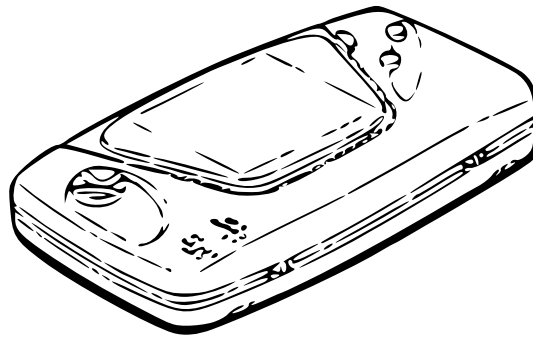


Hardware Reference Manual  
for the  
SEGA Game Gear Console



# Table of Contents

1. Precautions for Hardware	3
1. CPU clock	3
2. Memory area unusable area	3
3. VDP initialization	3
4. Interrupt	3
2. System control port	4
① I/O port 00H (Read Only) START/PAUSE button, region setting	4
② I/O port 01H (Read/Write) EXT connector	4
③ I/O port 02H (Read/Write) NMI	4
④ I/O port 03H (Read/Write) Serial communications send data	4
⑤ I/O port 04H (Read Only) Serial communications receive data	5
⑥ I/O port 05H (Read/Write) Serial communications mode setting	5
⑦ I/O port 06H (Write Only) Left-right distribution of sound	6
⑧ I/O port 30H, 31H (Read Only) Loader access (development board only)	6
⑨ I/O port DCH, DDH (Read Only) JOYSTICK board	7
3. Material: Using the ROM bank switching and backup RAM	8
A. When the memory capacity is 1 Megabit	8
B. If the capacity is 2 MBytes or more, or a backup RAM is provided	8
① Bank control register (FFFCH)	8
② Bank register 0 (FFFDH)	9
③ Bank register 1 (FFFEH)	9
④ Bank register 2 (FFFFH)	9
4. Material: MAPPING	10
MEMORY MAP, I/O MAP	10
5. Supplementary description for manual	11
A. System control port	11
③ I/O port 02H (Read/Write) NMI	11
B. System control port	11
⑥ I/O port 05H (Read/Write) Serial communications mode setting	11
C. System control port	11
⑦ I/O port 06H (Write Only) Left-right distribution of sound	11
D. Communications	12
① Connecting the communications cable	12
② Parallel communications	12
③ Serial communications	12
④ Coexistence of parallel and serial communications	13
6. VDP Manual	14
(1) GAME GEAR FEATURES	14
(2) Effective area and LCD display area	15
(3) Image display	15
① Access to VDP	15
a. Reading the status register	16
b. Writing to VDP registers (#0 to #10)	16
d. Reading from VRAM	18
e. VRAM special access	19
f. Writing to the color RAM	20
② VDP register (write only)	21
a. Register #0, register #1 (VDP control)	21
b. Register #2 (pattern name table)	22
c. Register #3	22
d. Register #4	22

e. Register #5 (sprite attribute table).....	22
f. Register #6 (sprite generator table).....	23
g. Register #7 (backdrop color).....	23
h. Register #8 (horizontal scroll).....	24
i. Register #9 (vertical scroll).....	24
j. Register #10 (interrupt).....	24
③ Standard VRAM mapping.....	26
④ Scroll screen display.....	27
a. Pattern name table.....	27
b. Pattern generator table.....	28
c. Color RAM.....	30
⑤ Displaying sprites.....	31
a. Sprite attribute table.....	31
b. Sprite generator table.....	31
c. Color RAM.....	31
d. Sprite coordinates.....	32
e. SIZE bit.....	32
f. Sprite display limits.....	32
⑥ H counter, V counter.....	33
a. H counter.....	33
b. V counter.....	34
7. PSG Manual.....	35
[1] Tone generator.....	35
(1) Method of calculating the 10-bit frequency division ratio n.....	35
(2) Tone frequency setting.....	35
(3) Example of frequency setting.....	36
a. Calculation of frequency division ratio n.....	36
b. Data sent to PSG.....	36
(4) Tone level setting.....	36
[2] Noise generator.....	37
(1) Noise generator circuit control.....	37
a. Synchronous noise (FB = 0).....	38
b. White Noise (FB = 1).....	38
(2) Noise level setting.....	38
[3] Register address feed.....	39
[4] Correlation between the sound elements and PSG.....	39
Relation between musical interval and frequency division ratio.....	39
Synchronous noise mode.....	42

# Precautions for Hardware

## 1. CPU clock

The CPU clock is 3.579545 MHz.

## 2. Memory area unusable area

### © DFF0H to DFFFH

When a ROM of 1 M or greater is used, the area for bank switching, and so on, is located between FFF0H and FFFH. This area is used for reading (using images) bank data, so do not use it as a normal work area. For this reason, it is necessary to set the stack pointer in such a way that the stack does not use this area.

Example: LD SP, 0DFF0H

### © E000H to FFFFH (excluding the area for bank switching, and so on)

This area contains the C000H to DFFFH RAM image. Do not access an image from this area but from the C000H to DFFFH area instead.

## 3. VDP initialization

Sometimes, when the power is switched ON, the reset of the CPU is canceled while the VDP remains reset. In order to prevent this, confirm that the value of the V counter in the VDP has become 80H and then access the data.

Example:

```
INIWAIT:
    IN    A,(07EH)      ; READ V-COUNTER
    CP    080H
    JP    NZ,INIWAIT
    RET
```

## 4. Interrupt

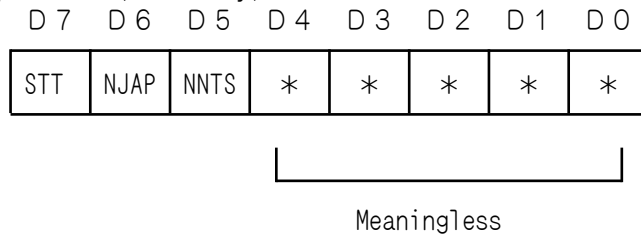
An interrupt is synchronized with the video timing (at the completion of the effective area or at an arbitrary vertical position).

This interrupt uses the Z80 mode 1 interrupt, hence "IM 1" is executed at the beginning of the program. A return from an interrupt routine is "RET".

## System control port

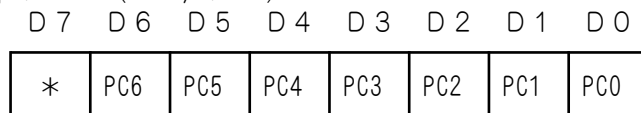
☆ Indicates the state after a power-on reset.

### ① I/O port 00H (Read Only)



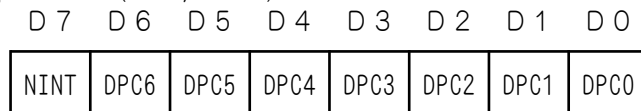
- STT  
This is an input from the START/PAUSE button.  
0: Switch ON  
1: Switch OFF
- NJAP  
0: This is the domestic (Japan) mode.  
1: This is the overseas mode.
- NNTS  
0: This is the NTSC mode.  
1: This is the PAL mode.

### ② I/O port 01H (Read/Write)



This port is used to read/write data when the EXT connector is used as a 7-bit input/output port. (The value after a power-on reset is indeterminate.)

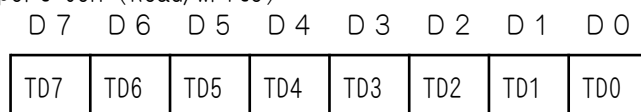
### ③ I/O port 02H (Read/Write)



- DPC6 to DPC0  
0: PCx becomes the output.  
☆ 1: PCx becomes the input.
- NINT  
0: When PC6 is input, an NMI is generated at the fall of PC6.  
☆ 1: The above operation is disabled.

※Be sure to set the status of this port to "1". When using this value, first, set the status to "1" after the first NMI is generated, then subsequently set it to "0". If you fail to do this, the next NMI will not be generated.

### ④ I/O port 03H (Read/Write)



- TD7 to TD0  
Used to set the send data during serial communications.

⑤ I/O port 04H (Read Only)

D 7   D 6   D 5   D 4   D 3   D 2   D 1   D 0

RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
-----	-----	-----	-----	-----	-----	-----	-----

- RD7 to RD0  
The receive data is set during serial communications.

⑥ I/O port 05H (Read/Write)

Serial communications mode setting

D 7   D 6   D 5   D 4   D 3   D 2   D 1   D 0

BS1	BS0	RON	TON	INT	FRER	RXRD	TXFL
-----	-----	-----	-----	-----	------	------	------

- TXFL (Read)
  - ☆ 0: The next send data is written.
  - 1: The send data cannot be written yet.
- RXRD (Read)
  - ☆ 0: There is no receive data.
  - 1: There is receive data.
- FRER (Read)
  - ☆ 0: There is no framing error.
  - 1: There is a framing error.
- INT (Read/Write)
  - ☆ 0: The following operation is disabled.
  - 1: An NMI is generated when data is received.
  - An operation such as that of I/O port 02H is unnecessary.
- TON (Read/Write)
  - ☆ 0: Sending is disabled.
  - 1: Sending is enabled. (PC4 is forcibly made the output.)
- RON (Read/Write)
  - ☆ 0: Receive disable.
  - 1: Receive enable. (PC5 is forcibly made the input.)
- BS1, BS0  
Baud rate setting

	B S 1	B S 0	Baud rate (bps)
☆	0	0	4 8 0 0
	0	1	2 4 0 0
	1	0	1 2 0 0
	1	1	3 0 0

⑦ I/O port 06H (Write Only)

Left-right distribution of sound

D 7   D 6   D 5   D 4   D 3   D 2   D 1   D 0

NOSL	TN3L	TN2L	TN1L	NOSR	TN3R	TN2R	TN1R
------	------	------	------	------	------	------	------

○ TN1R

0: The output of TONE 1 to the right is disabled.

☆ 1: The output of TONE 1 to the right is enabled.

○ TN2R

0: The output of TONE 2 to the right is disabled.

☆ 1: The output of TONE 2 to the right is enabled.

○ TN3R

0: The output of TONE 3 to the right is disabled.

☆ 1: The output of TONE 3 to the right is enabled.

○ NOSR

0: The output of NOISE to the right is disabled.

