR	
90H+	X	X	X	
		X	SSG-EG	

30H	CH1, OP1
31H	CH2, OP1
32H	CH3, OP1
34H	CH1, OP2
34H	CH2, OP2
36H	CH3, OP2
38H	CH1, OP3
39H	CH2, OP3
3AH	CH3, OP3
3CH	CH1, OP4
3DH	CH2, OP4
3EH	CH3, OP4

Each of 30H-90H has twelve entries, three channels x four operators.

Channels 1-3 become channels 4-6 in Part II.

PART I MEMORY MAP (cont.)

A0H+	FREQ. NUM			
A4H+	X	X	BLOCK	FREQ. NUM
A8H+	CH 3 SUPPLEMENTARY FREQ. NUM			
AC _H +	X	X	CH 3 SUPP BLOCK	CH3 SUPP FREQ NUM
B0H+	X	X	FEEDBACK	ALGORITHM
B4H+	L	R	AMS	X
				FMS

Each of the above has three entries. All follow the pattern

A0H	CH1
A1H	CH2
A2H	CH3

with the exception that A8H and ACH follow the pattern

A8H	CH3, OP2
A9H	CH3, OP3
AAH	CH3, OP4

"PART II" is a duplication of 30H-B4H, where channels 1-3 are replaced by 4-6.

The Registers:

22H	X	X	X	X	LFO EN	LFO FREQ
-----	---	---	---	---	--------	----------

LFO EN — 1 is enabled, 0 disabled.

LFO FREQ

Hz	0	1	2	3	4	5	6	7
	3.98	5.56	6.02	6.37	6.88	9.63	48.1	72.2

The LFO (Low Frequency Oscillator) is used to distort the FM sounds' amplitude and phase. It is truly enabled, as there is:

- a global enable in Register 22H
- a sensitivity enable on a channel by channel basis, in Registers B4H-B6H
- an amplitude enable on an operator by operator basis in Registers 60H-6EH.

If the LFO is desired, enable it by Register 22H. Next, select which channels will be affected by the LFO, to what degree, and whether their amplitude or frequency is affected, by setting Registers B4H-B6H. Finally, if a channel's amplitude is affected, make sure that it is only the "slots" that are affected by setting Registers 60H-6EH.

24H

TIMER A MSBs					
--------------	--	--	--	--	--

25H

X	X	X	X	X	X	TIMER A LSBs
---	---	---	---	---	---	--------------

Registers 24H and 25H are ganged together to form 10-bit TIMER A, with Register 25H containing the least significant bits. They should be set in the order 24H, 25H. The timer lasts

18* (1024 - TIMER) microseconds

TIMER A = all 1's \rightarrow 18 μ s = 0.108 ms

TIMER A = all 0's \rightarrow 18,400 μ s = 18.4 ms

26H

TIMER B

8-bit TIMER B lasts

288 * (256 - TIMER B) microseconds

TIMER B = all 1's \rightarrow 0.288 ms

TIMER B = all 0's \rightarrow 73.44 ms

27H

CH3 MODE	RESET		ENABLE		LOAD	
	B	A	B	A	B	A

Register 27H controls the software timers and the Channel 3 (and 6) mode, two entirely separate items.

CH3 MODE	D7	D6	
NORMAL	0	0	Channel 3 is the same as the others.
SPECIAL	0	1	Channel 3 has four separate frequencies.
ILLEGAL	1	X	—

A normal channel's operators use offsets of integral multiples of a single frequency. In SPECIAL mode, each operator has an entirely separate frequency. Channel 3 operator 1's frequency is in Registers A2 and A6. Operators 2 and 4 are in Registers A8 and AC, A9 and AD, and AA and AE, respectively.

No one at Sega has used the timer feature, but the Japanese manual says:

LOAD 1 starts the timer, 0 stops it.

ENABLE 1 causes timer overflow to set the read register flag. 0 means the timer keeps cycling without setting the flag.

RESET writing a 1 clears the read register flag, writing a 0 has no effect.

28H

OPERATOR	X	CHANNEL
----------	---	---------

This register is used for "key on" and "key off." "Key on" is the depression of the synthesizer key. "Key off" is its release. The sequence of operations is: set parameters, key on, wait, key off. When key off occurs, the FM channel stops its slow decline and starts the rapid decline specified by "RR", the release rate.

In a single write to Register 28H, one sets the status of all operators for a single channel. Sega always sets them to the same value, on (1) or off (0). Using a special channel 3, I believe it is possible to have each operator be a separate note, so there is possible justification for turning them on and off separately.

OPERATOR			
4	3	2	1

X CHANNEL

D2	D1	D0	
0	0	0	Channel 1
0	0	1	
0	1	0	
1	0	0	Channel 4
1	0	1	
1	1	0	

2AH

DAC DATA

Register 2AH contains 8 bit DAC data.

2BH

DAC EN	X	X	X	X	X	X
--------	---	---	---	---	---	---

If the DAC enable is 1, the DAC data is output as a replacement for channel 6. The only channel 6 register that affects the DAC is the stereo output portion of Register B4H.

Registers 30H-90H are all single-operator registers. Please see page 8 for how the twelve channel-operator combinations are arranged.

30H+

X	DT1	MUL
---	-----	-----

Both DT1 (Detune) and MUL (multiple) relate the operator's frequency to the overall frequency.

MUL ranges from 0 to 15₁₀, and multiplies the overall frequency, with the exception that 0 results in multiplication by 1/2. That is, MUL = 0 to 15 gives x 1/2, x 1, x 2, ... x 15.

DT1 gives small variations from the overall frequency x MUL. The MSB of DT1 is a primitive sign bit, and the two LSBs are magnitude bits. See the next page for a diagram.

D6	D5	D4	MULTIPLICATIVE EFFECT
0	0	0	No change
0	0	1	$\times (1 + E)$
0	1	0	$\times (1 + 2E)$
0	1	1	$\times (1 + 3E)$
1	0	0	No change
1	0	1	$\times (1 - E)$
1	1	0	$\times (1 - 2E)$
1	1	1	$\times (1 - 3E)$

where E is a small number

40H+

X	TL
---	----

TL (total level) represents the envelope's highest amplitude, with 0 being the largest and 127₁₀ the smallest. A change of one unit is about 0.75 dB.

To make a note softer, only change the TL of the slots (the output operators). Changing the other operators will affect the flavor of the note.

50H+

RS	X	AR
----	---	----

Register 50H contains RS (rate scaling) and AR (attack rate). AR is the steepness of the initial amplitude rise, shown on page 4.

RS affects AR, D1R, D2R, and RR in the same way. RS is the degree to which the envelope becomes narrower as the frequency becomes high.

RS

AR

The frequency's top five bits (3 octave bits and 2 note bits) are called KC (key code) in the following rate formulas:

- RS=0 \Rightarrow Final Rate = 2 * Rate + (KC/8)
- RS=1 \Rightarrow Final Rate = 2 * Rate + (KC/4)
- RS=2 \Rightarrow Final Rate = 2 * Rate + (KC/2)
- RS=3 \Rightarrow Final Rate = 2 * Rate + KC**

** Always rounded down.

As rate ranges from 0-31, this means that the RS influence ranges from small (at 0-3) to very large (at 0-31).

60H+	AM	X X	D1R
------	----	-----	-----

D1R (first decay rate) is the initial step amplitude decay rate (see page 4). It is, like all rates, 0-31 in value and affected by RS.

AM is the amplitude modulation enable, whether or not this operator will be subject to amplitude modulation by the LFO. This bit is not relevant unless both the LFO is enabled and Register B4's AMS (amplitude modulation sensitivity) is non-zero.

70H+	X X X	D2R
------	-------	-----

D2R (secondary decay rate) is the long tailoff of the sound that continues as long as the key is depressed.

90H+	X X X X	SSG-EG
------	---------	--------

This register is proprietary and should be set to zero.

The final registers relate mostly to a single channel. Each register is tripled; please see the diagram on page 9.

A0H+	FREQ NUM		
A4H+	X X	BLOCK	FREQ NUM
A8H+	CH 3 SUPP FREQ NUM		
ACH+	X X	CH 3 SUPP BLOCK	CH 3 SUPP FREQ NUM

Channel 1's frequency is in A0 and A4H.
Channel 2's frequency is in A1 and A5H.
Channel 3, if it is in normal mode (please see page 12) is in A2 and A6H.

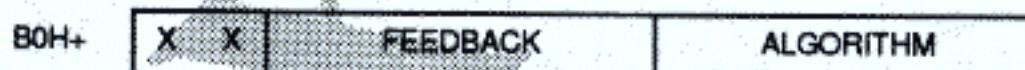
If channel 3 is in special mode:

Operator 1's frequency is in A2 and A6H
Operator 2's frequency is in A8 and ACH
Operator 3's frequency is in A9 and ADH
Operator 4's frequency is in AA and AEH.

The frequency is a 14-bit number that should be set high byte, low byte (e.g., A4H then A0H). The highest 3 bits, called the "block," give the octave. The next 10 bits give position in the octave, and a possible 12-tone sequence is:

Low	617
	653
	692
	733
	777 all in base 10
	823
	872
	924
	979
	1037
	1099
High	1164

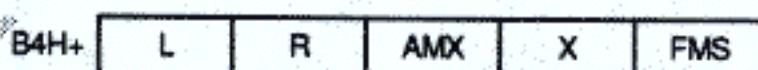
This sequence should be used inside each octave.



Feedback is the degree to which operator 1 feeds back into itself. In the voice library, self feedback is represented as this:



The ALGORITHM is the type of inter-operator connection used. Please see the list of the eight operators on page 3.



Register B4H contains stereo output control and LFO sensitivity control.

L — Left output, 1 is on, 0 is off

R — Right output, 1 is on 0 is off

Note: The stereo may only be heard by headphones.

AMS (amplitude modulation sensitivity) and FMS (frequency modulation sensitivity) are the degree to which the channel is affected by the LFO. If the LFO is disabled, this register need not be set. Additionally, amplitude modulation is also enabled on an operator-by-operator level.

AMS	0	1	2	3
dB	0	1.4	5.9	11.8

FMS	0	1	2	3	4	5	6	7
% of a halftone	0	± 3.4	± 6.7	± 10	± 14	± 20	± 40	± 80

TEST PROGRAM

Here a tested power-on initialization and sample note in the "Grand Piano" voice (page 27).

Register	Value	Comments
22H	0	LFO off
27H	0	Channel 3 mode normal
28H	0	Off
28H	1	Off
28H	2	Off
28H	4	Off
28H	5	Off
28H	6	Off
2BH	0	DAC off
30H	71H	
34H	0DH	DT1/MUL
38H	33H	
3CH	01H	
40H	23H	
44H	2DH	Total Level
48H	26H	
4CH	00H	
50H	5FH	
54H	99H	RS/AR
58H	5FH	
5CH	94H	
60H	5	
64H	5	AM/D1R
68H	5	
6CH	7	
70H	2	
74H	2	D2R
78H	2	
7CH	2	

Register	Value	Comments
80H	11H	D1L/RR
84H	11H	
88H	11H	
8CH	A6H	
90H	0	Proprietary
94H	0	
98H	0	
9CH	0	
B0H	32H	Feedback/Algorithm
B4H	C0H	Both speakers on
28H	00H	Key off
A4H	22H	Set frequency
A0H	69H	
28H	F0H	
<wait>		
28H	00H	Key on
		Key off
Notes:		
1. Write address then data.		
2. Loop until read register D7 becomes 0.		
3. Follow MSB/LSB sequence.		

Programmable Sound Generator (PSG)

The PSG contains four sound channels, consisting of three tone generators and a noise generator. Each of the four channels has an independent volume control (attenuator). The PSG is controlled through output port \$7F.

Tone Generator Frequency

The frequency (pitch) of a tone generator is set by a 10-bit value. This value is counted down until it reaches zero, at which time the tone output toggles and the 10-bit value is reloaded into the counter. Thus, higher 10-bit numbers produce lower frequencies.

To load a new frequency value into one of the tone generators, you write a pair of bytes to I/O location \$7F according to the following format:

First Byte:	1	R2	R1	R0	d3	d2	d1	d0
Second Byte:	0	0	d9	d8	d7	d6	d5	d4

The R2:R1:R0 field selects the tone channel as follows:

R2	R1	R0	Tone Channel
0	0	0	#1
0	1	0	#2
1	0	0	#3

10-bit data is: (msb) d9 d8 d7 d6 d5 d4 d3 d2 d1 d0 (lsb)

Noise Generator Control

The noise generator uses three control bits to select the "character" of the noise sound. A bit called "FB" (Feedback) produces periodic noises or "white" noise:

FB	Noise Type
0	Periodic (like low-frequency tone)
1	White (hiss)

The frequency of the noise is selected by two bits NF1:NF0 according to the following table:

NF1	NF0	Noise Generator Clock Source
0	0	Clock/2 (higher pitch, "less coarse")
0	1	Clock/4
1	0	Clock/8 (lower pitch, "more coarse")
1	1	Tone Generator #3

Note: "Clock" is fixed in frequency. It is a crystal controlled oscillator signal connected to the PSG.

When NF1:NF0 is 11, Tone Generator #3 supplies the noise clock source. This allows the noise to be "swept" in frequency. This effect might be used for a jet engine runup, for example.

To load these noise generator control bits, write the following byte to I/O port \$7F:

Out (\$7F):

1	1	1	0	0	FB	NF1	NF0
---	---	---	---	---	----	-----	-----

Attenuators

Four noise attenuators adjust the volume of the three tone generators and the noise channel. Four bits A3:A2:A1:A0 control the attenuation as follows:

A3	A2	A1	A0	Attenuation
0	0	0	0	0 db (maximum volume)
0	0	0	1	2 db Note: a higher attenuation results in a quieter sound
0	0	1	0	4 db
0	0	1	1	6 db
0	1	0	0	8 db
0	1	0	1	10 db
0	1	1	0	12 db
0	1	1	1	14 db
1	0	0	0	16 db
1	0	0	1	18 db
1	0	1	0	20 db
1	0	1	1	22 db
1	1	0	0	24 db
1	1	0	1	26 db
1	1	1	0	28 db
1	1	1	1	-Off-

The attenuators are set for the four channels by writing the following bytes to I/O location \$7F:

Tone Generator #1:

1	0	0	1	A3	A2	A1	A0
1	0	1	1	A3	A2	A1	A0
1	1	0	1	A3	A2	A1	A0
1	1	1	1	A3	A2	A1	A0

Tone Generator #2:

Tone Generator #3:

Noise Generator:

EXAMPLE

When the Mk3 is powered on, the following code is executed:

```
LD HL,CLRTB      ; clear table
LD C,PSG_PRT    ; psg port is $7F
LD B,4          ; load four bytes
OTIR
(etc.)
```

CLRTB defb \$9F, \$BF, \$DF, \$FF

This code turns the four sound channels off. It's a good idea to also execute this code when the PAUSE button is pressed, so that the sound does not stay on continuously for the pause interval.



SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #0

To: Developers and Third Parties
From: Mac Senour, Technical Support Specialist
Date: 3/13/91
Re: Genesis Technical Bulletins

The following bulletins were written to address certain areas of Genesis game development that required correction or clarification. Please replace all previous bulletins with the attached pages.

These Technical Bulletins may not, in whole or part, be copied photocopied, reproduced, or translated without prior written consent from Sega of America, Inc.

If you have any questions, comments or have a problem that you like to have addressed in the future, please contact:

Mac Senour, Technical Support Specialist
Sega of America
573 Forbes Blvd.
So. San Francisco, CA 94080
(415) 742-9300 ex352



SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #1

To: Developers and Third Parties

From: Mac Senour, Technical Support

Date: 3/13/91

Re: Errors within the Microtec examples

There are three errors in examples that are provided. Please make the changes as noted.