☆ 1: The output of NOISE to the right is enabled.

○ TN1L

0: The output of TONE 1 to the left is disabled.

☆ 1: The output of TONE 1 to the left is enabled.

○ TN2L

0: The output of TONE 2 to the left is disabled.

☆ 1: The output of TONE 2 to the left is enabled.

○ TN3L

0: The output of TONE 3 to the left is disabled.

☆ 1: The output of TONE 3 to the left is enabled.

○ NOSL

0: The output of NOISE to the left is disabled.

☆ 1: The output of NOISE to the left is enabled.

⑧ Loader access (development board only)

D 7   D 6   D 5   D 4   D 3   D 2   D 1   D 0

LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
-----	-----	-----	-----	-----	-----	-----	-----

I/O port 30H (Read)

○ LD7 - LD0

8-bit data which is read from the printer port

D 7   D 6   D 5   D 4   D 3   D 2   D 1   D 0

0	0	0	0	0	CLR	BUSY	NSTB
---	---	---	---	---	-----	------	------

I/O port 31H (Read)

○ NSTB

STROBE signal input (active Low)

○ BUSY

BUSY signal input (active High)

○ CLR

Loader clear button input (active Low)

When this button is pressed, LD7 to LD0 all become "0", the ACK signal becomes High, and the BUSY signal becomes Low.

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
*	*	*	*	*	*	*	ACK

I/O port 30H (Write)

○ ACK

0: The ACK signal becomes High.

1: The Ack signal becomes Low (active).

※ When the strobe signal falls, eight bits of data are latched, and simultaneously the BUSY signal automatically becomes High. Once the data has been read, the ACK signal is made Low, then High once again. The BUSY signal automatically becomes Low. The LED lights when the BUSY signal becomes High.

⑨ JOYSTICK board (Read Only)

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
DWE	UPE	TR1	TL1	RI1	LE1	DW1	UP1

I/O port DCH

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
THE	*	*	*	TRE	TLE	RIE	LEE

I/O port DDH

※ UP1-1P UP switch	UPE-EXT UP switch
DW1-1P DOWN switch	DWE-EXT DOWN switch
LE1-1P LEFT switch	LEE-EXT LEFT switch
RI1-1P RIGHT switch	RIE-EXT RIGHT switch
TL1-1P TRIGGER left button	TLE-EXT TRIGGER left button
TR1-1P TRIGGER right button	TRE-EXT TRIGGER right button
	THE-EXT (Same as PC6 input)

One bit is read to port as "1" when either there is nothing connected to the terminal or the corresponding switch is not pressed, and is read as "0" when the switch is pressed.



# Material

## Using the ROM bank switching and backup RAM

In this model, the capacity of the memory can vary between 1 and 4 Megabits by bank switching.

Note:

Area 0: 0000 to 3FFF

Area 1: 4000 to 7FFF

Area 2: 8000 to BFFF

The size of one bank is 16 Kbyte. The banks are arranged sequentially in the ROM without overlapping.

A. When the memory capacity is 1 Megabit

Area 0 is fixed to bank 0 (first 16 KBytes of the ROM), and area 1 is fixed to bank 1.

Area 2 enables the banks to be switched over by setting a value in register FFFFH.

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0	
0	0	0	0	0	1 / 0	1 / 0	1 / 0	FFFFH

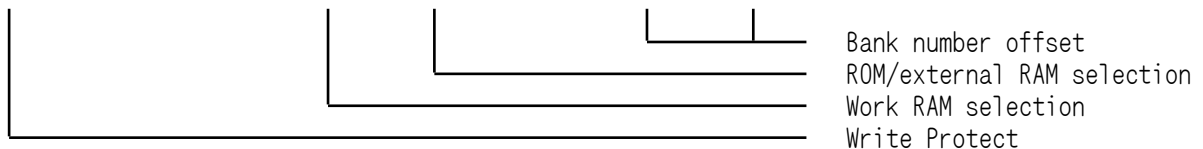
For bank 0, set 00H (same bank as for area 0).

For bank 7, set 07H.

B. If the capacity is 2 MBytes or more, or a backup RAM is provided

① Bank control register (FFFCH)

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0	
0 / 1	0	0	0	0 / 1	0	1 / 0	1 / 0	FFFCH



○ Bank number offset

All of the banks for areas 0 to 2 area shifted by 1 bank group (8 banks) time the value 1 to 3 is set here. If an offset number causes a bank number to exceed 1FH(31), it goes Bank 0. The value set here will become valid when it is subsequently set in one of bank registers 0 to 2.

○ Switching between ROM/external RAM

When 0: The ROM (bank) is assigned to area 2.

When 1: The external RAM is assigned to area 2 (when an external RAM exists).

○ Work RAM selection

If 1 is set here, normal access to the work RAM (C000H to DFFFH) of the main body will be prevented.

Be sure, therefore, to set 0.

○ WRITE PROTECT

If 1 is set here when the development RAM board is used, data re-write will not take place.

Set 0 in mass-production machines.

② Bank register 0 (FFFDH)

0000H to 03FFH of area 0 is fixed to bank 0. Bank selection of other areas can be performed by setting a value at FFFDH.

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0	
0	0	0	0 / 1	0 / 1	1 / 0	1 / 0	1 / 0	FFFDH

For bank 00: Set 00H.

For bank 1F: Set 1FH.

③ Bank register 1 (FFFEH)

A bank selection can be performed in area 1 by setting a value in FFFEH.

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0	
0	0	0	0 / 1	0 / 1	1 / 0	1 / 0	1 / 0	FFFEH

For bank 00: Set 00H.

For bank 1F: Set 1FH.

④ Bank register 2 (FFFFH)

A bank selection can be performed in area 2 by setting a value in FFFFH.

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0	
0	0	0	0 / 1	0 / 1	1 / 0	1 / 0	1 / 0	FFFFH

For bank 00: Set 00H.

For bank 1F: Set 1FH.

Caution:

The above registers are not initialized when the power is switched on. For this reason, be sure to initialize them program in 0 to 3FFFH.. (The first 1 KByte is fixed for this purpose.) Sometimes, the work RAM cannot be accessed normally, hence the work RAM (and also sub-routines, ) are allowed to be used after this initialization. When using a backup RAM, do not use the first and last addresses because there is a possibility of the data being changed when the power is switched on.

Example:

```

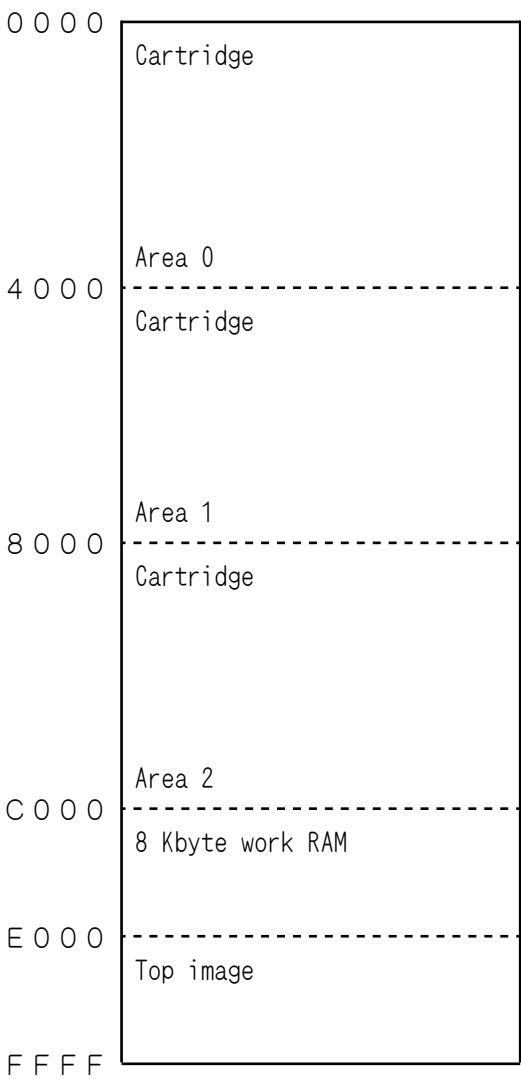
ORG      00000H
DI
IM       1
LD       SP,0DFF0H

LD       A,000H
LD       (0FFFCH),A
LD       A,000H
LD       (0FFFDH),A
LD       A,001H
LD       (0FFFEH),A
LD       A,002H
LD       (0FFFFH),A

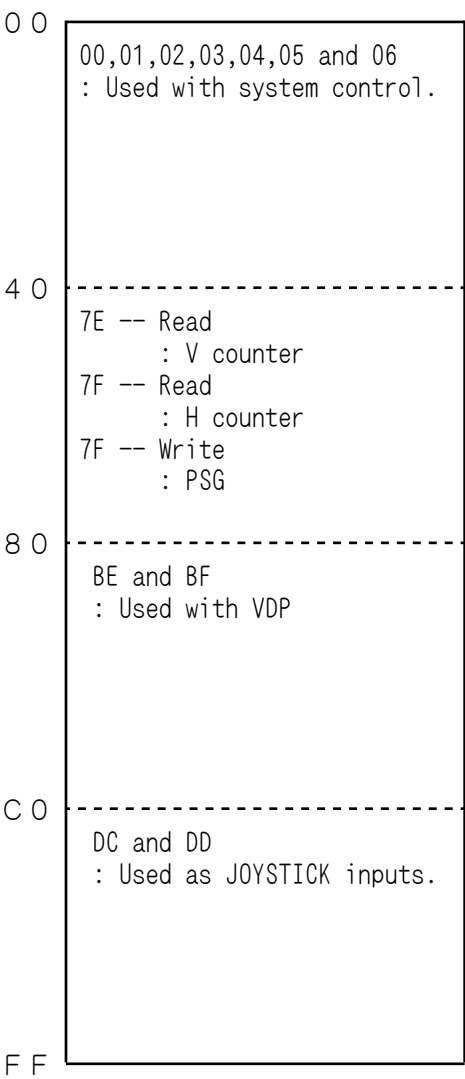
```

# Material MAPPING

MEMORY MAP



I/O MAP



Never access data using an image address.

## Supplementary description for manual

### A. System control port

#### ③ I/O port 02H (Read/Write)

\* Normally, be sure to make the status of this port "1". When using this value, first, set the status to "1" after the first NMI is generated, then subsequently set it to "0". If you fail to do this, the next NMI will not be generated.