In TESTC68K.bat:

The asm68k commands end with a semicolon, remove it and the file will assemble correctly.

The link command is incorrect. It should be:
LNK68K -c sieve.cmd -o sieve sieve

The C compiler documentation refers to a command line option of 'STRINGSINTEXT' for allocating string in code segment rather than data segment. The correct spelling for this option is 'STRINTEXT'.

SEGA®

SEGA OF AMERICA, INC.
Consumer Products Division

TECHNICAL BULLETIN

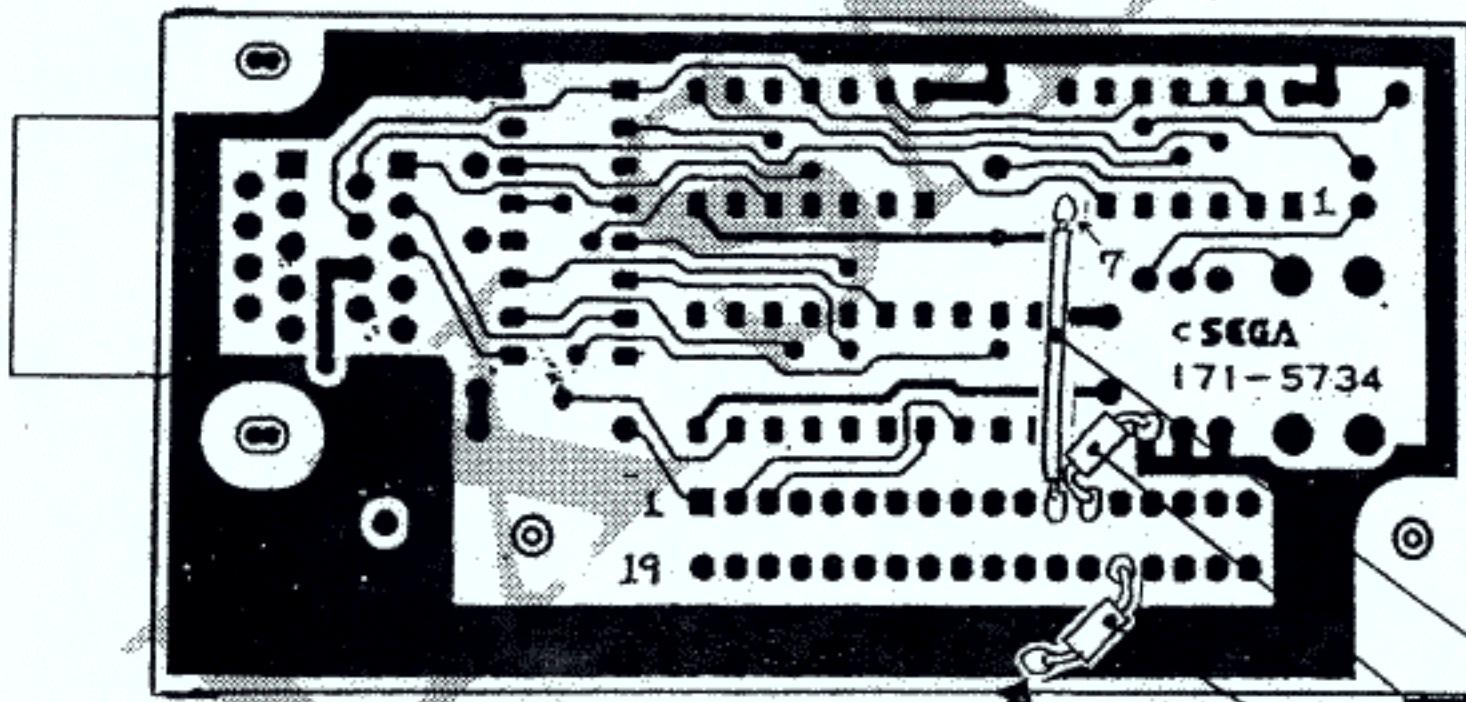
No. 002

To: Genesis Developers and Genesis Third Parties
From: Steve Hanawa *Steve*
Date: June 19, 1990
Re: Genesis Loader Board

In order to prevent loading errors, please have the following modification on Genesis Loader Board. You need to have:

2 Resister 4.7kΩ 1/8W
1 Jumper Wire

Please call me if you have any questions or special arrangements.



Remove the solder resister before soldering.

- 1) Place 4.7k resister between centronics connector pin #32 and +5V.
- 2) Place 4.7k resister between centronics connector pin #13 and +5V.
- 3) Place jumper wire between centronics connector pin #12 and ICI 74HC74 pin #7 (GND).



SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #3

To: Developers and Third Parties

From: Mac Senour, Technical Support *Mac*

Date: August 2, 1990

Re: Precaution when accessing the Z80 bus from a 68000 main (non-interrupt) routine.

If the following routine was executed and an interrupt occurred between step 2 and 3, a data error would occur IF the interrupt also access the Z80 bus.

- 1) Send Z80 bus request
- 2) Check acknowledge
- 3) Writing data to Z80 bus
- 4) Release Z80 bus

Essentially the main routine sends a bus request but before it can conduct it's data transfer the interrupt sends it's own bus request. When the interrupt completes, it releases the Z80 bus, as it should. Unfortunately the main routine expects the bus to be available, thus an error occurs. When this error occurs the Z80 RAM is corrupted. One symptom of this error is a sound is SOMETIMES interrupted.

The solution is really quite simple, before step one above, disable interrupts.

+ re-enable when done.

SEGA®

SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #4

To: Developers and Third Parties

From: Mac Senour, R&D/Technical Support *MSK*

Date: September 7, 1990

Re: Reading the controller pads

When reading a location within the range of \$A10000-\$A100FF, which includes the controller pads, the Z80 must have a bus request. If not the wait state will change from 250ns to 110ns. The shorter time will cause the Z80 to misread ALL further data. Once the ports have been read the Z80 may be released.

Included with this bulletin is a reprint of the routine that reads the control pads in the Logo Demo program. As you can see, this routine sends a bus request, reads the pads, then releases the Z80.

PRE

```

;----- ( TC )
;----- ( TA )
;----- ( ST )
;sw_data1+1 is the address for edge data of port_1

;----- sw:
        z80_diw                                ; set busreq
        bsr.b  rs_sub
        z80_ei
        rts

;----- rs_sub:
        lea      sw_data1,a5                  ;switch data store address
        lea      port_1,a6                  ;port for p1
        bsr      ??rs_0
        addq.w  #2,a6                  ;a6 = port_2
;

;----- ??rs_0:
        move.b  #$0,(a6)                  ;set TH = 0
        nop
        nop
        move.b  (a6),d7                  ;input data when TH = 0
        asl.b  #2,d7
        move.b  #$40,(a6)                  ;channe data when TH = 0 & wait
        andi.w  #11000000b,d7
        move.b  (a6),d6
        andi.w  #00111111b,d6
        or.b   d6,d7
        not.b  d7
        move.b  (a5),d6
        eor.b  d7,d6
        move.b  d7,(a5)+                  ;set TH = 1
        and.b   d7,d6
        move.b  d6,(a5)+                  ;channe data when TH = 0 & wait
        rts                                ;input data when TH = 1
;

;----- ;push -> 1
;----- ;copy
;----- ; switch data
;----- ;switch edge data

;----- ****< port initial >****;
;----- sw_init:
        moveq  #$40,d7                  ;port initial
        move.b  d7,cont_1
        move.b  d7,cont_2
        move.b  d7,cont_3
        rts                                ;set TH out

;----- ****<----->****;
;----- input  a6.1 = controller port address
;----- output d7.b == $00      Modem
;-----           $0d      Mega Drive Joy Stick
;-----           $0e      Ram Disk
;-----           $0f      ETC. or None
;----- use    d0,a0
;----- ****<----->****;
;----- cont      set      $0006

;----- port:
        movem.l d1-d2/a5,-(sp)          ; register push
        dis
        z80_diw
        lea      ??and_data(pc),a5      ; data table address set
        move.b  (a5),cont(a6)          ; Th bit set output
        moveq  #0,d7
        moveq  #508,d1  # 106          ; return register initial
        moveq  #508,d1
;

;----- CONFIDENTIAL
;----- PROPERTY OF SEGA
;----- , counter set

```

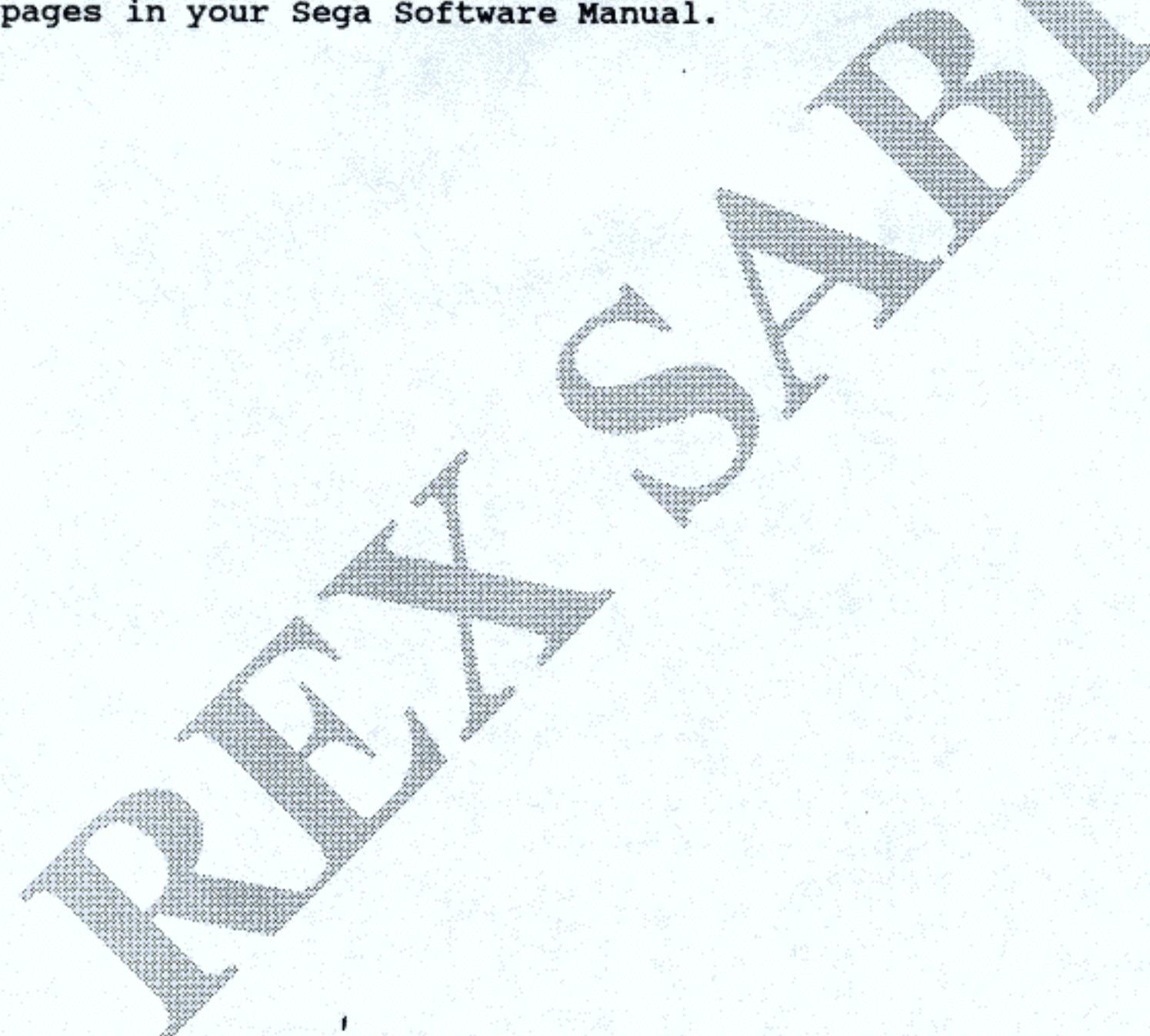


SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #5

To: Project Managers, Developers and Third Parties
From: Mac Senour, Technical Support Specialist
Date: November 26, 1990
Re: Cartridge identification

Every product must have, at location 100h, an IDTABLE. The table is slightly different for each type of developer. The difference is small, affecting the product code and the copyright, both important for product approval. Please replace the following pages in your Sega Software Manual.



```
*****  
*  
*   SEGA GENESIS CARTRIDGE ID TABLE  
*  
*   STANDARD FORMAT FOR SEGA OF AMERICA  
*  
*   NOV 26 1990 SEGA OF AMERICA  
*  
*****  
  
check_sum equ $0000  
ORG $0100  
-----  
dc.b 'SEGA GENESIS' ;100  
dc.b '(C)SEGA 199X.XXX' ;110 release year.month  
dc.b 'your game title' ;120 Japan title  
dc.b ' ' ;130  
dc.b ' ' ;140  
dc.b 'your game title' ;150 US title  
dc.b ' ' ;160  
dc.b ' ' ;170  
dc.b 'GM MK-XXXX -XX' ;180 product #, version  
dc.w check_sum ;18E check sum  
dc.b 'J' ;190 controller  
dc.l $00000000,$0007ffff,$00ff0000,$00ffff ;1a0  
dc.b ' ' ;1B0  
dc.b ' ' ;1C0  
dc.b ' ' ;1D0  
dc.b ' ' ;1E0  
dc.b 'U' ;1F0
```

```
;*****  
;*  
;* SEGA GENESIS CARTRIDGE ID TABLE  
;*  
;* STANDARD FORMAT FOR THIRD PARTIES  
;*  
;* NOV 26 1990 SEGA OF AMERICA  
;*  
;*****  
  
check sum equ $0000  
ORG $0100  
-----  
dc.b 'SEGA GENESIS' ;100  
dc.b '(C)T-XX 199X.XXX' ;110 release year.month  
dc.b 'your game title' ;120 title  
dc.b ' ' ;130  
dc.b ' ' ;140  
dc.b 'your game title' ;150 title  
dc.b ' ' ;160  
dc.b ' ' ;170  
dc.b 'GM T-XXXXXX XX' ;180 product #, version  
dc.w check_sum ;18E check sum  
dc.b 'J' ;190 controller  
dc.l $00000000,$0007ffff,$00ff0000,$00ffff ;1a0  
dc.b ' ' ;1B0  
dc.b ' ' ;1C0  
dc.b ' ' ;1D0  
dc.b ' ' ;1E0  
dc.b 'U' ;1F0
```



SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #6

To: Developers and Third Parties

From: Mac Senour, Technical Support 

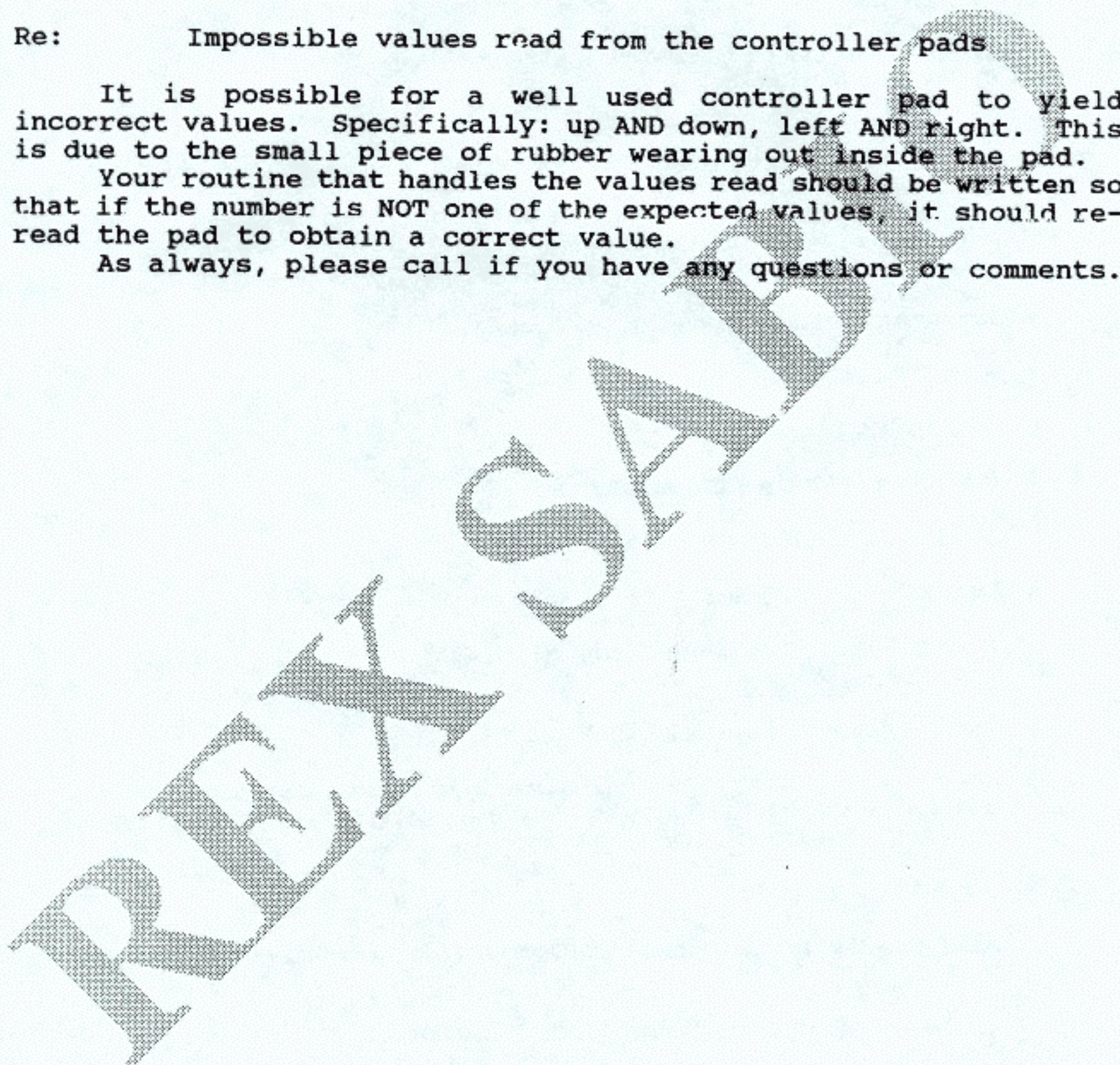
Date: 4/2/91

Re: Impossible values read from the controller pads

It is possible for a well used controller pad to yield incorrect values. Specifically: up AND down, left AND right. This is due to the small piece of rubber wearing out inside the pad.

Your routine that handles the values read should be written so that if the number is NOT one of the expected values, it should re-read the pad to obtain a correct value.

As always, please call if you have any questions or comments.





SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #11

To: Developers and Third Parties

From: Mac Senour, Technical Support *Mac*

Date: 9/9/91

Re: Problems during sound access

Sound output stops during a game.

Problem:

The busy flag was read in the FM sound generator YM2616 like this (address 4001h)

$(CS, RD, WR, A1, A0) = (0, 0, 1, 0, 1)$

However, in the case of the YM2612, it's output is not regulated according to the conditions set above. This results in the device being read as "not busy" and as a consequence, ends up outputting sound.

Fix:

When the FM sound generator's busy flag is read, do not access anything else other than:

$(A1, A0) = (0, 0)$ (address 4000h)



SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #12

To: Developers and Third Parties

From: Mac Senour, Technical Support *Mac*

Date: 9/9/91

Re: Problems with repeated resets

The software seems to go out of control when resets are repeated.

The Problem:

When the reset occurs the CPU is reset, however the VDP is not. When the reset occurs during a DMA the VDP continues the DMA. The VDP is accessed right after the reset. If this is done while the VDP is executing a DMA then this access is ineffective.

The Fix:

Before accessing the VDP after the initialization program (ICD_BLK4), check the DMA BUSY status register. If a DMA is being executed, do not attempt to access the VDP.

If the problem persists, ensure that you are not executing a DMA right after a reset.



SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #13

To: Developers and Third Parties

From: Mac Senour, Technical Support *Mac*

Date: 9/9/91

Re: Corrections to the Genesis Software Manual

When discussing VRAM, CRAM and VSRAM access, the manual states in pages 22-27 that byte access is possible. This is incorrect. Access is limited to word or long word.

On page 77 it implies that the 68000 may set the bank switches. The bank switches MUST be set by the Z80.

These changes affect version 1.0 of the manual, later versions will reflect this correction.



SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #14

To: Developers and Third Parties

From: Mac Senour, Technical Support *MS*

Date: 9/5/91

Re: ROM splitting

As we all know Genesis products must be split into 128k odd & even pieces. Sega expects the ROM images to be in the following format:

0:Even	1:Odd
2:Even	3:Odd
.	
.	
.	

For larger products, continue the above pattern. We would appreciate it if all products would conform to this method of splitting. Please request the utility to split ROMs, M2B or Split4, if your current tool can't output files in this manner.



SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #27

To: SEGA & Third Party Developers
From: Technical Support Group
Date: January 24, 1994
Re: Genesis Technical Information

The following tech note contains information on Genesis peripheral devices.

1. Warning regarding control pad data reads

For both the 3 and 6 button pads, the pad data is determined 2 μ sec after TH is modified. Therefore, as shown in the sample below, read data from the pad 2 μ sec (4 nop's) after TH is modified.

Joypad Reads

Pad data from the 3 and 6 button controllers are read 2 μ sec after TH is modified. The wait is necessary because the data in the chip needs time to stabilize after TH is modified. If data is read without this wait, there is no guarantee that the data will be correct.

Moreover, the 2 μ sec time is equivalent to 4 nop, including the 68000's prefetch.

```
Z80BusReq    equ    $A11100
Z80Reset     equ    $A11200
*****
; _JSID      joystick id get.
; This routine essentially performs connection check of the Genesis I/O
; ports. Figures out if some kind of controller is attached to I/O port and
; distinguishes between different classes (regular vs handshaking)
; controllers. This identification scheme is based on Addendum #1 in the
; Genesis Software Manual
; in:        d1.w  :  port number (0...2)
; out:       d0.l== ID
;           $0d  MD FP6B(six button) or MD 3button
;           $03  SOA/SOJ mouse
;           $07  SEGA TEAM PLAYER
;           $ff  input error(passed incorrect port number)
;           $0f  unknown or nothing connected
*****
; _JSID:
;     cmp.b    #3,d1
;     blt      _JSID_10
;     moveq   #-1,d0
;     rts
; _JSID_10:
;     movem.l d2-d3/a0,-(sp)
;     move.w  #$100,Z80BusReq    ;Z80 bus request
;     move.w  #$100,Z80Reset    ;Z80 reset line high
;     btst.b  #0,Z80BusReq    ;Z80 bus grant acknowledge
;     bne.s   *-8              ;wait until bus granted
;     move.l  #$00a10003,a0
```

```

moveq    #0,d0
move.b   d1,d0
add.l    d0,d0
adda.l   d0,a0
moveq    #0,d0
move.l   d0,d1
move.l   6(a0),d3      ;save original status of control port
swap     d3
move.b   (a0),d3      ;original status of I/O port
move.b   #%"01000000,6(a0) ;I/O control : TH (PD6) output
nop
nop
nop
nop
move.b   #%"01000000,(a0) ;set TH=1(high)
nop
nop
nop
nop
move.b   (a0),d0
move.b   #%"00000000,(a0) ;set TH=0(low)
nop
nop
nop
nop
move.b   (a0),d1
move.b   #%"01000000,(a0) ;set TH=1(high)
move.w   d0,d2
lsr.b    #1,d0
ror.w    #1,d0
lsr.b    #1,d0
rol.w    #1,d0
ror.w    #1,d2
lsr.b    #1,d2
rol.w    #1,d2
or.b     d2,d0      ;ID2=(PD6=1) & (PD1 OR PD0)
and.w    #$03,d0
lsl.b    #2,d0
move.w   d1,d2
lsr.b    #1,d1
ror.w    #1,d1
lsr.b    #1,d1
rol.w    #1,d1
ror.w    #1,d2
lsr.b    #1,d2

```

```

rol.w      #1,d2
or.b       d2,d1      ;ID0=(PD6=0) & (PD1 OR PD0)
and.w     #$03,d1
or.b       d1,d0      ;return id value
move.b    d3,(a0)    ;restore original status of I/O port
swap      d3
move.b    d3,6(a0)   ;restore original status of I/O control port
move.w    #0,Z80BusReq ;Z80 bus release
movem.l   (sp)+,d2-d3/a0
rts

```

2. Warning regarding Sega Mouse data requests

After the 10th data read, always issue an END DATA command. Set both TH & TR bit high to issue an END DATA command (see example below). Abnormal operation may occur if data beyond the 11th state is read.

```
*****
* MOUSE_GET      mouse/tablet driver
* THIS DRIVER READS THE SEGA MOUSE
* Access this routine from within the V_INT.
* The Z80 Enable/Disable code is provided as a safety check, to
* prevent unfavorable bus access.
*
* in   d1.w : port number ( 0..2 )
* out: d0.l & d2.l
*      d0.l contains connection status and reserved bits
*      d2.l contains MOUSE bits
*
*      d0.l = xxxx|xxxx|xxxx|xxxx|t1 m1 1 1|1 1 1 1|1 1 1 1
*      If mouse is connected, value returned in d0.l is as follows:
*      d0.l = xxxx|xxxx|xxxx|xxxx|1 0 1 1|1 1 1 1|1 1 1 1
*      if mouse connected, bit t1 =1 m1=0
*      if tablet connected, bit t1=0 m1 =1
*      note bits t1 and m1 cannot simultaneously be zero.
*
*      In case of error (nothing connected, other device, or incorrect port value)
*      the driver returns 0 in register d0.l. If mouse/tablet is actually
*      connected, the value in d0.l cannot be zero.
*
*      In case of timeout, the driver sets bit 31 in d2.l, and returns 0 in d0.l
*      MOUSE DATA is returned in register d2.l as follows:
*      d2.l = 0xxx|xxxx|YoXoYsXs|CMRL|X7X6X5X4|X3X2X1X0|Y7Y6Y5Y4|Y3Y2Y1Y0
```

```

*
* x denotes don't care bits
* Yo (Yover) 1 when Y data exceeds 255(ffff)
* Xo (Xover) 1 when X data exceeds 255(ffff)
* Ys (Ysign) 0 when Y data is positive, 1 when negative
* Xs (Xsign) 0 when X data is positive, 1 when negative
* C Center button (START/PAUSE)
* M Middle button (TRG-B)
* R Right button (TRG-C to CANCEL)
* L Left button (TRG-A to SELECT)
* X7-X0 absolute values of movement in X direction
* Y7-Y0 absolute values of movement in Y direction
* REGISTERS: register d1 is exclusively used for input to the routine.
* register d0 & d2 is used for output
*****

```

MOUSE_GET:

```

movem.l    d1/d3/d4/d7/a0,-(sp)
move.w    #$100,Z80BusReq
move.w    #$100,Z80Reset
btst.b    #0,Z80BusReq
bne      *-8
moveq    #0,d0          * error flag
cmp.w    #$0002,d1
bhi      MOUSE_EXIT
add.w    d1,d1
move.l    #$00a10003,a0

```

MOUSE_CONNECT:

```

move.b    #$60,6(a0,d1.w)
nop
nop
move.b    #$60,(a0,d1.w)      * TH,TR=11 (END DATA command)
moveq    #0,d2
moveq    #0,d3
mouse_rdy_lp:
btst.b    #4,(a0,d1.w)
beq.b    mouse_rdy_lp
move.b    (a0,d1.w),d4      * d4.b=? 1 1 1|0 0 0 0
and.b    #$0f,d4
tst.b    d4
bne      MOUSE_EXIT      * nothing connected/different controller
move.b    #$20,(a0,d1.w)      * select t1 m1 1 1
move.w    #$fe,d7          * timeout parameter
mouse_lp1:
btst.b    #4,(a0,d1.w)

```

bne.b	MOUSE_10	
dbra	d7,mouse_lp1	
bra	MOUSE_ERR	
MOUSE_10:		
move.b	(a0,d1.w),d0	* d0 = xxxx xxxx xxxx t1 m1 1 1
lsl.w	#8,d0	*d0 = xxxx t1 m1 1 1 0000 0000
move.b	#\$00,(a0,d1.w)	* select 1(rsv1) 1(rsv2) 1(rsv3) 1(rsv4)
nop		
mouse_lp2:		
btst.b	#4,(a0,d1.w)	
beq.b	MOUSE_20	
*	bne.b	MOUSE_20
	dbra	d7,mouse_lp2
	bra	MOUSE_ERR
MOUSE_20:		
move.b	(a0,d1.w),d3	* d3 = xxxx xxxx xxxx 1 1 1 1
move.b	#\$20,(a0,d1.w)	* select 1(rsv5) 1(rsv6) 1(rsv7) 1(rsv8)
lsl.w	#8,d3	* d3 = xxxx 1 1 1 1 0000 0000
mouse_lp3:		
btst.b	#4,(a0,d1.w)	
bne.b	MOUSE_30	
dbra	d7,mouse_lp3	
bra	MOUSE_ERR	
MOUSE_30:		
move.b	(a0,d1.w),d3	* d3 = xxxx 1 1 1 1 xxxx 1 1 1 1
lsl.b	#4,d3	
lsr.w	#4,d3	
move.b	#\$00,(a0,d1.w)	* select Yo Xo Ysgn Xsgn
or.w	d3,d0	* d0 = xxxx t1 m1 1 1 1 1 1 1 1 1 1 1
moveq	#0,d3	
mouse_lp4:		
btst.b	#4,(a0,d1.w)	
beq.b	MOUSE_40	
dbra	d7,mouse_lp4	
bra	MOUSE_ERR	
MOUSE_40:		
move.b	(a0,d1.w),d2	* d2 = xxxx xxxx xxxx Yo Xo Ys Xs
move.b	#\$20,(a0,d1.w)	* select Center Middle Right Left
lsl.w	#8,d2	*d2 = xxxx Yo Xo Ys Xs xxxx xxxx
mouse_lp5:		
btst.b	#4,(a0,d1.w)	

bne.b	MOUSE_50	
dbra	d7,mouse_lp5	
bra	MOUSE_ERR	
 MOUSE_50:		
move.b	(a0,d1.w),d2	* d2 = xxxx Yo Xo Ys Xs xxxx C M R L
move.b	#\$00,(a0,d1.w)	* select X high X7 X6 X5 X4
lsl.b	#4,d2	
lsl.w	#4,d2	* d2 = Yo Xo Ys Xs C M R L xxxx xxxx
 mouse_lp6:		
btst.b	#4,(a0,d1.w)	
beq.b	MOUSE_60	
dbra	d7,mouse_lp6	
bra	MOUSE_ERR	
 MOUSE_60:		
move.b	(a0,d1.w),d2	* d2 = Yo Xo Ys Xs C M R L xxxx X7 X6 X5 X4
move.b	#\$20,(a0,d1.w)	* select X low X3 X2 X1 X0
lsl.b	#4,d2	
lsl.l	#4,d2	* d2 = xxxx Yo Xo Ys C M R L X7 X6 X5 X4 xxxx xxxx
 mouse_lp7:		
btst.b	#4,(a0,d1.w)	
bne.b	MOUSE_70	
dbra	d7,mouse_lp7	
bra	MOUSE_ERR	
 MOUSE_70:		
move.b	(a0,d1.w),d2	* d2 = Yo Xo Ys Xs CMRL X7 X6 X5 X4 xxxx X3 X2 X1 X0
move.b	#\$00,(a0,d1.w)	* select Y High Y7 Y6 Y5 Y4
lsl.b	#4,d2	
lsl.l	#4,d2	
*d2=Yo Xo Ys Xs CMRL X7 X6 X5 X4 X3 X2 X1 X0 xxxx xxxx		
 mouse_lp8:		
btst.b	#4,(a0,d1.w)	
beq.b	MOUSE_80	
dbra	d7,mouse_lp8	
bra	MOUSE_ERR	
 MOUSE_80:		
move.b	(a0,d1.w),d2	
* d2 = Yo Xo Ys Xs CMRL X7 X6 X5 X4 X3 X2 X1 X0 xxxx Y7 Y6 Y5 Y4		
move.b	#\$20,(a0,d1.w)	* select Y Low Y3 Y2 Y1 Y0
lsl.b	#4,d2	
lsl.l	#4,d2	
*d2 = Yo Xo Ys Xs CMRL X7 X6 X5 X4 X3 X2 X1 X0 Y7 Y6 Y5 Y4 xxxx xxxx		

```
mouse_lp9:
    btst.b      #4,(a0,d1.w)
    beq.b       MOUSE_90
    dbra        d7,mouse_lp9
    bra         MOUSE_ERR

MOUSE_90:
    move.b      (a0,d1.w),d2
    *d2=YoXoYsXs|CMRL|X7X6X5X4|X3X2X1X0|Y7Y6Y5Y4|xxxx|Y3Y2Y1Y0
    move.b      #$60,(a0,d1.w)      * TH TR =11(END DATA command) and exit
    lsl.b       #4,d2
    lsr.l       #4,d2
    *d2= xxxx|xxxx|YoXoYsXs|CMRL|X7X6X5X4|X3X2X1X0|Y7Y6Y5Y4|Y3Y2Y1Y0

mouse_lp10:
    btst.b      #4,(a0,d1.w)
    beq.b       mouse_lp10
    or.l        #$00000000,d2

MOUSE_EXIT:
    move.w      #0,Z80BusReq
    movem.l    (sp)+,d1/d3/d4/d7/a0
    rts

MOUSE_ERR:
    move.b      #$60,(a0,d1.w)
    nop
    nop

mouse_erlp:
    btst.b      #4,(a0,d1.w)
    beq.b       mouse_erlp
    or.l        #$80000000,d2
    moveq      #0,d0
    move.w      #0,Z80BusReq
    movem.l    (sp)+,d1/d3/d4/d7/a0
    rts
```

3. Warning regarding the Sega infrared 6 button cordless pad

The following software issues were discovered prior to the release of the Mega Drive infrared 6 button cordless pad. Detailed technical specifications regarding the infrared pad will follow at a later date.

- **Change in the method to switch between the Fighting Pad 6B and the 3B Pad.**

The mode switch is made by the operation described below. Do not implement the following button combinations in game controls.

1. After installing the infrared 6 button pad receiver on the Mega Drive, turn on power to the Mega Drive.
2. When the receiver has power, push the **MODE**, **X** and **C** buttons on the cordless 6 button pad simultaneously.
3. The controller is now in 3 button mode.

4. Warning regarding Fighting Pad 6B (hereafter FP6B) data

The shaded data in the table below is used for expansion uses. The data is not currently used by the FP6B. The data there is invalid, therefore, please do not use them.

	TH(bit6)	TR(bit5)	TL(bit4)	R(bit3)	L(bit2)	D(bit1)	U(bit0)	Comments
1	Hi	TRG-C	TRG-B	RIGHT	LEFT	DOWN	UP	
2	Low	START	TRG-A	0	0	DOWN	UP	
3	Hi	TRG-C	TRG-B	RIGHT	LEFT	DOWN	UP	
4	Low	START	TRG-A	0	0	DOWN	UP	
5	Hi	TRG-C	TRG-B	RIGHT	LEFT	DOWN	UP	
6	Low	START	TRG-A	0	0	0	0	Recognition
7	Hi	TRG-C	TRG-B	MODE	TRG-X	TRG-Y	TRG-Z	Expansion
8	Low	START	TRG-A	1	1	1	1	" "
9	Hi	TRG-C	TRG-B	----	----	----	----	

Fighting Pad 6B Data Table

5. Cartridge Information (Corrections and Additions to Genesis Technical Bulletin #26, Sec. 3.)

The data "-yy" entered in cartridge information ID 18Bh~18Dh is the *version number* of the game. In some games however, the data suffix "-zz" as shown in numbers 2) and 5) below denotes the *sales territory* for the cartridge. Please make sure not to confuse the two.

Explanation

There are currently 5 product number types.

- 1) T-xxxxx ; third party brand <Japan>
- 2) T-xxxxx-zz ; third party brand <international>
- 3) G-tttt ; Sega brand <Japan>
- 4) MK-0000 ; Sega brand <US>
- 5) MK-0000-zz ; Sega brand <international other than US>

For European third party products, the product number is "T-xxxxx-50" (zz=50). Only "T-xxxxx" is necessary when the information is entered in the cartridge (-50 is unnecessary).

Example

If the product number is "T-12345-50" (European version), the cartridge information is as follows:

Incorrect

180h:"GM T-12345 -50xx" *Disk kind, good number, ver no.
checksum

Correct

180h:"GM T-12345 -00xx" *Disk kind, good number, ver no.
checksum

Note: The information contained here is not included in any previously released documentation.



SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #28

To: SEGA & Third Party Developers

From: Technical Support Group

Date: January 31, 1994

Re: Genesis Technical Information

-
1. In accordance with an increased number of Mega Drive peripheral devices, I/O port usage data is added at 190H of the cartridge ID.
For more information on I/O port usage data, please see Page 1 of the "GENESIS SOFTWARE DEVELOPMENT MANUAL COMPLEMENT."

Items added

- Sega mouse "M"
- Fighting pad 6B "6"
- Multi-tap "4"

Setting Conditions

- Fighting Pad 6B
Only when the XYZ buttons are effective. For the rest, enter "J" in the usual manner.
- Multi-tap
When multi-tap multi-switch is on.
(When data can be read with multi-switch turned on even if it is not geared for the multi-play.)

Multi-Tap Supplement

- Multi-tap switching setup has the following functions:
 - A~D : Terminal selectors (any one terminal can be selected)
 - MULTI: Used with games designed for more than 3 players
2. For the country information to be added at 1FOH of the Mega Drive cartridge ID, please input such data in accordance with A10001H of the I/O port.

BIT 7	0:	Domestic } BIT 6 0:	NTSC } BIT 7 1:	Overseas } BIT 6 0:	NTSC } BIT 7 1:	Overseas } BIT 6 1:	PAL }	Input as "J." Japan/Korea/Taiwan sales hardware Input as "U." US/Canada sales hardware Input as "E." Europe/Hong Kong sales hardware
-------	----	------------------------	--------------------	------------------------	--------------------	------------------------	-------	---

Note: For the above I/O port, see Page 72 of the "GENESIS SOFTWARE MANUAL."



SEGA OF AMERICA, INC. 275 Shoreline Drive, Redwood City, CA 94065 • (415) 508-2800

GENESIS TECHNICAL BULLETIN #29

To: SEGA & Third Party Developers
From: Technical Support Group
Date: March 10, 1994
Re: Genesis Technical Information

The information on the bottom of page 2 of the Genesis Sound Software Manual is incorrect. It should be changed to:

CONFIRMATION OF BUS STATUS:

This information is in \$A11100, bit 8. (not bit 0 as stated in the manual)

0 = Z80 is inactive, 68k can access
1 = Z80 is active

(Please note that this is reversed from what is in the manual.)



SEGA OF AMERICA, INC.
Consumer Products Division

GENESIS TECHNICAL BULLETIN #29A

To: SEGA & Third Party Developers

From: Technical Support Group

Date: April 11, 1994

Re: Genesis Technical Information

This is a clarification on Genesis Technical Bulletin #29.

The information on the bottom of page 2 of the Genesis Sound Software Manual is incorrect. It should be changed to:

CONFIRMATION OF BUS STATUS:

This information is in \$A11100, bit 8 when accessed by WORD and bit 0 by BYTE.

0 = Z80 is inactive, 68k can access

1 = Z80 is active

(Please note that this is reversed from what is in the manual.)



SEGA OF AMERICA, INC. Product Development • 150 Shoreline Drive, Redwood City, CA 94065

GENESIS TECHNICAL BULLETIN #31

To: **Sega and Third Party Developers**

From: **Developer Technical Support**

Date: **November 22, 1994**

Re: **Genesis ID table update**

1. Changes to cartridge data (ID) specifications

Please note that the specifications below also apply to the Genesis 32X.

The following changes have been made to the code entered in 1FOh.

- Old: Country code data
- New: 1 byte ASCII hardware enable code in 1FOh.
1F1h to 1FFh are filled with the "space" character code (20h).
- Explanation
The hardware enable code is used when the application reads the hardware data information in \$A10001 in order to perform territory lockout. The code is used by SEGA to check hardware and software compatibility.

\$A10001		Hardware Type	Main Sales Territories	Hardware Enable Code (numbers from 0 to F below)														
Bit 7	Bit 6			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0	0	Japan, NTSC	Japan, S. Korea, Taiwan	X	O	X	O	X	O	X	O	X	O	X	O	X	O	X
0	1	Japan, PAL		X	X	O	O	X	X	O	O	X	X	O	O	X	X	O
1	0	Overseas, NTSC	N. America, Brazil	X	X	X	X	O	O	O	O	X	X	X	X	O	O	O
1	1	Overseas, PAL	Europe, Hong Kong	X	X	X	X	X	X	X	X	O	O	O	O	O	O	O

↑ Example 2

↑ Example 1 ↑

o: Enable

x: Disable

Ex. 1) The hardware enable code "F" is entered in applications that do not limit hardware compatibility (i.e., a universal compatibility ROM). As can be seen in the table above, the hardware enable code "F" is compatible with all hardware types.

Ex. 2) For applications that are compatible only with the US version of the Genesis, the hardware enable code "4" is used. The application is compatible only with hardware that shows Bit 7 = 1 and Bit 6 = 0 at \$A10001.

Address	1F0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hardware Enable Code	4															
ASCII Code	34	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20

NOTE: Software containing errors in the code above will not be acceptable for final master ROM release purposes.



SEGA OF AMERICA, INC. Product Development • 150 Shoreline Drive, Redwood City, CA 94065

GENESIS TECHNICAL BULLETIN #32

To: Sega and Third Party Developers
From: Developer Technical Support and Catapult Entertainment
Date: 30-Jan-95
Re: Getting Games Ready For XBAND™

NOTE: This document is provided to Sega Certified Developers under the confidential terms of your Sega Developer Agreement.

1. Introduction

This document provides an introduction to the XBAND Video Game Modem and Network, and a preliminary overview of game design guidelines that will make it easier for your game to be adapted for network play. If you are developing a new multi-player game that you would like to be compatible with the XBAND Network, or if you would like the XBAND Network to be compatible with an existing multi-player game, this is the first document that you should read.

NOTE: Many of the technologies described in this document are proprietary to Catapult Entertainment, Inc. and are either patented or patent-pending.

2. How to reach XBAND

Catapult Entertainment, Inc.
c/o Developer Programs
20823 Stevens Creek Blvd.
Cupertino, CA 95014

Voice: 408.366.1735
Fax: 408.366.1729
Internet: developers@catapent.com

3. What is XBAND?

XBAND is made up of two major components: The XBAND Video Game Modem and the XBAND Network. The XBAND Video Game Modem is a device that

plugs in-between a Sega Genesis cartridge slot and a video game cartridge. The XBAND Network is an on-line system built into the XBAND Video Game Modem that automatically matches players to play video games with each other, keeps track of game statistics, provides e-mail and electronic newspapers, and much more.

As a video game developer you will be mostly concerned with the operation and programming of the XBAND Video Game Modem during the execution of your video game. The information you will need regarding this starts in Section 4.

3.1. The XBAND Video Game Modem

The XBAND Video Game Modem is more than just a modem. Physically, it plugs into a Sega Genesis cartridge slot and accepts a video game cartridge on a connector on the top. The phone line plugs into an RJ-11 jack on the side. There is also a slot to accept an ISO 7816-compliant Smart Card, whose use will be explained later, and a "pass-through" switch which allows you to play the attached cartridge in single-player mode.

The XBAND Video Game Modem has 512KBytes (4 MBits) of ROM that holds the XBAND OS (XOS) code, XOS graphics and sound data, and the XOS GameTalk routines (which can be directly called by game cartridges). The XBAND Video Game Modem also contains 64KBytes (512KBits) of battery-backed SRAM that holds the player's personal information (X-Mail, Player List, Player ID), XOS system patches, cartridge-specific game patches, game results from the last game played, and the XBAND on-line newspapers (BANDWIDTH and XBAND News).

The XBAND Video Game Modem contains a proprietary custom chip, which provides all of the logic functions for the operation of the XBAND Video Game Modem and also the means for controlling the execution of plugged-in cartridges.

Finally, the XBAND Video Game Modem contains a special modem chip that provides low-latency, noise-tolerant communications as well as special detection mechanisms for Call Waiting and other noise events. This allows the modem to run at only 2400 bits per second, transmitting a maximum of 4 bytes per frame.

3.2. The XBAND Network

The XBAND Network is different than traditional on-line systems. If a player of a traditional on-line service wishes to play a game with another person, the player must be connected to the service's Server for the duration of the game. With XBAND, players are only connected to the XBAND Server for a small percentage of their connect time, resulting in a lower cost and lower latency connection (typically less than 50 msec end-to-end).

When an XBAND player wishes to play against another person, the player's XBAND Video Game Modem connects to the XBAND Server. During the brief connection to the XBAND Server (typically between 10 seconds and a minute), the XBAND Video Game Modem uploads the player's phone number, any outgoing mail the player might have, and the game results from the last game played. Simultaneously, the XBAND Video Game Modem downloads the phone number of an opponent (or a number of minutes to wait for a call), any incoming mail, the two XBAND on-line newspapers, any new game patch that is needed (to be explained below), and any new system updates for the XOS.

Once the transaction is complete, the XBAND Video Game Modem hangs up the phone, and if it downloaded an opponent's phone number, it immediately dials the opponent. If it received instructions to wait for an opponent, it notifies the player of how long it expects to wait, and then waits for the telephone to ring.

Either by dialing out or by receiving an incoming call, the XBAND Video Game Modem connects with another XBAND Video Game Modem. There is a handshake where they exchange player information, then the video game cartridge is started up, the two games synchronize to each other, and the video game is played. Once the game is completed, the XBAND Video Game Modems disconnect, storing the results of the game in battery-backed up RAM, to be stored until one of the XBAND Video Game Modems connects to the XBAND Server again and uploads the data.

Since the phone lines used for XBAND are often used for voice calls or fax machines as well, it's important to gracefully deal with Call Waiting and line noise events. Such disruptions typically cause modems to lose carrier and hang up. The XBAND Video Game Modem, however, will pause the video game once disrupted, and then make every effort to wait out the noise event, restore carrier, and resume the game. Even if the modem has no choice but to hang up, it will redial the opponent and attempt to restore carrier and resume the game. This level of tenaciousness is essential when dealing with shared voice lines. We have found that about one in every three games experiences a line noise event that would have resulted in a lost connection with a conventional modem.