#### Addition:

An NMI is enabled after the execution of one command from when the port is set to "0". This is to prevent the NMI from becoming active once again in the NMI routine. Normally, therefore, perform the following processing.

NMI:

```
.  
.   
.   
.   
LD      A,11XXXXXXB  
OUT      (002H),A  
LD      A,01XXXXXXB  
OUT      (002H),A  
RETN                                ; An NMI is enabled after this command.
```

### B. System control port

#### ⑥ I/O port 05H (Read/Write)

Mode setting for serial communications

\* INT

\* There is no need to perform an operation such as that of I/O port 02H.

#### Addition:

When using this function (serial communications NMI), set NINT of I/O port 02H to the disable state ("1"). An NMI will be generated at the fall of the pulse at the NMI terminal. If, however, a serial communications NMI is generated, the NMI terminal will go LOW, preventing the next NMI from becoming active. The NMI terminal is made HIGH as a result of reading the data of I/O port 04H, so read the data each time an NMI is generated. If it is conceivable that the NMI terminal may already be LOW at the start of the communications, perform a "dummy" read operation once.

### C. System control port

#### ⑦ I/O port 06H (Write Only)

Left-right distribution of sound Supplementary explanation. When the headphones are plugged in, the output from the speaker is cut off and instead the sound will be heard in stereo from the headphones. When the earphones are not plugged in, the sound will be heard from the speaker in monaural. In the latter case, the distribution of the sound from all channels (three tones & noise) will be enabled. If the output from the left and right channels was disabled not by attenuator control but by distribution, the sound will not be heard from the headphones but will be heard from the speaker. To turn off both the left and right channels of the speaker, use the PSG, by PSG side control.

#### D. Communications

① Connecting the communications cable Cross-connect the game gear communications cable as shown below.

Communications connector

1	P C 0	—————	To opposite side PC2
2	P C 1	—————	To opposite side PC3
3	P C 2	—————	To opposite side PC0
4	P C 3	—————	To opposite side PC1
5	+ 5 V		
6	P C 4	—————	To opposite side PC5
7	P C 6	—————	To opposite side PC6
8	G N D	—————	To opposite side GND
9	P C 5	—————	To opposite side PC4
1 0	N C		

#### ② Parallel communications

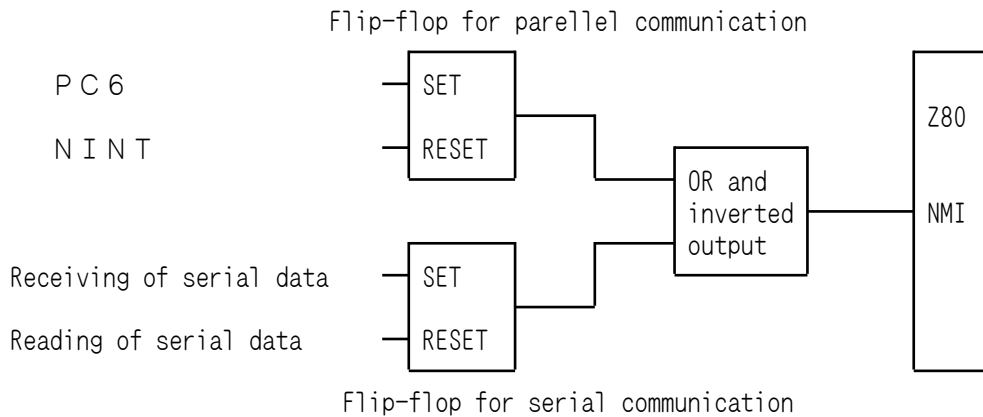
PC0 to PC6 can be set to an arbitrary input or output by means of the control register of the I/O port. Be sure to set the connecting terminals so that the terminal on one side is the output, and that on the opposite side is the input. (Never make the terminals on both sides the output.) In the case of parallel communications, control the exchange of data either by polling using software (check the data), or by applying an interrupt (NMI) using PC6. When applying an NMI using PC6, however, it is necessary to take noise into account because an NMI will be generated by a momentary change in PC6.

#### ③ Serial communications

Serial communications can be performed in one of two single directions, from PC4 (output from one's own side) → PC5 (input to opposite side), or from PC5 (input to one's own side) ← PC4 (output from opposite side). Serial and parallel conversion and interrupt (NMI) generation (when data is received) accompanying the receiving or sending of data take place automatically when the hardware is connected to these terminals. To perform serial communications, set TON and RON of I/O port 05H to "1". By doing this, PC4 will automatically become the output, and PC5 the input. These settings will take priority over the PC4 and PC5 input/output settings. Bits other than those of PC4 and PC5 will become the settings of I/O port 02H. (Like (2), never make both terminals the output.) When performing serial communications only, be sure to set NINT of I/O port 02H to prevent PC6 from generating an NMI.

#### ④ Coexistence of parallel and serial communications

When performing serial communications, PC4 and PC5 are used to send and receive serial data. Parallel communications can be performed using the bits other than these. An NMI can be generated by PC6 and also by receiving of serial data. In the former case, care must be taken because PC6 does not have a data receiving flag such as the serial RXRD. The blocks of the circuit used to generate these NMI are shown below.



An NMI is generated when one of the two reset flip-flops is set. (This is because an NMI is generated not when the pulse level is LOW but when the pulse falls.) It should be appreciated that it is necessary to reset the set flip-flop so that the next NMI can be generated.

- END -

# VDP Manual

## (1) GAME GEAR FEATURES

- \* 16-kbyte VRAM

- \* Scroll SCREEN: Horizontal 20 cells x vertical 18 cells (one cell = 8 x 8 dots)  
The virtual area is 32 cells in the horizontal direction x 28 cells in the vertical direction, permitting smooth scrolling in the horizontal and vertical directions.

- \* The scroll screen can be left-right reversed, and up-down reversed, and also its priority with respect to sprites can be selected.

- \* 64 colors out of a total of 4096 colors can be displayed for each dot of the scroll surface. By designating a palette for each cell, two kinds of 16-color groups can be selected.

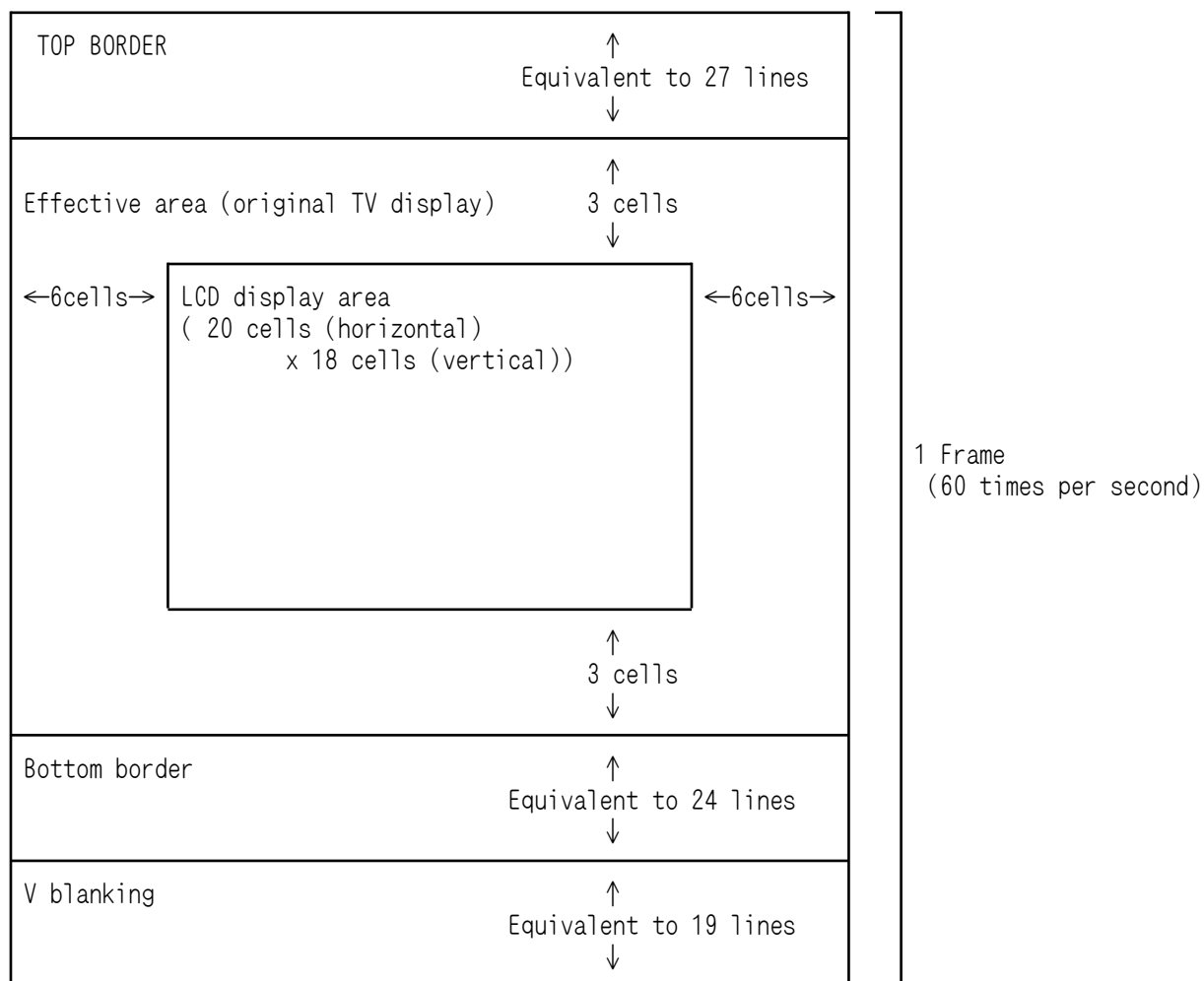
- \* Sixteen sprites (movable objects) can be displayed on a single screen. Up to eight sprites of either 8 x 8 dots or 8 x 16 dots (vertical) in size can be selected on the same horizontal line.

- \* Sixteen colors out of a total of 4096 can be displayed for each dot of a sprite.