4. Getting Network Games to Run In Sync

The essential problem of real-time point-to-point modem game play is maintaining synchronization between the two video game machines. Although the video game hardware and cartridge software are identical on both ends of the phone line, slight variances between the systems as well as the limited timing resolution of the modem data stream creates a situation where it is quite easy for the two machines to drift out of synchronization. Once the two video games are out of sync even slightly, they will drift even further apart as game play continues, and in the end the game will likely have a different outcome.

There is no single technique that will magically maintain synchronization for all games. Catapult has found that there are about as many techniques for maintaining synchronization as there are techniques for designing main game loops. And, we have also come very quickly to the realization that it would be ludicrous to try to impose a single mechanism upon all video games developers. You need to look at the particular architecture of your game, see which issues apply in your case and which issues don't, and make use of the tools and techniques that Catapult has developed. And if that doesn't do the trick, we're here to help you. Many times we've been told that because of this and that a game will never run synchronized over a modem. We have yet to find one that we could not get to work with XBAND.

4.1. Why Synchronization Errors Occur

One would think that two identical video game machines running the same software and fed the same controller inputs would have identical execution. Unfortunately, this is not at all the case. There are several reasons why two identical video game machines running identical software will have varying execution behavior. This list is by no means complete, but it covers the major issues that we have encountered.

To begin with, any two free-running systems will run at different speeds. That is to say, the crystal oscillator that governs the clock speed of the video game processor and the timing of the video scanning is only accurate to a certain precision. In practical terms, we have found that the worst case speed differential between two video game systems will cause one machine to get a full frame ahead of the other in about two minutes. So, if your game uses Vertical Blanking Interrupt (VBI) as a timing reference, you will find that given two free-running video game systems, you can only hope to keep both systems on the same frame for about two minutes.

[Fortunately, since VBI synchronization is essential for many games, Catapult has developed a technology (called Sync-O-Tron™) which keeps VBIs locked together for the duration of the game, despite the drift between the two crystal oscillators. This technology usually helps solve game synchronization problems, but it is not always a complete solution. It is discussed later in Section 5.6.]

Another common source of synchronization problems occurs in the critical relationship between VBI and the main game loop. One would think that given identical hardware, identical software, and identical state (e.g. the same controller input), the VBI interrupt for a given frame will occur after the exact same instruction in the main game loop in both machines. If it were the case that there were no other asynchronous events occurring in the video game machine, this indeed would be true. Unfortunately, events such as DRAM refresh and sound DMA are unpredictable and asynchronous to the CPU execution causing the two CPUs to have the same average execution speed, although not the same cycle-by-cycle execution profile. So, while it is the case that VBI will occur at

approximately the same instruction in each main game loop, on a given VBI it might be one or two instructions off. If there is a game state decision that occurs in one of these instructions that is affected by a variable changed during VBI, the game execution will be different on both machines.

This same slight variation in CPU timing can also affect the number of VBI frames per game loop. For example, if the main game loop takes almost exactly three frames to execute, it might finish executing just before VBI on one machine and just after VBI on the other machine. If the number of frames per game loop affects game logic (as it often does), one game will count three frames and the other will count four, causing the two games will exhibit different behavior.

The slight variation in CPU timing can also affect reads from hardware registers that affect game behavior. For example, some games read the current HV counter at various points in the game loop for use as a random number. Clearly, given the execution variances just mentioned, the number fetched could have widely differing values on the two machines.

DMA timing can also introduce variations in execution for many of the same reasons. On the Sega Genesis, when graphics DMA occurs, it shuts down the CPU completely. Normally, DMA is only used during vertical blanking, allowing the CPU to run during active video, but sometimes when games are doing major screen updates (e.g. between segments of the game), they activate DMA continuously until the update is complete. If the DMA continues for more than a frame time, an entire VBI may be missed. And if the DMA completes at approximately the same moment as VBI, it may complete before VBI on one machine, but after VBI on the other machine. This can create a situation where the two machines will count a different number of VBIs resulting in differing game execution.

The game cartridge itself can introduce variances. Many games today have battery-backed cartridge RAM that they use for storing results from previous games. If these results affect game play (as is often the case with sports games), there will be varying execution between the two cartridges. It isn't a reasonable solution to just erase the RAM or update both with the same values. Very often the information stored in the RAM is of great importance to the game player (e.g. hard-earned stats, etc.). So, the game must have a mode of execution where the RAM status is disregarded.

In some situations there is more than one release of the same game cartridge. For example, NBA® JAM™ for Sega Genesis had two releases, and there were significant differences in the code (e.g. one of the basketball players was changed, and some of the timings were changed) between the two releases. A separate Game Patch had to be developed for each version to reconcile the execution differences so that a Rev. 1 could play against a Rev. 2 and stay in sync.

Finally, there is occasionally code that samples player reaction time so precisely that the exact same timing cannot be reproduced on both machines. For example,

a game might enter a spin loop to count the length of time before a player presses start for the purpose of getting a random number seed. The timing resolution of a CPU spin loop is far more precise than the time resolution of a modem, so the two game machines will certainly measure the player reaction time differently, resulting in a different random number seed.

4.2. How XBAND Synchronizes Existing Game Cartridges

Multi-player Sega Genesis video games played across XBAND must be carefully designed to 1) accept controller input from the remote Sega Genesis system, 2) maintain synchronized execution between the Sega Genesis systems, and 3) have appropriate options for networked play. Clearly, games which are designed from the start with the knowledge of the XBAND Network and Video Game Modem can design in these capabilities. However, cartridges which are already out on the market require these capabilities to be added in the field. The XBAND Network and Video Game Modem work together to make these in-field extensions possible.

The process goes as follows: When someone hits the "Challenge" button to register for a match, the XOS running on the XBAND Video Game Modem dials into the XBAND Server and uploads the checksum of the cartridge that the player wants to play. The XBAND Server checks to see whether the cartridge is a supported XBAND game, and if it is, it downloads to the XBAND Video Game Modem a "Game Patch" which contains the code necessary to augment the video game cartridge to make it function properly over the modem. When the XBAND Video Game Modem connects to an opponent and the handshake is complete, control is passed to the video game cartridge. The video game cartridge executes as it would normally, except for the changes to the cartridge provided by the Game Patch. These changes route the controller reads through the modem to the remote Sega Genesis machine, resolve synchronization issues between the two systems, and record results of the game execution for later reporting to the XBAND Server. When game completes, the XBAND Video Game Modem hangs up the phone, and control returns to the XOS.

The custom chip in the XBAND Video Game Modem provides a very flexible set of capabilities for augmenting the execution of a game cartridge. The chip permits you to make any change to a cartridge ROM that is needed to make it work on XBAND.

There are a number of ways that a Game Patch is developed for an existing cartridge. Ideally, it is developed by the programmer who wrote the original game cartridge, but often this person is not available. In fact, sometimes the original source code for the game is not even available or is commented in a foreign language, rendering it unreadable. Catapult has found that given the relatively small changes required to augment a game for use on XBAND, it is quite feasible to get the game to work from the object code using reverse-

engineering techniques, and several of the Game Patches on XBAND were developed using these methods.

The following sections cover the basics of game synchronization, and give several examples. If you are designing a new game, we hope that this will cover enough of the "gotchas" to keep it "sync clean". If you are developing a Game Patch for a game that is already in the field, this should give you a starting point in understanding key places where the game must be patched.

4.3. Synchronization Approaches

There are two main communication approaches that are used between video games to maintain synchronization: Game State Communication and Controller Communication.

With the Game State approach, game state information (such as player position, number of enemies, number of lives, etc.) is transmitted between the game machines. This approach is commonly used in point-of-view games. The advantage of this approach is that the games truly free run on each machine and data is only sent when there is relevant data to send. For example, if the two players are in different rooms, only information that might affect the other player is transmitted. The disadvantage of this approach is that a great deal of information must sometimes be transmitted in real-time (e.g. if both players are shooting at each other). If you are using this approach for your game, then most of the discussion that follows does not apply to you since you are just trying to achieve identical game state (not identical execution) on both machines. Synchronization issues for Game State approach games are quite game-specific and are not addressed in this document. Please contact Catapult for more information.

With the Controller Communication approach, player game controller input is transmitted between the game machines. This approach is commonly used by games where all players are visible on the screen at the same time such as sports and fighting games. The advantage of this approach is that there is a relatively small, fixed amount of data that must be transmitted, allowing the data to be transmitted every frame time, thereby allowing real-time response. The disadvantage of this approach is that it requires identical real-time game logic execution on both game machines. The following discussions review the issues surrounding the synchronization of Controller Communication approach games.

5. Game Design Guidelines

This section provides a number of guidelines for game design that either avoid synchronization problems or suggest architectures in which the synchronization problems can be controlled.

5.1. Random Number Generators

DO:

- Implement a single random number generator routine in your code that starts from a settable seed and follows a reproducible sequence (mathematical algorithms work best). The random number generator should only be called by a single thread of execution (i.e. either the main game loop or the VBI routine, but not both). One of the game machines can determine the initial seed by any means you wish, and then it will transmit to the other game machine.

DON'T:

- Don't scatter random number functions throughout your code. Call to a central routine!
- Don't implement an irreproducible random number sequence. For example, reading the HV Counter or using whatever register D0's value happens to be upon entry to the VBI routine are both no-no's.
- Don't call the random number generator from more than one thread of execution (e.g. from both the main game loop and the VBI routine). VBI will occur at an unpredictable point in the main game loop. If the random number generator is being called (or the seed is being updated) in both threads of execution, the sequence in which it is being called will be unpredictable.

5.2. Battery-Backed Cartridge RAM Usage

DO:

- Provide a mode whereby the current contents of the battery-backed cartridge RAM on your cart are not utilized by the game logic. Careful! Make sure that in this mode none of the game logic considers the value of the RAM.

DON'T:

- Don't leave stray references to the battery-backed RAM in your code for modes that supposedly don't consider the RAM (e.g. Exhibition mode in sports games). Such stray references result in subtle synchronization bugs that only come up when one player has a particular value in the RAM and the other player does not. They are a hassle to track down.

5.3. Sega Genesis DMA Usage

DO:

- Utilize DMA as you normally would in a game, but make sure the DMA is complete and you have returned to the main game loop when the next VBI occurs.

DON'T:

- Don't have giant DMA operations that extend through one or more VBIs. This may result in a different number of VBIs occurring in each game machine.
- If you decide how many DMA operations you do during a VBI by checking whether vertical blanking is over or not, don't use the number of completed DMA operations in your game logic. For example, you are in the VBI routine, you are doing a number of DMA operations, and before each DMA operation you check to see whether you've run out of vertical blanking time. Because of variances in execution speed, one game machine may report that vertical blanking is over, and the other one may report the opposite. If the main game loop makes game logic decisions based on the number of completed DMA operations, the games will get out of sync.

5.4. Controller Reads and the Main Game Loop

DO:

- Ideally, read the controllers only once per game loop, before the start of the game loop. It doesn't matter whether you read the controllers during the VBI routine or in the main game loop (although Sega encourages you to read the six-button controllers and the multi-play adapter during VBI).
- Often, a game loop is so long (e.g. 3 frame times or more) that the controllers must be read more frequently than once per game loop to provide adequate responsiveness. There are two approaches to handling this situation:

(a) sample the controllers each VBI, but create a data structure that reflects the controller behavior through the duration of the game loop (e.g. Up on 2nd frame, A and Right on 3rd frame), then feed this data structure, all at once, into the next game loop. This data structure, rather than raw controller data, will be transmitted to the other game machine, once per game loop. This is the preferred method.

(b) sample the controllers each VBI, but only feed in the updated controller information at predictable points in the main game loop. For example, if you have 16 objects on the screen that are updated in sequence in the main game loop, and it typically takes 8 frame times to update them all, feed in each VBIs new controller value after every two objects are updated. Take

care to design your code such that the game loop is not usually held up waiting for VBI. There are a number of other subtleties that must be considered since we can't be certain that both game machines will have the same number of VBIs per game loop. Overall, this is a doable, but tricky, solution to the problem.

DON'T:

- Don't sample the controllers asynchronously to the main game loop and feed in their values at unpredictable points. It is critical that both machines utilize the same controller data for the same game logic. Since VBI will occur at unpredictable points in the game loop, controller values that are sampled at VBI and immediately fed into the game loop may result in differing game logic between the two game machines.

5.5. Game Loop Duration and Game Logic

Game loops typically are longer than one VBI in duration. Unfortunately, it is often very difficult to predict how many VBIs of duration a particular game loop will take, and in fact, for the execution reasons mentioned above, it is quite possible that the two game machines may take a different number of VBIs to execute the same game loop code.

Also, there are situations when the game logic execution is the same on both machines, but the display code execution is vastly different. Point-of-view games, such as racing games, that are split screen when played on a single video game machine, can be full-screen on each game machine when played on XBAND. The game logic is the same, since the state of both players is updated on both machines, but the display code for each player's point-of-view could be vastly different. Consequently, the execution time per game loop on each machine can be different.

If you can predict the number of VBIs that will elapse in your game loop easily (i.e. both machines can determine in advance the worst case number of frames per a given game loop), that simplifies the synchronization problem significantly. But if you can't (don't feel bad, most games can't), and you need to consider the number of elapsed VBIs in your game logic (e.g. for animation speed regulation), Catapult has developed a synchronization technology that will maintain sync between the two machines, provided you stay within the DOs and DON'Ts guidelines.

DO:

- Ideally, create a game loop with a predictable number of VBIs. For example, if you have a list of routines you need to go through in your game loop, and you have a worst case execution time for each routine, you might be able to determine how many VBIs will pass in the game loop. If you happen to complete

the game loop sooner than worst-case, you would wait it out until the predicted number of VBIs has elapsed.

- If you don't have a predictable number of VBIs per game loop, and the number of elapsed VBIs is used as part of the game logic (e.g. for animation speed regulation), then make sure that you use a single variable that contains the elapsed VBIs of the *previous* game loop. This variable will be controlled by a synchronization mechanism available from Catapult to maintain synchronization between the two game machines.

DON'T:

- Don't use the number of VBIs elapsed in the *current* game loop in your game logic. If you need to use elapsed VBIs for speed regulation, use the elapsed VBI count from the *previous* game loop, and make sure you use a single variable to store the value.

5.6. Using Sync-O-Tron™

Sync-O-Tron is a technology developed by Catapult that maintains synchronization between the VBIs of two video game systems. The technology is available for your use by your game while it is running on the XBAND Network. It provides two key advantages.

First, Sync-O-Tron often simplifies the development of synchronized game loops. Without Sync-O-Tron two free-running game machines can drift apart by an entire frame within 2 minutes. With Sync-O-Tron you are guaranteed that both game machines will always be frame synchronous to each other.

Second, Sync-O-Tron reduces communications latency. Without Sync-O-Tron there might be as much as a full frame of skew between the two game machines. To allow for this worst-case scenario it is necessary to add an additional frame of latency to the communications. With Sync-O-Tron there is always a fixed relationship between the VBIs of the two systems and minimum latency can be achieved.

Sync-O-Tron requires a very small routine that runs in VBI once per game loop.

6. User Interface Considerations

When a multi-player game is played over a telephone line by people who don't know each other (and can't talk to each other), there are a number of user interface issues that need to be considered. Although it is possible to get existing multi-player games to run "as is" on XBAND, we have found certain user interface changes are essential to avoid disputes between players and dissatisfaction with game play. These issues are discussed below:

6.1. Fair Access to Critical Controls

- Have a means for either player to gain access to critical controls without the other player interfering. Don't create a situation where one player who is faster at pushing buttons can prevent another player from accessing an important control screen.
- Have a reasonable time-out for every screen so that players don't tie up the game indefinitely. Don't create a situation where both players are required to push a button for the game to continue. One player may deliberately try to delay the game just to be obnoxious. Also, sometimes players are out of the room when the game starts up (which means they forfeit). It is fair that one player should be able to play the game from beginning to end if the other player is not there.