- \* The maximum number of character pattern definitions (8 x 8 dots) is 448 for both the scroll surface and sprites (256 of these are used for sprites, however this is reduced to 128 for 8 x 16 sprites).

## (2) Effective area and LCD display area

This VDP was initially designed on the basis of a TV (NTSC) format, hence only a portion of the picture created by the VDP is displayed on the LCD. This relationship is shown in the figure below.



Consequently, when a developing board or TV adapter is installed, the LCD display part will appear on part of the screen, and the backdrop color will be displayed on the remaining part. (Set to a color approaching that of a game screen.) When the screen is ON and timing is in the effective area, the VDP will generate an image, hence (even for a part which is not displayed on the LCD) if timing is within this area, the game program will be subjected to various restrictions (wait condition, etc.).

## (3) Image display

Note: In the following description, there are bits that are fixed at "0" or "1", and should remain in this default position.

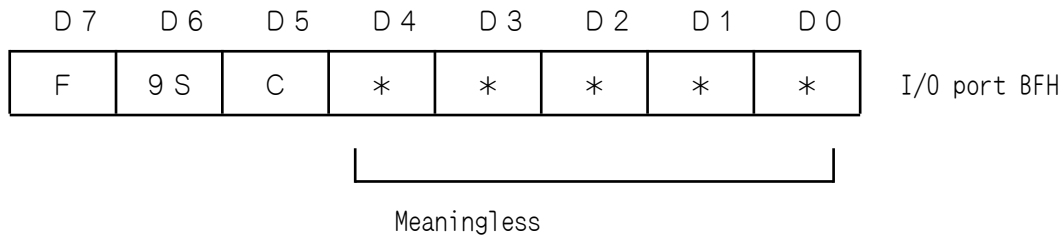
### ① Access to VDP

It is necessary to set data in the VDP register, color RAM, VRAM, and so on, in order to display an image on the screen. This is because data cannot be accessed directly from the CPU, and must be accessed instead via the VDP assigned to the I/O port.



a. Reading the status register

The status register indicates various states of the VDP. It can read the CPU by reading the I/O port BFH. This register can read the CPU at any time without any need to take account of the VDP delay.



○ Interrupt flag (F)

This flag is "1" when the effective area is completed. If, at this time, the IE bit of the VDP register #1 is set to "1", the interrupt line from the VDP to the CPU will become Low, causing an interrupt to be applied (generally called a V interrupt). If the program leaves the interrupt routine in this state, the interrupt will be applied again immediately (an interrupt of Z80 will not occur at the edge of plus, hence the status register will be read at the beginning of the interrupt routine and the flag will be reset).

Example:

```

ORG    00038H
PUSH   AF
IN      A,(0BFH)      ; RESET STATUS FLAG
;
POP     AF
EI
RET
  
```

\* This flag is not set in the case of an interrupt (normally called an H interrupt) at an arbitrary vertical position, hence by checking this flag it is possible to identify two kinds of interrupts. A Z80 interrupt uses mode 1, hence "IM 1" is implemented.

○ 9th sprite (9S)

If the 9th and higher sprites exist on the same horizontal line (in the effective area), and the interrupt flag (F) is "0", the 9S bit will be set to "1". (within the effective area)

○ Collision flag (C)

This flag is set to "1" if dots of color codes other than 0, of two or more sprites collide (coincide). --In the effective area

※ The above flags are reset by a power-on reset and a status register read-out, and are renewed each frame. Once a flag is set, it will not be reset again apart from a power-on reset, so long as the status register is not read. For example, if sprites overlap each other, the collision flag will be set, however if the status register is not read, the flag will remain set even if the sprites subsequently separate from each other.

b. Writing to VDP registers (#0 to #10)

A selection of the display mode or other functions and also data for each base address setting are written to the VDP registers (write-only registers). Data transfer from the CPU takes place in the following format. Data can be written to these registers without any need to take account of VDP delay.

	b7	b6	b5	b4	b3	b2	b1	b0
First byte: Data set in the register (I/O port BFH)	D7	D6	D5	D4	D3	D2	D1	D0
Second byte: Register Selection (I/O port BFH)	1	0	0	0	R3	R2	R1	R0

To set data in a VDP register, the data is inputted in the first byte. The second byte is used to indicate the register where the data is to be transferred.

The bottom four bits (R3 to R0) of the second byte designate the data transfer destination registers (#0 to #10). b7 must be "1" and b6 to b4 must be "0". Never attempt to access registers that do not exist (#11 to #15).

Example: When setting E0H in register #1

```
[IN  A,(0BFH)]    ; INITIALIZE
LD   A,E0H
OUT  (0BFH),A      ; DATA
LD   A,081H
OUT  (0BFH),A      ; REGISTER NO.
```

※ The logic inside the VDP determines whether data sent to the VDP is the first byte or the second byte. This internal logic is set to receive the first byte in the following cases.

- After a power-on reset
- After the status register has been read
- After the second byte has been written
- I/O port BEH write or read (VRAM or color RAM access)

Consequently, if data is written in the correct sequence, [IN A, (0BFH)] in the previous example can be omitted. Here, it is necessary to take steps to prevent the CPU from accepting an interrupt while data is being written to a register or VRAM address setting is being carried out (described later). If an interrupt is applied after the first byte has been sent, the data will fail to be transferred correctly if the status register is read during the interrupt (the second byte written after the program leaves the interrupt routine will be received as the first byte).

d. Reading from VRAM

The CPU reads data from the VRAM via the VDP. The addresses are auto-incremented.

(See sub-section "Writing to VRAM".)

	b7	b6	b5	b4	b3	b2	b1	b0
First byte: Address set-up (I/O port BFH)	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0
Second byte: Address set-up (I/O port BFH)	0	0	A13	A12	A11	A10	A 9	A 8
Third byte: Write Data (I/O port BEH)	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
Necessary number of repetitions								

The bottom eight bits of the VRAM address are set up by the first byte.

The top six bits are set up by the second byte. Be sure to set b7 and b6 to "0".

The data is read by the third byte.

Once the address register has been set up, the data will be incremented automatically each time the third byte data is transferred.

○ A delay time is necessary for the CPU to read data from the VRAM. A wait of at least 28 clock pulses is necessary during the first third-byte transfer after the completion of address setting prior to a read operation (this is slightly different to the case of a write operation to the VRAM).

Insert the same wait for reading the subsequent third bytes. This applies only to cases where the VDP displays a picture (effective area). In the following two cases, there is no need for any wait whatever. (This is the same as for a write operation to the VRAM.)

○ When the blank bits of the VDP register #1 are set to "0" and the screen is turned OFF.

○ During the interval of 4.3 m from when an interrupt occurs upon completion of the effective area until the commencement of the next effective area.

Example: Write two addresses continuously from VRAM address 0000H. (effective area & screen ON)

```

LD      A,000H
OUT     (0BFH),A
LD      A,000H
OUT     (0BFH),A
PUSH    IX          ; WAIT 15 CLOCK
POP     IX          ; WAIT 14 CLOCK      TOTAL 29 CLOCK
IN      A,(0BEH)
PUSH    IX          ; WAIT
POP     IX          ; WAIT
IN      A,(0BEH)
PUSH    IX          ; WAIT
POP     IX          ; WAIT

```

e. VRAM special access

If data is written to a VRAM using address auto increment, it will be written to a continuous address. Data can be written to discrete addresses by dummy reading it. Observe the wait conditions while a screen is being displayed in the effective area.

Example: Write 256 bytes 01H to each address from VRAM address 3800H.  
(effective area & screen ON)

```
LD      A,000H
OUT     (0BFH),A
LD      A,078H
OUT     (0BFH),A
LD      B,000H
LOOP:
LD      A,001H      ;      7 CLOCK  TOTAL 41 CLOCK
OUT     (0BEH),A    ;      11 CLOCK
PUSH    IX          ; WAIT  15 CLOCK
POP     IX          ; WAIT  14 CLOCK  TOTAL 29 CLOCK
IN      A,(0BEH)
PUSH    AF          ;      11 CLOCK
POP     AF          ;      10 CLOCK
DJNZ    LOOP        ;      13 CLOCK
```

○ Do not perform irregular access to VRAM (e.g. setting read addresses and writing data to them, or vice-versa). Also, do not use the above method to write data to a color RAM.

f. Writing to the color RAM

The VDP contains a 12-bit x 32-word color RAM. This color RAM is a write-only RAM. The CPU transfers data to the color RAM via the VDP, using an auto increment address register.

	b7	b6	b5	b4	b3	b2	b1	b0
First byte: Address set-up (I/O port BFH)	0	0	A 5	A 4	A 3	A 2	A 1	A 0
Second byte: 0C0H set-up (I/O port BFH)	1	1	0	0	0	0	0	0
Third byte: Write Data (I/O port BEH) Repetition of write necessary number of times	(For even addresses) G 3 G 2 G 1 G 0 R 3 R 2 R 1 R 0 (For odd addresses) 0 0 0 0 B 3 B 2 B 1 B 0							

The first byte sets up five bits of the color RAM address.

The second byte always sets C0H.

The third byte transfers data.

Once the address register is set up, it is automatically incremented each time the third byte of data is transferred. The color RAM has two addresses (even and odd addresses) which comprise a single color (12 bits). Normally, therefore, the even address is set up, then the R & G data and B data are set in that sequence. Actual writing of data to the color RAM takes place when data is set in the odd address (12 bits of data are written). Note, however, that when the odd address is set up and B data written, the R & G data previously set will be written.

○ The delay time conditions when the CPU writes data to the color RAM are exactly the same as those for writing data to the VRAM.

Example:

Write 0FH & 00H (bright red) and F0H & 00H (bright green) from color RAM address 04H.  
(effective area & screen ON)

```

LD      A,004H
OUT     (0BFH),A
LD      A,0C0H
OUT     (0BFH),A
LD      A,00FH
OUT     (0BEH),A
PUSH    AF          ; WAIT 11 CLOCK
POP     AF          ; WAIT 10 CLOCK
LD      A,000H      ;      7 CLOCK    TOTAL 28 CLOCK
OUT     (0BEH),A
PUSH    AF          ; WAIT
POP     AF          ; WAIT
LD      A,0F0H
OUT     (0BEH),A
PUSH    AF          ; WAIT
POP     AF          ; WAIT
LD      A,000H
OUT     (0BEH),A
PUSH    IX          ; WAIT 15 CLOCK
POP     IX          ; WAIT 14 CLOCK    TOTAL 29 CLOCK

```

○ If data is written to the color RAM when a TV scan is on the screen (including the border),

the screen will flicker. This cannot be avoided even by setting the BLANK bit to "0" to turn the screen OFF. For this reason, write data to the color RAM during V blanking.

② VDP register (write only)

a. Register #0, register #1

(These two registers are reset to "0" at power switch-on.)

Register #0

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
MVS	0	0	IE1	EC	1	1	0

Register #1

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1	BLANK	IE	0	0	0	SIZE	0

○ EC (Early Clock)

0: Normal

1: The horizontal position of all sprites shifts eight dots to the left.

○ IE, IE1 (Interrupt Enable)

IE is an interrupt enable bit used at the completion of the effective area.

0: Disable

1: Enable

IE1 is an interrupt enable bit used at an arbitrary vertical position.

0: Disable

1: Enable

○ MVS

0: Normal

1: The two cells at the right end of the LCD screen are not scrolled in the vertical direction.

※ A horizontal scroll is not disabled.

○ SIZE

0: Normal

1: The sprite size becomes 8 x 16 dots. In this case, the top seven bits of the character No. are enabled (number of definitions = 128 kinds).

○ BLANK

0: Nothing is displayed on the screen. In this case, the backdrop color is displayed, and the wait used for VDP access is unnecessary.

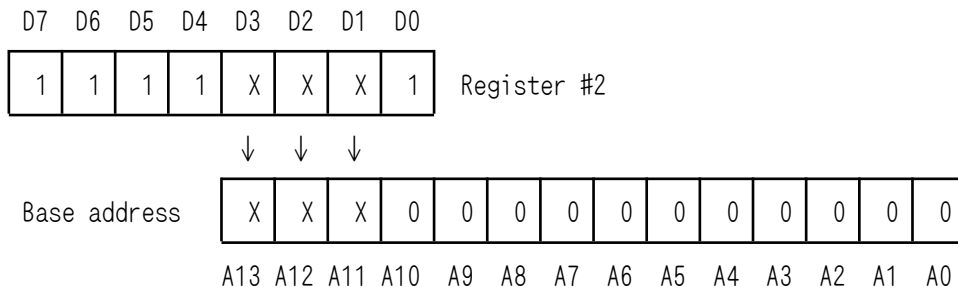
1: An image is displayed on the screen.

※ The screen display can be turned ON and OFF at any time.

b. Register #2

(This register is not reset at power switch-on, hence its contents are indeterminate.)

This register determines the base address (starting address) of the pattern name table in the V RAM. The pattern name table requires 32 cells (horizontal) x 28 cells (vertical) x 2 bytes = 1792 (700H) bytes. The relation between the set data and the base address is shown below.



$$((\text{Data}) - \text{F1H}) \times 400\text{H} = \text{Base address}$$

Set data	Base address
F 1 H	0 0 0 0 H
F 3 H	0 8 0 0 H
F 5 H	1 0 0 0 H
F 7 H	1 8 0 0 H
F 9 H	2 0 0 0 H
F B H	2 8 0 0 H
F D H	3 0 0 0 H
F F H	3 8 0 0 H

※ Normally FFH is set and the pattern name table started from 3800H.

c. Register #3

(This register is not reset at power switch-on, hence its contents are indeterminate.)

※ Be sure to set FFH in this register with a program.

d. Register #4

(This register is not reset at power switch-on, hence its contents are indeterminate.)

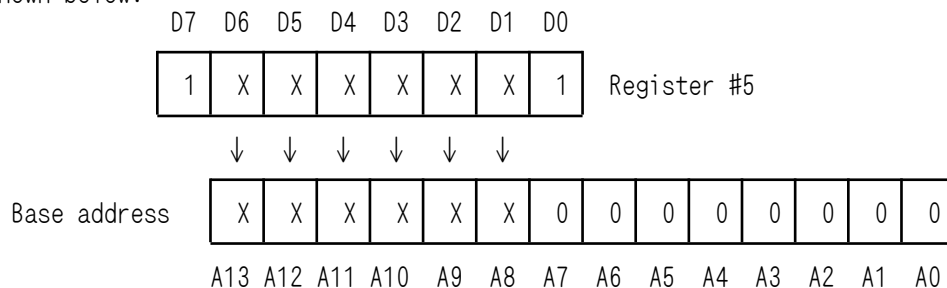
※ Be sure to set FFH in this register with a program.

e. Register #5

(This register is not reset at power switch-on, hence its contents are indeterminate.)

This register determinates the base address of the sprite attribute table in the VRAM.

The sprite attribute table consists of 256 (100H) bytes. The relation between the set data and the base address is shown below.



((Data)- 81H) x 80H = Base address

Data	Base addr	Data	Base addr	Data	Base addr	Data	Base addr
81H	0000H	A1H	1000H	C1H	2000H	E1H	3000H
83H	0100H	A3H	1100H	C3H	2100H	E3H	3100H
85H	0200H	A5H	1200H	C5H	2200H	E5H	3200H
87H	0300H	A7H	1300H	C7H	2300H	E7H	3300H
89H	0400H	A9H	1400H	C9H	2400H	E9H	3400H
8BH	0500H	ABH	1500H	CBH	2500H	EBH	3500H
8DH	0600H	ADH	1600H	CDH	2600H	EDH	3600H
8FH	0700H	AFH	1700H	CFH	2700H	EFH	3700H
91H	0800H	B1H	1800H	D1H	2800H	F1H	3800H
93H	0900H	B3H	1900H	D3H	2900H	F3H	3900H
95H	0A00H	B5H	1A00H	D5H	2A00H	F5H	3A00H
97H	0B00H	B7H	1B00H	D7H	2B00H	F7H	3B00H
99H	0C00H	B9H	1C00H	D9H	2C00H	F9H	3C00H
9BH	0D00H	BBH	1D00H	DBH	2D00H	FBH	3D00H
9DH	0E00H	BDH	1E00H	DDH	2E00H	FDH	3E00H
9FH	0F00H	BFH	1F00H	DFH	2F00H	FFH	3F00H

※ Normally FFH is set and the sprite attribute table started from 3F00H.

f. Register #6

(This register is not reset at power switch-on, hence its contents are indeterminate.)

This register determines the base address of the sprite generator table. The relation between the set data and the base address is shown below.

D7 D6 D5 D4 D3 D2 D1 D0

1	1	1	1	1	X	1	1
---	---	---	---	---	---	---	---

Register #6

↓

Base address

X	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---

A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

((Data)- FBH) x 800H = Base address

Set data	Base address
FBH	0000H
FFH	2000H

g. Register #7

(This register is set to "0" at power switch-on.) They are used to set the backdrop color.

D7 D6 D5 D4 D3 D2 D1 D0

0	0	0	0	C3	C2	C1	C0
---	---	---	---	----	----	----	----

There are 40H addresses in the color RAM. Of the 16 sets of color data in addresses 20H to 3FH (palette 1 side), the sets designated by the bottom four bits (C3 to C0) of this register constitute the backdrop color.



h. Register #8

(This register is set to "0" at power switch-on.)

It is used to set the horizontal scroll.)

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
HS 7	HS 6	HS 5	HS 4	HS 3	HS 2	HS 1	HS 0

Each time a value of 1 is set in this register, the scroll screen moves one dot to the right in the horizontal direction. The number of dots in the horizontal direction in the virtual area is 256. Consequently, if a value of -1 (FFH) is set, the screen will move one dot to the left in the horizontal direction. This register is effective only for the scroll screen. It has no effect on sprites. The value written to this register is latched at the timing of the H counter F4H (see "(5) H counter, V counter"), then becomes active. Consequently, horizontal scrolling can be performed one line at a time by using an interrupt at an arbitrary vertical position to re-write the data.

i. Register #9

(This register is set to "0" at power switch-on.)

It is used to set the vertical scroll.)

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
VS 7	VS 6	VS 5	VS 4	VS 3	VS 2	VS 1	VS 0

Each time a value of 1 is set in this register, the scroll screen moves one dot in the upward direction. The number of dots in the vertical direction in the virtual area is 224. Consequently, if a value of 224 or more is set, the screen will scroll in the upward direction by an amount corresponding to that value minus 224. Also, the value that was set in this register immediately in front of the effective area (for line 511) will become effective during that frame, preventing the vertical scroll from being changed while display timing.

j. Register #10

(This register is set to "1" at power switch-on.)

It is used to control an interrupt at an arbitrary vertical position.

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
VC 7	VC 6	VC 5	VC 4	VC 3	VC 2	VC 1	VC 0

The value set in this register is loaded in the down counter in the VDP. This counter counts down each line. When the count is 0, an interrupt is generated. A count-down takes place only for the line in the effective area and the line immediately preceding it (line 511). For other lines, the down counter simply continues to load the value written to this register without counting down or generating an interrupt. This interrupt is generated during H blanking (H counter F4H), and the value in the register at this time is loaded once again to the down counter.

When 00H is set in the register, an interrupt is generated at every line, and when 01H is set in the register, an interrupt is generated at every second line.

As an example, consider a method of horizontal scrolling as shown in the figure below.

1) After completion of the effective area, set "1" in IE1 of register #0, and set 0BH in register #10. These values continue to be loaded in the down counter until line 511 appears.

2) An interrupt is generated after line 10 has been scanned. 0BH will be loaded in the down counter once again. The status register is read and the interrupt cleared (this takes place each time), and 00H is set in register #10.

3) An interrupt is generated after line 22 has been scanned. 00H is loaded in the down counter. 03H is set in register #8 as the horizontal scroll. (It becomes effective for the first time when the next H counter F4H arrives.)

4) An interrupt is generated after line 23 has been scanned. 00H is loaded in the down counter once again. The interrupt is generated at the timing of the H counter F4H, hence the horizontal scroll 02H set in 3) is also effective at this time. 05H is set in register #8 as the horizontal scroll.

5) An interrupt is applied after line 24 has been scanned by the horizontal scroll 03H. 00H is loaded in the down counter. The horizontal scroll 05H is effective. 07H is set in register #8, and 5CH (92) is set in register #10.

6) An interrupt is applied after line 25 has been scanned by the horizontal scroll 05H. A horizontal scroll of 07H is effective, and 5CH is loaded in the down counter. 00H is set in register #10.