6.2. Tournament Mode

- Provide a tournament mode which has locked game play options (e.g. fixed game duration, secret power-up codes disabled, battery-backed cartridge RAM disabled, etc.). Don't leave it up to the players to amicably agree on settings for modem play. Decide on what is an appropriate tournament configuration, and lock it down.
- Make sure any options that should be available during a tournament should be settable by team. Don't provide options that affect both teams in tournament mode. The players will literally go back-and-forth, fighting over the option screen. If options can't be settable by team, then lock it down to a fixed setting.

6.3. Non-gameplay Features during Gameplay

- Provide reasonable limits to non-gameplay features that may delay the game (or annoy the other player). For example, if you provide an instant replay feature, you might want to limit each player to one instant replay per quarter. Don't expect the players to discipline themselves in the use non-gameplay features. When they are losing, some players will do anything to annoy the winning players.

6.4. Call Waiting and Line Noise Dialog Boxes

- Provide a means to display text during gameplay to alert the player about Call Waiting or Line Noise. The text will only be displayed when the game is frozen, so if characters need to be moved off the screen during the display of text, they can be moved back after the text has been removed from the screen. Don't rely on players to guess what is going on when the screen freezes during a Line Noise event. Also, if you cannot display text during Call Waiting, then the game must be terminated when the Call Waiting event occurs. When the player is done with the phone call, the game must start again from the beginning.

6.5. 4-Player Controller Allocation

- Make sure that during tournament play the controllers are allocated as controllers 1 and 2 against 3 and 4. Although it is interesting to split players in different configurations, it is very confusing, and usually results in players not knowing what is going on. Don't assume that the players can talk to each other to decide on a controller configuration.
- Make sure that all 4 players have a chance to join in the beginning, but there should be no additions after the game begins. We have found that players get very upset when a person joins in later in the match. Don't let players join in at any arbitrary time. Limit it to a single screen and make it very clear how many players are playing.
- Ideally, you should have a tournament mode for 1-on-1 and 2-on-(1 or 2). XBAND will eventually provide special matching for people who only want to play 1-on-1.

7. Special XBAND Features for New Games

The following features can significantly add to the game play value of your games, utilizing the special capabilities of the XBAND systems. These are just a few ideas. If you have special needs or a really cool idea, let us know, and we'll see if we can code it into XBAND.

7.1. Full-screen for Point-of-View 2-Player Games

If you are currently splitting the screen in your 2-player point-of-view game, you might consider a full-screen mode when playing through XBAND.

7.2. Updated Stats Throughout the Season

If you are writing a sports game with real players, you can have the player roster updated with the latest trades, injuries, etc. whenever the player logs into the XBAND Server. Catapult is working on content relationships to provide you with this data for your sports games.

7.3. New Levels, Mazes

The XBAND Server can download new levels and mazes for RPG or DOOM-like games, within available RAM constraints.

7.4. Eight-player Games

XBAND can host 4-against-4 games for teams of players using multi-play adapter in each game machine.

7.5. Multi-Way Gaming

In the future XBAND will be able to support multi-way gaming for 10s or even 100s of simultaneous players, competing against each other in the same game.

Contact Catapult for more details if you have a game that would take advantage of this functionality.

7.6. Handicapping

Your game patch can provide information for the XBAND Network to use to handicap players. For example, you could start out a better player in a fighting game with a pre-existing "damage" level, or assign a higher point level to blows he/she receives.

7.7. Bug Fixes

Your Game Patch can include bug fixes to your game. Whether the bugs are crashers or just imperfections that should be fixed, it is trivial to use the Game Patching mechanism to fix the bugs when the cartridge is played on XBAND. Indeed, most of the games running on the XBAND network today have been patched with bug fixes.

7.8. Your Idea Here

If you've got a unique idea for information to store on the XBAND Server and integrate with your XBAND savvy game, we're interested in hearing it. Contact Developer Relations at (408) 366-1735.

XBAND, BANDWIDTH and Sync-O-Tron are trademarks of Catapult Entertainment, Inc. All other trademarks are of their respective owners.

Addendum 1- Abstract from Page 2

Joysticks/Peripherals

It is possible to connect various peripherals to the I/O port other than joysticks. Each peripheral has its own ID code and it is possible for the CPU to check the code to identify what is connected. Be aware that there are some Master System peripherals which cannot be identified by this method.

S1 ID Code

The ID code for the external ports CTRL1, CTRL2, and EXT. which correspond to the I/O port's DATA1, 2, 3 (PD3, PD2, PD1, and PDO) are shown by a logical "or" operation. In concrete terms, setting the TH pin to output mode and outputting either 1 or 0 as data will yield the following:

$$\begin{aligned} ID3 &= (PD6 = 1) \text{ AND } (PD3 \text{ OR } PD2) \\ ID2 &= (PD6 = 1) \text{ AND } (PD1 \text{ OR } PDO) \\ ID1 &= (PD6 = 0) \text{ AND } (PD3 \text{ OR } PD2) \\ ID0 &= (PD6 = 0) \text{ AND } (PD1 \text{ OR } PDO) \end{aligned}$$

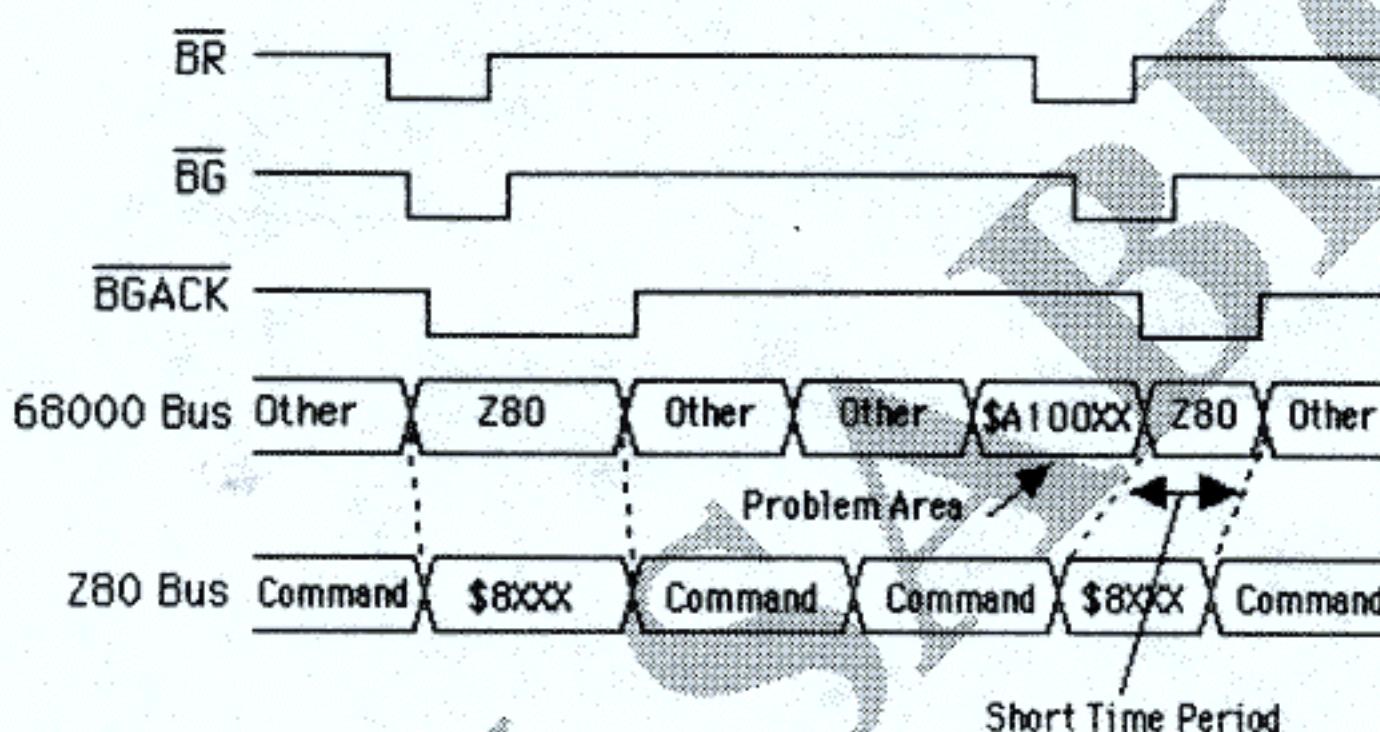
The relationship between the ID code and the peripheral hardware is shown below.

Device Name	ID3	ID2	ID1	ID0	
Old style joystick (2 trig):	1	1	1	1	• (\$F) if no joystick
Undefined:	1	1	1	0	(\$E)
New style joystick (3 trig):	1	1	0	1	(\$D) + power stick
Sega Reserved:	1	1	0	0	(\$C)
Undefined:	1	0	1	1	(\$B)
Sega Reserved:	1	0	1	0	(\$A)
Undefined:	1	0	0	1	(\$9)
Undefined:	1	0	0	0	(\$8)
Undefined:	0	1	1	1	(\$7)
Undefined:	0	1	1	0	(\$6)
Sega Reserved:	0	1	0	1	(\$5)
Undefined:	0	1	0	0	(\$4)
Undefined:	0	0	1	1	(\$3)
Undefined:	0	0	1	0	(\$2)
Undefined:	0	0	0	1	(\$1)
Sega Reserved:	0	0	0	0	(\$0)

Addendum 3- Abstract From Page 1

1) When the Z80 Cannot Do a 68000 Bus Access

There are times when the Z80 cannot read or write good data in the 68000's addresses. If the 68000's bus cycle accesses \$A100XX prior to the Z80's interrupt, the bus cycle for the Z80 interrupt is shortened. When this occurs, the Z80 cannot read or write good data.



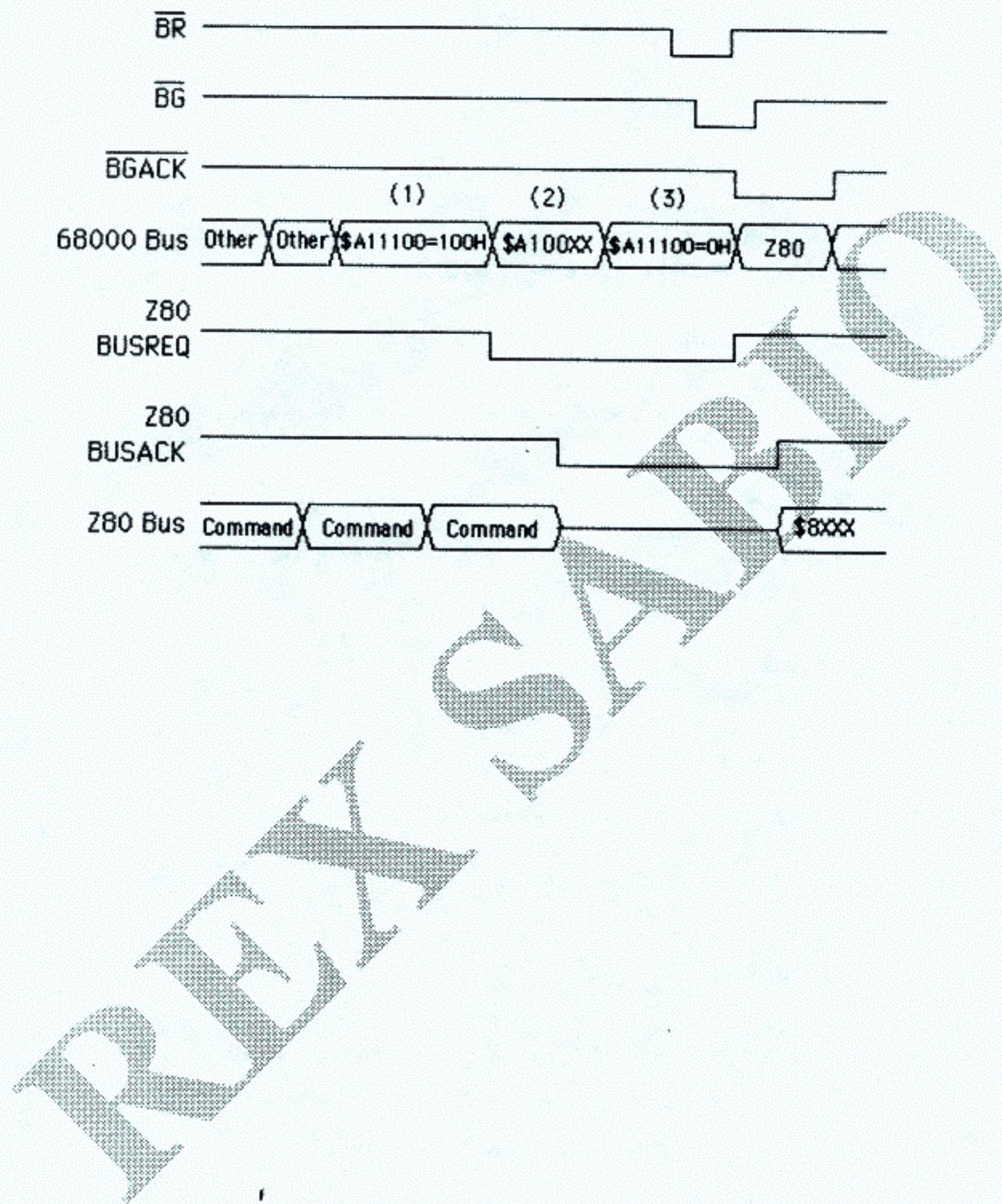
When the 68000 is accessing \$A100XX

- 1) A BUSREQ is sent to the Z80. (Stops the Z80 bus access- \$A11100=0100H)
- 2) \$A100XX is accessed.
- 3) The BUSREQ from the Z80 is cleared. (The Z80 begins bus access- \$A11100=0000H)

Change the 68000's programs to follow the steps shown above.

Miscellaneous

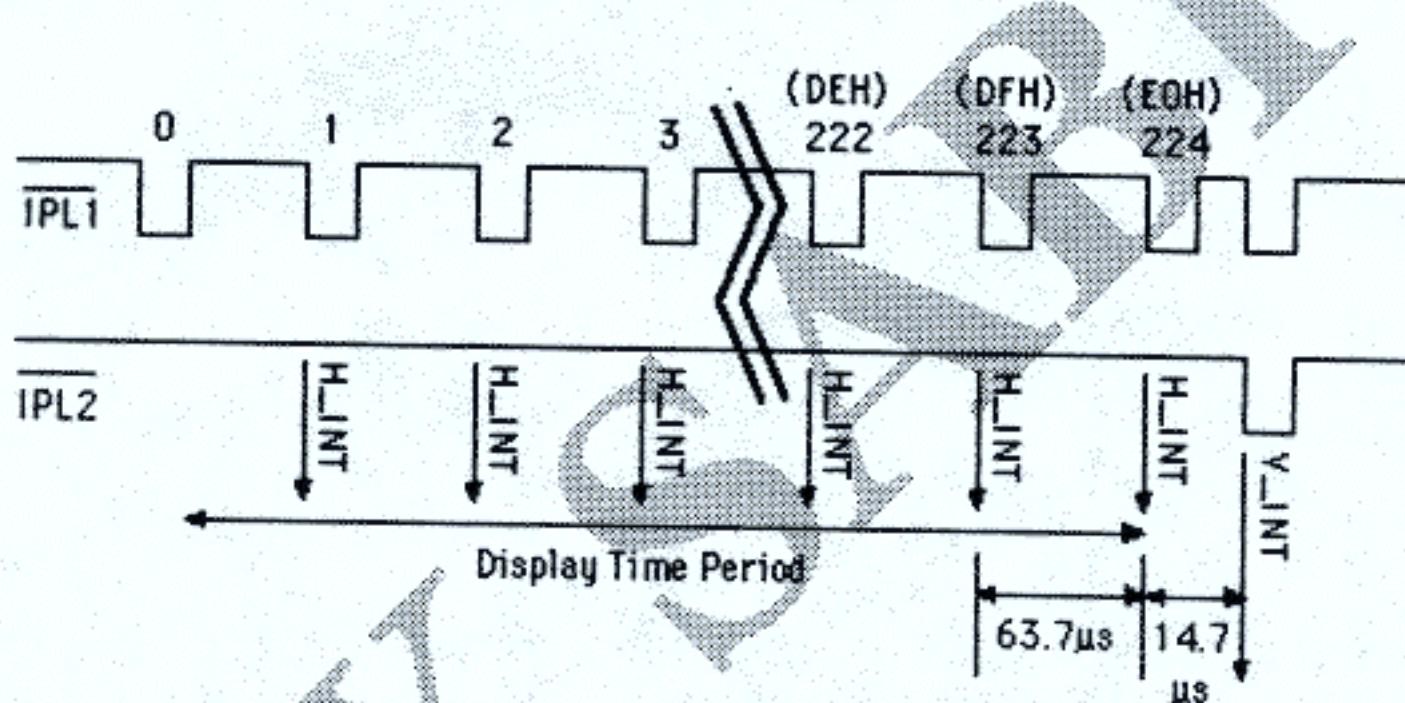
If the program structure uses something like V_INT for sync, there should be no problems as long as it is set up so that unfavorable bus access patterns are not created.