7) The next interrupt is applied after line 118 has been scanned. 00H is loaded in the down counter, then an interrupt is applied at each line. 09H is set in register #8.

8) An interrupt is applied after line 119, and horizontal scroll 09H becomes effective. The horizontal scroll up to now is 07H. 0BH is set in register #8. Subsequently, "0" is set in IE1 and the interrupt is disabled.

※ The value of register #8 (horizontal scroll) becomes effective after H counter F4H (delayed by one line). Note that in this case the down counter is re-loaded when an interrupt occurs at an arbitrary vertical position.

	Line 24 horizontal scroll: 03H
	Line 25 horizontal scroll: 05H
Horizontal scroll between line 26 and line 119: 07H	
	Line 120 horizontal scroll: 09H
Horizontal scroll between line 121 and line 167: 0BH	
	Line 167

※ Line 24 and line 167 are equivalent to the top and bottom lines, respectively, on the LCD screen.

### ③ Standard VRAM mapping

Example: Setting the register

```

LD    HL,TBLREG
LD    B,11
LD    C,080H

LOOP:
LD    A,(HL)
INC   HL
OUT   (0BFH),A
LD    A,C
INC   C
OUT   (0BFH),A
DJNZ  LOOP
RET

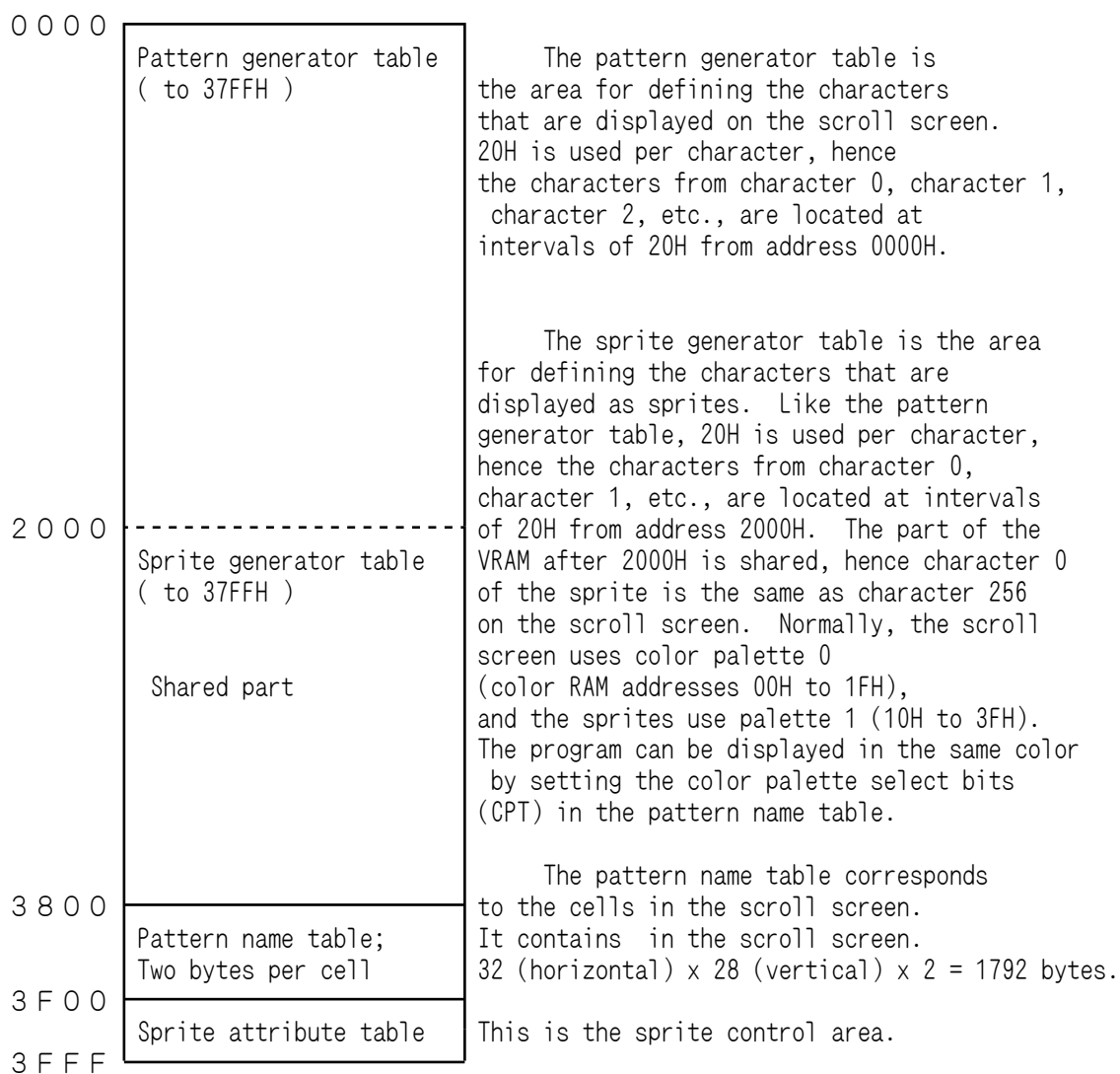
```

TBLREG:

```

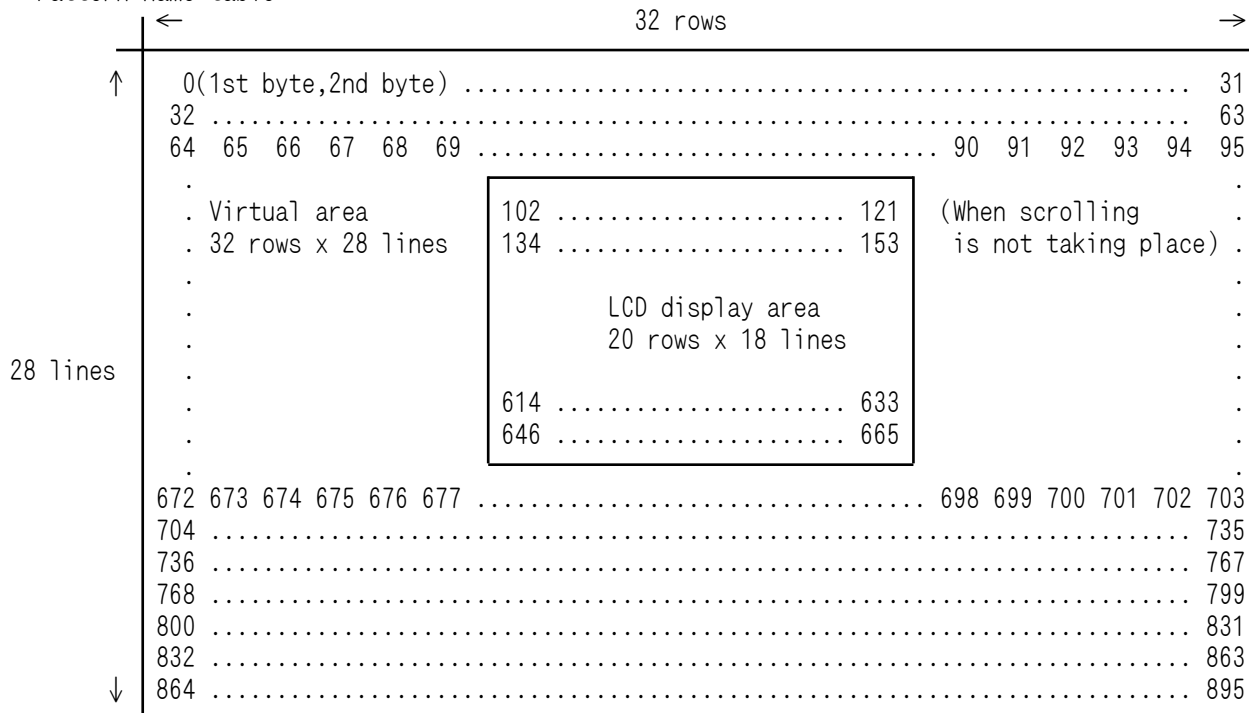
DEFB  036H,0E0H,0FFH,0FFH,0FFH,0FFH,0FFH,000H
DEFB  003H,006H,001H

```



#### ④ Scroll screen display

##### a. Pattern name table



The pattern name table starts from the position determined by register #2. Two bytes correspond to one cell of the scroll screen. These bytes are arranged in the sequence byte 1, byte 2. A pattern name table consisting of  $32 \times 28 \times 2 = 1792$  (700H) bytes is used for the virtual area. Part of this is used for the LCD.

A description is given below of the contents of the two bytes which correspond to each cell.

Byte 1

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0

Byte 1

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
*	*	*	PRI	CPT	RVV	RVH	CH 8

- CH0 to CH8  
Set the No. of the character to be displayed on the cell, in these nine bits.
- RVH  
0: Normal  
1: The cell character pattern is left-right reversed.
- RVV  
0: Normal  
1: The cell character pattern is up-down reversed.
- CPT  
This is the color palette select bit.  
0: Selects color palette 0 (color RAM addresses 00H to 1FH)  
1: Selects color palette 1 (color RAM addresses 20H to 3FH)
- PRI  
0: Normal (The sprites are displayed at the over of the scroll screen.)  
1: When the color code setting for the scroll-screen dots is other than 0, the scroll screen is prioritized over sprites.

○ D7 to D5 of byte 2

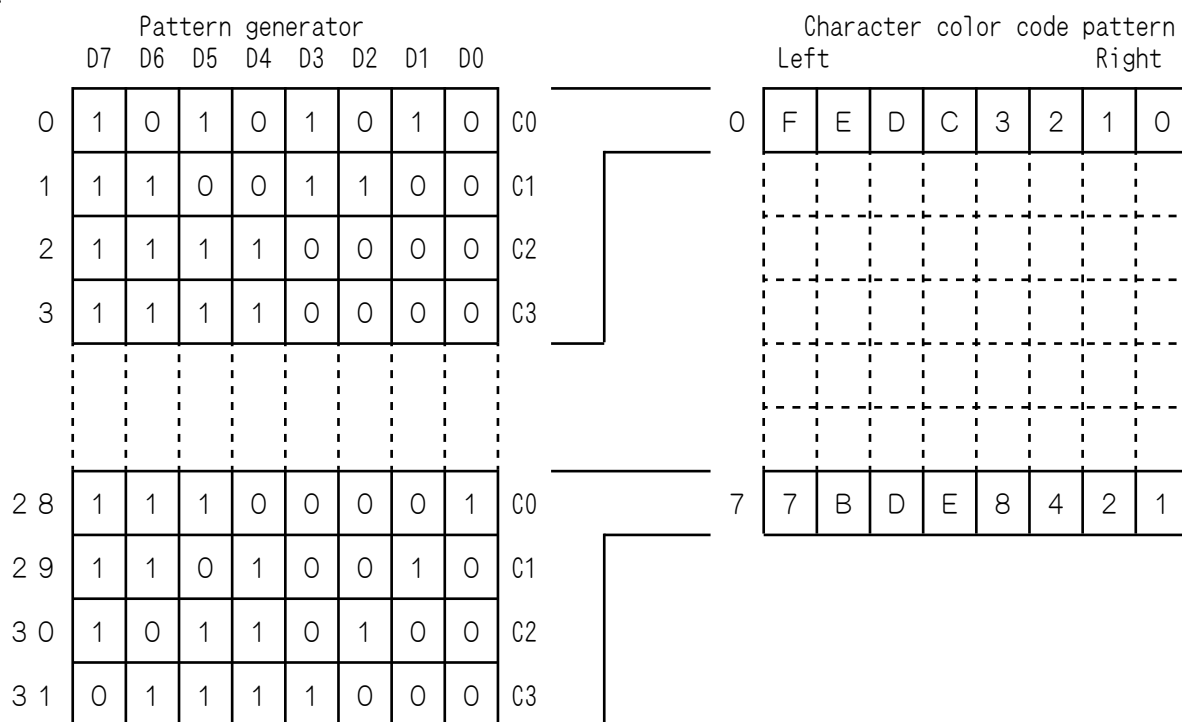
The data in this part is not used in the hardware, hence it can be used in software, such as for flags.

b. Pattern generator table

The pattern generator table always starts from address 0000H. Eight dots in the horizontal direction are represented by four bytes, and one character is represented by  $4 \times 8 = 32$  (20H) bytes.

32 bytes correspond to the color code pattern of the characters, as shown in the example below. (Please be aware that the respective dots constitute a color code which is represented by the four bits C3 to C0.)

Example:



In other words, if an address consisting of 32 bytes is divided as follows,

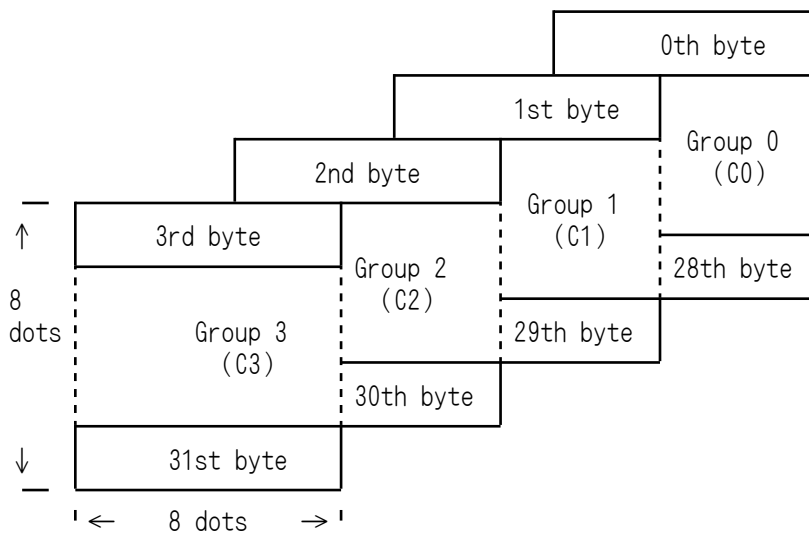
Group 0: 0, 4, 8, 12, 16, 20, 24, 28

Group 1: 1, 5, 9, 13, 17, 21, 25, 29

Group 2: 2, 6, 10, 14, 18, 22, 26, 30

Group 3: 3, 7, 11, 15, 19, 23, 27, 31

A pattern will be obtained in which Group 0 corresponds to the 0th bit of the color code, Group 1 to the 1st bit, Group 2 to the 2nd bit, and Group 3 to the 3rd bit. (These groups can be considered to correspond to four planes.)



Example: Character color code pattern assuming that the following data was input from address 0000H, the pattern of the color code of character 0 will change as follows.

Data					Color code pattern from character 0	
					Left	Right
ADDRESS	+ 0	+ 1	+ 2	+ 3		
0 0 0 0	A A	C C	F 0	F 0	0	F E D C 3 2 1 0
0 0 0 4	0 1	0 2	0 4	0 8	1	0 0 0 0 8 4 2 1
0 0 0 8	1 0	2 0	4 0	8 0	2	8 4 2 1 0 0 0 0
0 0 0 C	0 F	0 F	F 0	F 0	3	C C C C 3 3 3 3
0 0 1 0	0 0	8 1	8 0	0 0	4	6 0 0 0 0 0 0 2
0 0 1 4	3 F	0 0	F 0	0 0	5	4 4 5 5 1 1 1 1
0 0 1 8	5 5	6 6	7 8	8 0	6	8 7 6 5 4 3 2 1
0 0 1 C	E 1	D 2	B 4	7 8	7	7 B D E 8 4 2 1

c. Color RAM

When the color code pattern is determined by the pattern generator table, RGB data will be read from the corresponding color RAM, resulting in a character color pattern. The color RAM has a capacity of 12 bits x 32 words, and the color code expresses A4 to A1 of the color RAM addresses. A5 is determined by the palette. In the case of a scroll screen character, it is determined by the CPT bit of the second byte in the pattern name table. Four bits each of the 12-bit data are assigned to R, G and B, respectively, enabling a total of 4096 colors to be displayed. Thirty two colors from these 4096 colors are set and displayed with the palettes and color codes.

Palette 0

	Color code	Color RAM data
Address	C3 C2 C1 C0	Component
0 0 H	0 0 0 0	Red & Green
0 1 H		Blue
1 E H	1 1 1 1	Red & Green
1 F H		Blue

Palette 1

	Color code	Color RAM data
Address	C3 C2 C1 C0	Component
2 0 H	0 0 0 0	Red & Green
2 1 H		Blue
3 E H	1 1 1 1	Red & Green
3 F H		Blue

The relationship between the color RAM data and color is shown below.

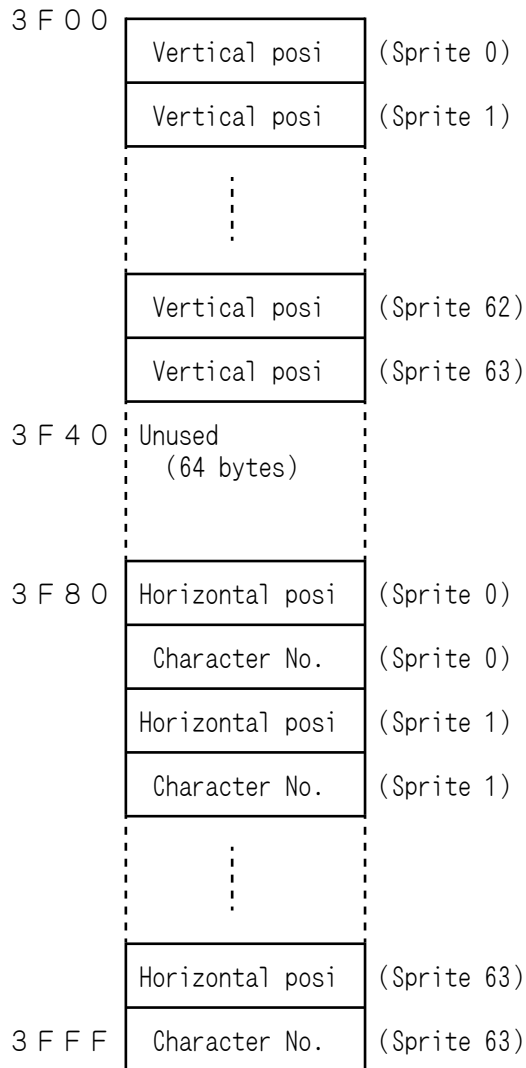
Even addresses	Odd addresses	:	Color
00H	00H	:	Black
0FH	00H	:	Bright red
F0H	00H	:	Bright green
00H	0FH	:	Bright blue
FFH	0FH	:	White

## ⑤ Displaying sprites

### a. Sprite attribute table

A maximum of 64 sprites, each defined by vertical position, horizontal position and character No., can be displayed, hence the sprite attribute table uses  $3 \times 64 = 192$  bytes. An actual sprite attribute table consists of an area of 256 bytes, 64 bytes of which are unused. At a vertical position, D0H has the meaning of an end code, hence if D0H is written to a vertical position, the display of all subsequent sprites will be disabled. To prevent a particular sprite from being displayed, set E0H in the corresponding vertical position.

When the base address of the sprite attribute table is address 3F00H



※ Do not use the unused 64 bytes as a character generator table.

### b. Sprite generator table

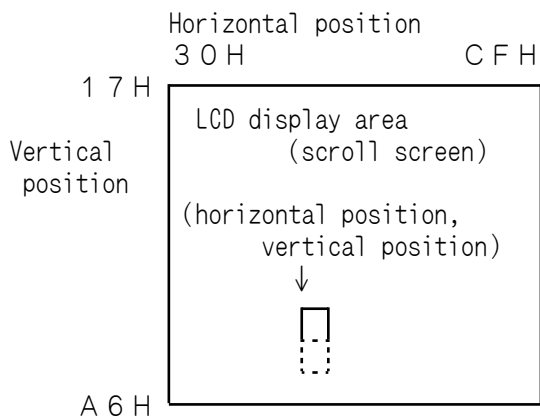
The base address of the sprite generator table is determined by register #6. Apart from this, the sprite generator functions in the same way as the pattern generator table. Here too, each 20H is allocated to one sprite character.

### c. Color RAM

The color RAM is treated in exactly the same way as the scroll screen except for the fact that the palette on the 1 side is always selected. The part corresponding to color code 0 is transparent, even if color data is set in it.