Addendum 3- Abstract from Page 2

2) When Using H_INT and V_INT with a Genesis Program

When the VDP register *0's IE1=1 and register *10=00H, H_INT and V_INT are created with the timing scheme described below. However, a problem occurs during No. 224 H_INT's timing. Since the amount of time which is available before V_INT is only 14.7 μ s, the acceptance of the No.224 H_INT means that the V_INT will miss the V_INT occurrence period. As a result, V_INT is cancelled. Moreover, when the V_INT is cancelled, the No. 225 H_INT occurs instead of the V_INT.



Solution A

- 1) Set register *0's IE1=0 before No.223 H_INT ends in order to prevent No.224 from occurring.
- 2) Set register *0's IE1=1 before V_INT is accepted in order to make H_INT valid (i.e. No. 224 also becomes valid.)
- 3) Process No.224 H_INT.

Solution B

- 1) Set the 68000's SR=25XX before No.223 H_INT ends in order to prevent No. 224 from being processed.¹
- 2) Restore the 68000's SR prior to the end of V_INT processing (Enables No.224 processing).
- 3) Process No.224 H_INT.

¹Warning: When manipulating SR, make sure not to destroy the flag.

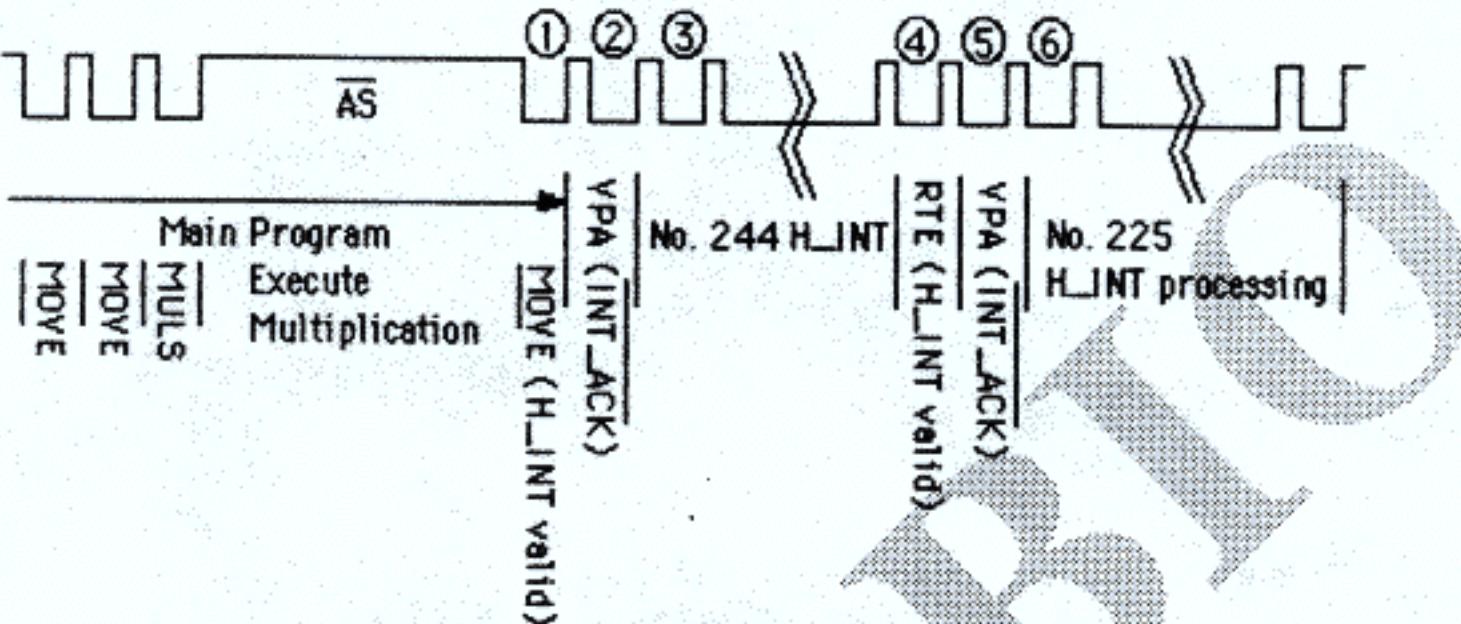


Diagram of the Problem Caused by a Multiplication Command

- 1) The MOVE command given after multiplication readies the 68000 to process H_INT.
- 2) The 68000 creates a VPA (INT_ACK). Since the VDP (for processing No. 224) does not see a VPA after H_INT occurs, H_INT is not processed. After V_INT occurs, the 68000 receives a VPA and mistakes this for V_INT processing. H_INT is stored.
- 3) Processing is executed since H_INT (No. 224) was received in part 1).
- 4) Given a RTE after processing H_INT, the 68000 readies to process H_INT again.
- 5) The 68000 creates a VPA (INT_ACK). The VDP (equivalent to No. 225) processes the stored H_INT.
- 6) Receiving H_INT (No.225) in part 4), processing occurs again.

Addendum 3- Abstract from Page 4

4) The Interrupt Problem During Data Transfer

Part 2) discussed the problems encountered when using H_INT and V_INT in a program. The same kinds of problems occur during a) an INT involving H_INT data reception; and b) an INT involving V_INT and data reception (Refer back to part 2) for details.).

The "INT during data reception" does not occur in sync in relation to V_INT and H_INT, therefore the solutions to the problem are different.

Solution in the Case of a)

During data reception, the receive flag is set at the same time of the INT. The receive flag is cleared when the data is read by the "INT during data reception" which does not overlap with H_INT. When H_INT occurs again, it overlaps and passes as the second "INT during data reception." After this occurs, the receive flag is left in clear status.

Solve the problem by discriminating between the "INT during data reception" and the H_INT which passes as the "INT during data reception." This can be determined by checking on the status of the receive flag.

Solution in the Case of b)

Set the mode which prevents the occurrence of V_INT's and then use the VDP's V_BLK status bit to detect V-timing. Use this method to perform the same type of processing as V_INT

Miscellaneous

It is possible to differentiate H_INT and V_INT timing by using H_BLK, V_BLK, and the HV_ counter.

It is possible distinguish between the "INT during data reception" and the "H and V that pass as the INT during data reception" by analyzing the receive flag.

Try to think of other solutions by using IEO, IE1, and IE2 settings within the VDP register.

Addendum 4

I. Regarding the DMA

When using DMA's other than *FILL VRAM* and *VRAM COPY*, use the following method to program

- 1) Send a bus request to the Z80.
 - 2) Base the DMA destination address set section in RAM.
 - 3) Rewrite the one leading word of data after the end of the DMA.
- 1) and 2) are used in all cases except *FILL VRAM* and *VRAM COPY*.
3) is used for WRAM to VRAM, WRAM to COLOR RAM, and WRAM to VSRAM.

II. Compatibility with the MKIII Mode

VDP registers #11 through #23 cannot be written over unless bit 2 of register #1 is not set to "1."

Reasons:

- Registers #11 through 23 are masked for MKIII mode's error trapping purposes.
- MKIII uses registers #0 through #10.
- The Genesis mode is active when register #1's bit2=1.
- The registers are set to 0 on power up.

IV. Warnings Regarding Use of Back Up RAM

- The back up RAM data is unformatted when the cartridge (product) is shipped. Always make sure to initialize during power up (In most cases the back up RAM data is cleared with \$FF at the factory, however, don't count on this as a matter of fact.).
- Although this does not occur frequently, there may be occasions when the data becomes garbled. Because of this possibility, make sure to check the back up data and perform reproduction/retrieval processing. Moreover, do not place critical data at leading and ending RAM addresses since one word in both addresses have a high probability of becoming garbled.

SEGA SOFTWARE DEVELOPMENT AND GAME STANDARDS

The following Sega of America Game Standards Policy should serve as a guideline for the development of Sega licensed cartridges by defining the types of content and themes inconsistent with Sega's corporate, marketing, and product development philosophy. SOA will not approve cartridges (i.e., audio visual work, packaging and instructions manuals)

- with sexually suggestive or explicit content;
- which reflect ethnic, racial, religious, nationalistic or sexual stereotypes or language;
- which depict excessive graphic violence;
- which use profanity in any form or which incorporate language that could be offensive by prevailing public standards and tastes;
- which encourage or glamorize the use of drugs, alcohol, or tobacco;
- which make an overt political statement;
- which contain depictions of symbols or groups which are an anathema to racial, religious or ethnic groups.
- which is a potential infringement of a legally protected copyright, or trademark

It is not possible for a single document to cover every possible eventuality in an area as broad and subjective as thematic content. For this reason, these standards of guidance should be one of common sense, practicality and prevailing public taste.

GENESIS SOFTWARE DEVELOPMENT STANDARDS

Revised 01/03/91

I. INITIAL DISPLAY

- A. The trademark, "SEGA TM", shall be displayed in the center of screen. Logo code will provided by SEGA.
- B. Check the control pad(s) while SEGA logo is displayed. If no control pad is connected, or only the 2P control pad is connected, make the loop as follows:

SEGA logo - Title - Demo - SEGA logo

(While no button is pressed.)

If the START button on either controller is pressed, the game title should be displayed. If the Player 1 controller is connected, a button press will exit from the Logo to the title. Another press should exit the title.

II. TITLE DISPLAY

- A. Display the title sequence and the text, "PUSH START BUTTON". If any button on either controller is pressed at this point, the Start/Option Select screen shall appear.
- B. Copyright marking (user can exit via any button, but can not bypass altogether.)
 - 1. Original software
©, the year of publication, Copyright holder,
e.g., ©1990 SEGA
 - 2. Licensed software
e.g., ©1987 TEKKON
PROGRAMMED GAME © 1990 SEGA
(or as required by contract.)
- C. After the title sequence the screen automatically proceeds to the demonstration mode within 5-10 seconds.

III. DEMONSTRATION

- A. If the player does not start the game during the title screen (5-10 seconds), the screen shall proceed to the demonstration.
- B. Turn on the sound. (If not already on)
- C. If the START button is pressed during the demo, the Logo screen shall appear again.

- D. If the START button is not pressed during the demo, the SEGA logo shall appear after a certain time.

POWER ON

SEGA

SEGA logo screen

When the START Button is pushed or after an interval of about 2 seconds.

CHAMPION
FINGER
WRESTLING

Game software title

PUSH START BUTTON

©1990 SEGA

Copyright notice

If START button pressed,
go on to game screens.
Otherwise in
approximately 5-10
seconds

DEMONSTRATION

10-20 seconds
Return to SEGA logo
screen. If START button
is pushed, go to Logo
screen.

IV. START/OPTION SCREEN

- A. The Start/Option screen is required with the exception of transferred games and games which might lose their game play characteristics by having the screen.

- B. Control pad maneuvers and screen display

Selection shall be made by moving D-Button upward or downward or left & right and pressing the START button.

One Player

**1 PLAYER START
OPTIONS**

If the player selects 1 PLAYER START, he can play alone.

Two player simultaneous play

**1 PLAYER START
2 PLAYER START
OPTIONS**

1. 1P control pad only

Although the screen indicates 2 PLAYER START, the cursor shall be controlled so that the player cannot select it. In order to make it clear (whether the player can select or not), different colors should be used.

When the player selects 1 PLAYER START, he can play alone.

2. 1P and 2P control pads

- a. When the 1P control pad's START button is pressed:

**1 PLAYER START
2 PLAYER START
OPTIONS**

The player can select all three; however, the player can only play with 1P control pad.

1 PLAYER START -

the player can play alone with 1P side.

2 PLAYER START -

2 players can play simultaneously (1P control pad controls the player 1 and 2P control pad controls the player 2).

3. When the 2P control pad's START button is pressed:

1 PLAYER START
2 PLAYER START
OPTIONS

The player cannot select 1 PLAYER START and he can only control the 2P control pad.

2 PLAYER START - 2 players can play simultaneously (1P control pad controls the player 1 and 2P control pad controls the player 2).

4. Two player alternate play (if applicable)

a. 1P control pad only

1 PLAYER START
2 PLAYER START
OPTIONS

The player can select all three.

1 PLAYER START - the player can play along with 1P side.

2 PLAYER START - 2 players can play alternate with 1P side control pad.

b. Option Screen

LEVEL	EASY
PLAYER	5
SOUND TEST	1
CONTROL	A JUMP B SHOT
RAPID	C SHOT
EXIT	ON

When the player selects **OPTIONS** on the Start/Option screen, the option screen shall appear.

The player can select items by pressing D-Button upward or downward and change by pressing D-Button sideways.

How to return to the title or option screen:
Select EXIT and press A, B, or C button or press the START button (the placement of cursor should not matter).

The following options shall be implemented when applicable:

LEVEL	-	Difficulty setting
PLAYER	-	Number of players
SOUND TEST	-	Sound test
CONTROL	-	Control setting
RAPID	-	Continuous shooting setting

V. PASSWORD/NAME ENTRY SCREEN

A. Select a clear font so there will be no confusion.

e.g., 0, 0, o, and Q, 1 and l

Passwords should be 12 characters or less. It is a good idea to subgroup them and use different colors for each. Do not use punctuation marks or BOTH upper and lower case letters.

B. Maneuver

PASSWORD

1. To enter password or name, select letters by D-Button and enter by button C.
2. When finished, put the cursor on END.
3. To delete letters, hold button C and move the cursor by the D-Button or use RTN(return) and PCD(proceed) function.
4. Even if the password is wrong, the letters entered shall not be erased.
5. Movement of the cursor shall be a loop (up-down-left-right).

VI. RESET/PAUSE

- A. When the reset button is pressed, during DEMO, the screen shall return to SEGA logo (the high scores, the option settings, and the password should not be cleared).
- B. During the play, the start button shall work as the pause button. During the 2 player simultaneous play, both control pads shall be able to pause and cancel.

- C. During pause, the sound shall be silenced. The background music only shall continue immediately when the pause is released.
- D. The pause function shall not work when the SEGA logo or the title logo are displayed or during the demonstration.
- E. PAUSE shall be indicated on the screen during the pause.

NOTE: Adventure and role-playing games may not require the pause mode.

VII. BASIC MANEUVERS DURING THE GAME PLAY

- A. General Maneuvers (may vary based on specific requirements of individual games)

START Button	Start/Pause.
D-Button	Used to determine the directions.
Button A	Special functions and rare maneuvers.
Button B	Passive maneuvers such as CANCEL.
Button C	Active maneuvers such as DECIDE.

- B. Sample Quick Maneuvers for action and shooting games

1. 2 actions (e.g., Ghouls 'n Ghosts, Thunder Force II)

Button A	Jump/Sub Shot
Button B	Shot/Main Shot
Button C	Jump/Sub Shot

2. 3 actions (e.g., Trojan, Altered Beast)

Button A	Punch
Button B	Kick
Button C	Jump

- C. Sample Slow Maneuvers for role-playing, simulation, and adventure games.

1. 2 actions

Button A	Decide/Opening windows
Button B	Cancel
Button C	Decide/Opening windows

2. 3 actions (e.g., Phantasy Star II)

Button A	Check Special functions
Button B	Cancel
Button C	Decide/Opening windows

D. Others

Even if there are two actions for the game, all buttons should be utilized. It is convenient if maneuvers can be changed in the option mode.

VIII. CONTINUE/ENDING

A. CONTINUE

It shall appear when the game is over, if appropriate. (This may not apply to sports or RPG's, etc.)

The CONTINUE shall be limited to a certain number of times.

The screen should return to the SEGA logo after 10-20 seconds.

B. The ending screen shall not be canceled, but it can be exited via the START button.

The screen shall return to the SEGA logo after the ending (after 10-20 seconds).

IX. GAME OVER

A. This shall appear for 60 seconds and then the screen shall display SEGA logo.

B. Pressing the START button while GAME OVER is indicated shall cause the screen to change and show the Sega logo.

X. CONTENTS OF THE GAME

A. For important indications such as the score, there shall be a two cell margin on the right and left side and a one cell margin on the top and the bottom of the screen (some monitors do not show these areas well). Do not put important score, text or time information in the top, bottom, leftmost or rightmost row or columns.

- B. When background music composed by a third party is utilized, pay careful attention to the copyright in order to avoid any infringement therein. Copyrights shall be secured for all 3rd Party music when applicable.
- C. Names which are closely related to any particular manufacturer, product, and character shall not be used.
- D. Buttons shall work as soon as they are pressed, not released, unless there is a specific approved reason to violate this rule.
- E. The "Start Button" shall work separately from other buttons.
- F. Indication of Score and High Score
High scores and previous players shall be up-to-date scores shown on TITLE, DEMONSTRATION screens.
If 2-player game, then both scores shall be displayed simultaneously.
- G. Title music shall start when the title appears and finish before the demonstration starts.
- H. The title of the game and names used within the game shall not infringe upon the trademarks of Third Parties.
- I. At the end of the game, any words or expressions which imply "copyright", "patent", or "invention" shall not be made during credits.
- J. Developer credits may be included at the end of the game, subject to approval by SEGA.

NOTE: In the case where any question arises as regards the ITEMS set forth in the SEGA GAME SOFTWARE DEVELOPMENT STANDARDS, or matters for which no stipulation is made, the developer shall, from time to time as required, consult with the SEGA Software Department and resolve said problem subject to the outcome of such consultation.

II. Corrections to the Manual

When discussing VRAM, CRAM, VSRAM access, the manual states in pages 22 through 27 that byte access is possible, however, in reality this is false.¹ The reason for this is that after the VDP address register is set, an instruction for a CPU word access (ex. a fetch operation) causes the VDP to think during the next access that it is still a word access.

From now on, please consider the text regarding byte access in pages 22-27 as being null and void.

Corrections to Mega Drive Addendum 6, Page 3

III. About Control Pads

Although the control pad is designed and produced so that simultaneous up/down or left/right input is impossible, there are times when simultaneous input occurs due to fatigue of the internal rubber parts or force on the pads which exceed design specs. Because of this, from now on, all software should have switch read routines that can process simultaneous directional input.

Corrections to MD Manual Page 77

IV.

Although the manual states that the 68000 may set the bank registers, however, in reality, this is not the case. Please execute bank switching in the Z80 from the Z80.

¹Translator's note: The page numbers are from the original Japanese-language manual.

END

I. Software Guidelines

1) The software

1) When accessing the Z80 area from within the 68000's main routine, beware of the following:

- While in the main routine
- That is, when
- \oplus On
- 1) Execute a bus request to the Z80.
- 2) Confirm execution.
- 3) An interrupt occurs.
- 4) A Z80 bus request is executed within the interrupt routine (ex. for the purpose of reading the controller pad).
- 5) The bus request is cleared within the interrupt.
- 6) Return to main routine.
- 7) Write data to Z80 area.
- 8) \ominus Off

In principle, the Z80 is left with the Z80 bus request in effect, however, in this example, it is cleared in step 5).

When this occurs, the following things occur:

- The RAM of the Z80 gets damaged.
- Incorrect data is read during Z80 bank access.
- A fix for the problem

Disable interrupts prior to step 1).

2) Problems during sound access

- 1) Sound output stops during the game.
- Diagnosis of the problem :

The busy flag was read in the FM sound generator YM2612 like this (address 4001H was accessed in the Mega Drive):

$$(CS, RD, WR, A1, A0) = (0, 0, 1, 0, 1)$$

However, in the case of the YM2612, its output is not regulated according to the conditions set up above. This results in the device being read as 'not busy' and as a consequence, ends up outputting sound.

- The fix for the problem:

When the FM sound generator's busy flag is read, do not access anything else other than

$$(A1, A0) = (0, 0) \quad (\text{Mega Drive address } 4000H).$$

3) Repeated Resets

1) The software goes out of control when resets are repeated.

- Diagnosis of the problem:

- When the reset occurs, the CPU is reset, however, the VDP is not reset.
- When the reset occurs during DMA, the VDP continues the DMA.
- The VDP is accessed right after the reset. If this is done while the VDP is executing a DMA, then this access becomes ineffective.

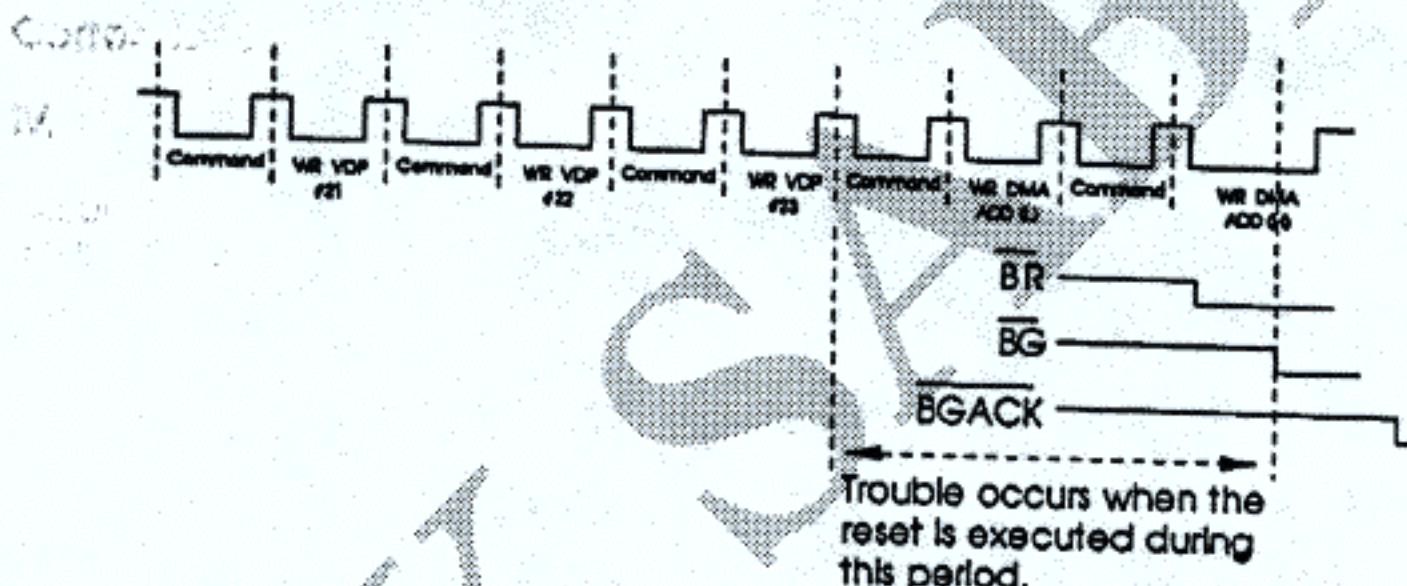
- The fix for the problem:

Before accessing the VDP after the initialization program (ICD_BLK4), check the DMA BUSY status register. If a DMA is being executed, do not attempt to access.

2) The problem still persists even after the precautions above are taken.

- Diagnosis of the problem:

The problem will occur when the reset takes place between the period of time when the CPU has finished setting the parameters to execute the DMA and the actual execution of the DMA itself.



- The fix for the problem:

On top of the precautions taken in 1), as an example, make sure not to execute a DMA right after a reset.

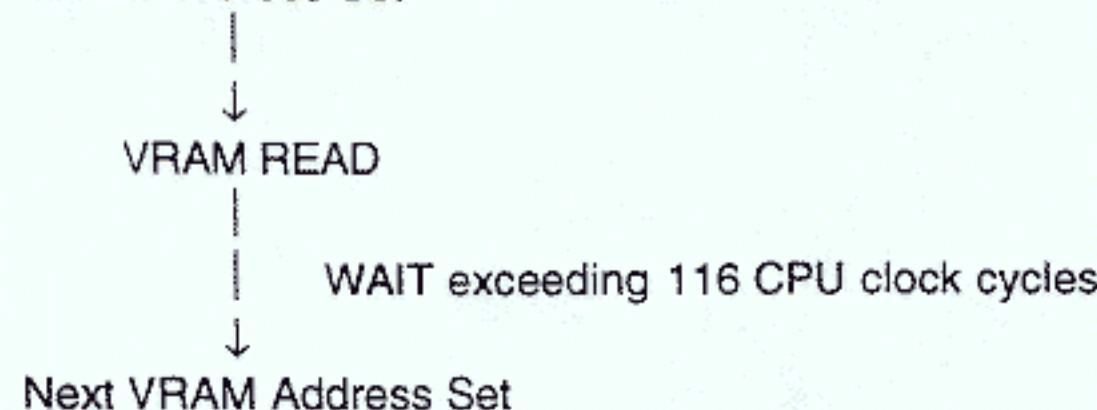
MEGA DRIVE TECHNICAL MEMO #4

- WAIT during Mega Drive VRAM READ

In the Mega Drive, during VRAM READ, when the following conditions are not met, data may not be read correctly.

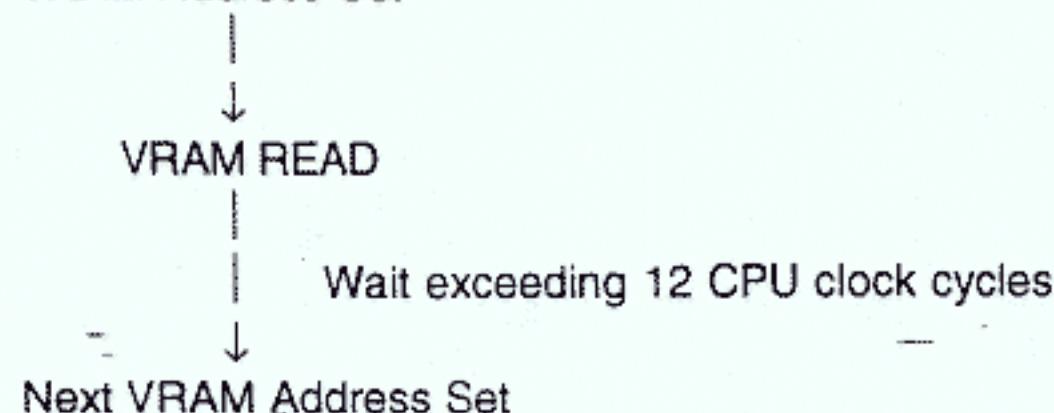
During display period and H-blank

VRAM Address Set



During V-blank

VRAM Address Set



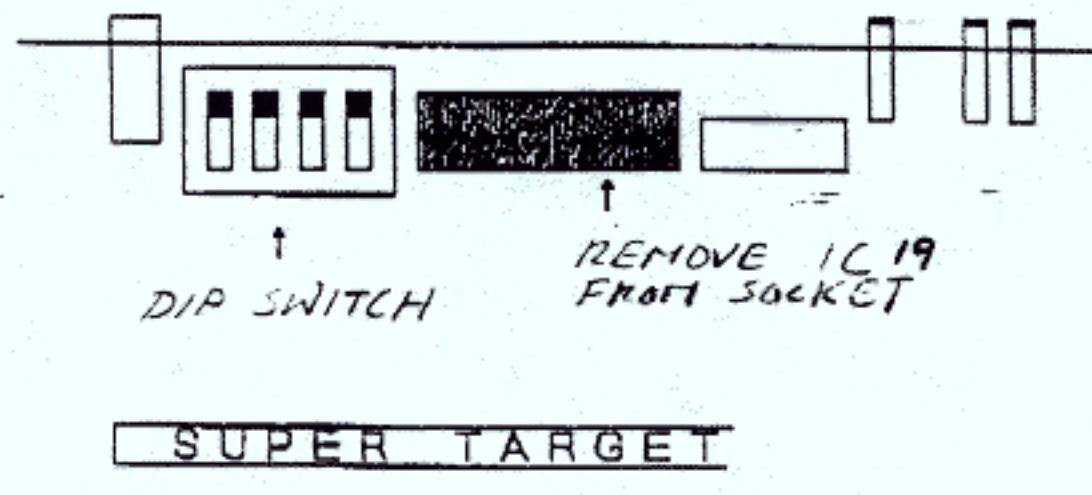
As seen above, after the last VRAM READ is completed, WAIT is required before the next VRAM address is set. With respect to VRAM READ, it can be read continuously.

- SUPER TARGET IC 19

When 68000 CPU accesses an illegal area, "DATA ACKNOWLEDGE" signal may not be sent back - causing "hang up" in the production unit. Depending on some ROM emulators or ICE (+SUPER TARGET), even the emulator may hang up. To avoid such hang ups, measures have been taken so that SUPER TARGET would return ACKNOWLEDGE signal to the CPU without system hang ups. This is done via IC 19.

As a result, due to this IC function, for SUPER TARGET to function normally, the hang up occurs in the production unit. Please take the following steps to correct the problem:

- (1) Before checking the software through SUPER TARGET + CPU, remove IC 19 from the socket.
- (2) When using ICE (ROM emulator), IC19 may be removed. Reinstall the IC should ICE (ROM emulator) act abnormally.
- (3) The above measures are common with the MEGA DRIVE and the MEGA CD.



Scot:

Here is what I discovered in the past couple of days while working with the Scaling Hardware.

Address \$FF8002: Memory Mode

Priority Mode works as follows:

In Mode OFF all scaled pixel data is written to Image Buffer

In Write Up mode only non-zero scaled pixel data is written to Image Buffer

In Write Down mode non-zero data in Image Buffer is not written to by scaled data

Address \$FF8058: Stamp Size

RPT : Repeat

This bit works exactly opposite as it is documented in the Software Manual on page 19.

Address \$FF805C : Image Buffer V-Cell Size

VCS0-4 : Specifies the number of vertical cells MINUS ONE in the image buffer.
ie. To get a buffer twelve cells tall you would store an eleven.

Trace Vector Info:

All Vector Values have fractional parts

Start Position : Has three bits of fractional data
ie. The Position one is represented by a value of 8

Deltas: Has 11 bits of fractional data.
ie. To get a step of one pixel, \$0800 would be used

Hope this helps,

Jerry Bennett

If you have any questions give me a call at (818) 774-0600