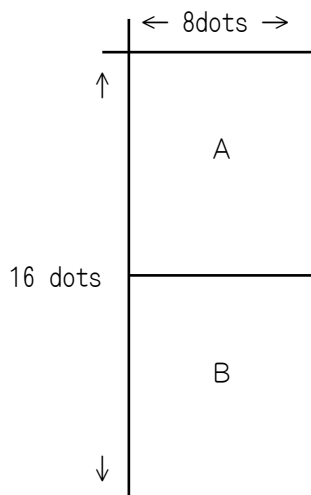
d. Sprite coordinates



The coordinate system of a sprite is as shown in the figure at left. It is displayed so that the dot at the top left of the sprite is located at the top (horizontal position, vertical position). This also applies for a size of 8 x 16 dots. When the horizontal position is outside the range 30H to CFH, the sprite exists in the effective area. Also, in the case of the vertical position, the sprite exists in the effective area so long as it is in the range FFH to 16H and A7 to BEH.

e. SIZE bit

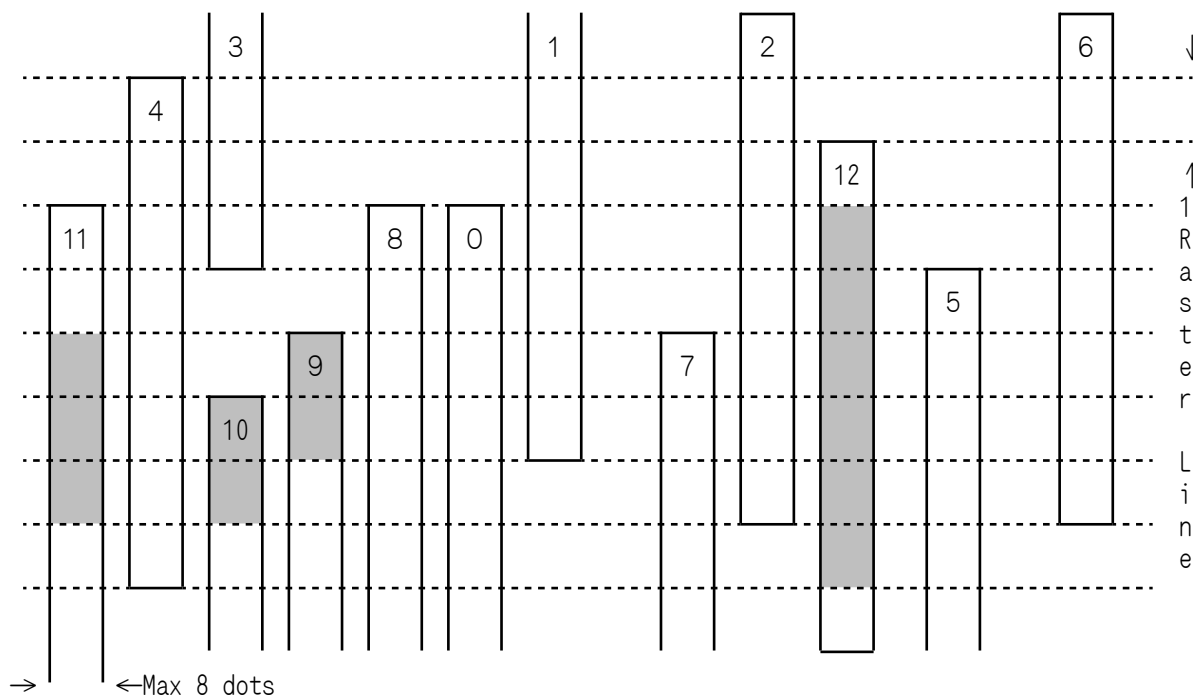
If "1" is set in the SIZE bit, an 8 x 16 dot (vertical length) sprite will be displayed. In this case, the top seven bits of the character No. set in the sprite attribute table will be effective. If the number of character No. is b7b6b5b4b3b2b10 (or b7b6b5b4b3b2b11; b0 is ineffective), the character b7b6b5b4b3b2b10 will be displayed in the position of A, and the character of b7b6b5b4b3b2b11 in the position of B.




f. Sprite display limits

- Up to eight sprites can be displayed on the same horizontal line. This limit also applies if there are sprites in the left and right effective areas, even if they are not displayed in the LCD display area.
- Sprite 0 has the highest priority, and sprite 63 the lowest priority.

○ If two sprites overlap each other, the sprite with the higher priority will be displayed. Also, if there are nine or more sprites on the same horizontal line, the eight sprites with the highest priority will be displayed, and the ninth and higher sprites will not be displayed. This judgment of priority display is made for each line.



 The shaded areas become transparent, and the background pattern is displayed.

#### ⑥ H counter, V counter

The VDP displays an image based on the H counter and the V counter. The H and V counter values can also be read from the CPU. The value of the H counter is effective only when a special clock pulse.

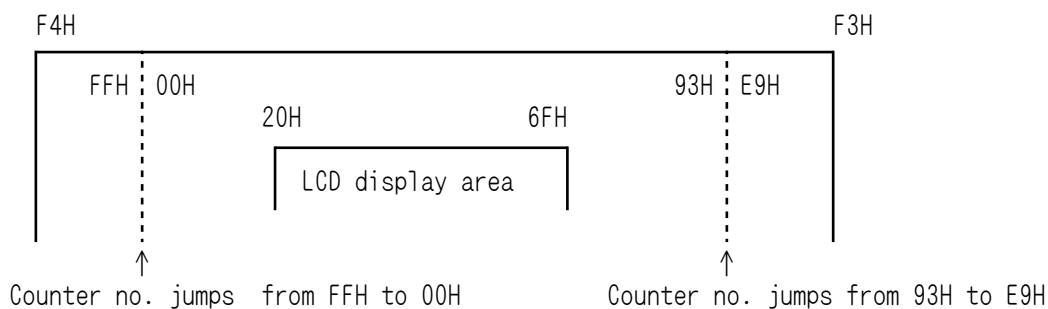
The H counter corresponds to dots, and the V counter corresponds to lines. Both are 9-bit counters. The CPU reads the top eight bits of the H counter, and reads the bottom eight bits of the V counter.

These data can be read at any time.

##### a. H counter

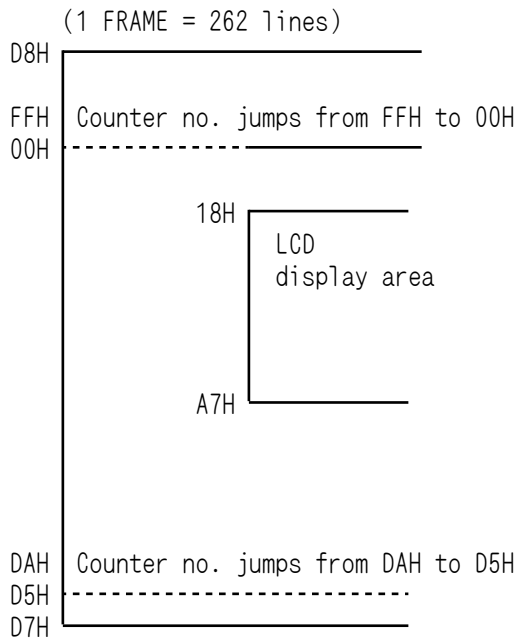
Reads I/O port 7FH (PSG control in the case of a write operation).

Two dots are equivalent to one count, and three counts are equivalent to four CPU clock pulses. (1H = 342 dots = 171 counts = 228 CPU clock pulses)



※ The generation of an interrupt at an arbitrary vertical position and the count-up of the V counter take place in the case of F4H.

- b. V counter
  - Reads I/O port 7EH
  - One line is equivalent to one count.



※ The interrupt for the completion of the effective area is generated by C0H (and F4H in the case of the H counter). The FFH line is line 511 described before.

VDP manual END

# PSG Manual

The PSG (Programmable Sound Generator) contains three tone generators and one noise generator. Each of the tone and noise generators can be distributed left and right, enabling a pseudo stereo effect to be generated. (See "System Control Port".)

Control of the PSG itself, which is described below, is performed by means of the write operation to I/O area 7FH.

The basic clock is 3.579545 MHz. The data to be sent from the CPU is immediately latched in the PSG, hence there is no need for a wait. The sound output goes OFF in the case of a power-on reset. Design the software so that the output goes OFF at the beginning of the program as well.

## [1] Tone generator

Each tone generator consists of a frequency setting section (programmable counter) and a level setting section (programmable attenuator).

### (1) Method of calculating the 10-bit frequency division ratio n

At the frequency setting section, the basic clock is frequency-divided to 1/32. This is further frequency divided by the tone counter set by the 10 bits F9 (MSB: top bit) to F0 (LSB: bottom bit).

Consequently, the basic clock frequency is divided by 32, then the desired frequency can be output by setting the value obtained by dividing the frequency-divided clock by the desired frequency in F9 to F0.

$$n = N / (32 \times f)$$

Where n = 10-bit frequency division ratio (F9 to F0)

N = Basic clock

f = Desired frequency

### (2) Tone frequency setting

Set the 10-bit frequency division ratio (F9 to F0) in the tone counter in order to obtain the desired frequency. The 1st and 2nd bytes are identified by means of the top bit.

1st byte

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

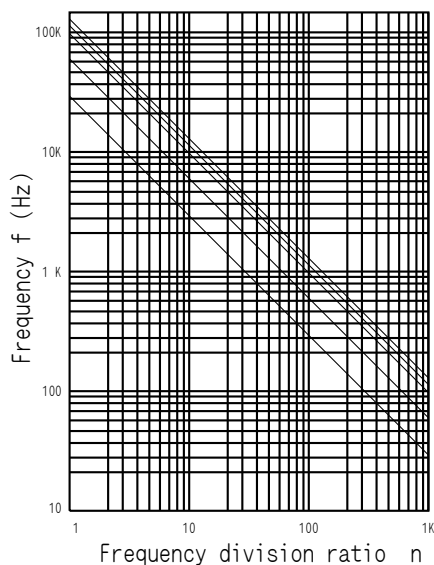
*	REG. ADDR.	n					
1	R2	R1	R0	F3	F2	F1	F0

R2	R1	R0	Control register allocation	1st
0	0	0	Tone generator 1	8 ×
0	1	0	Tone generator 2	A ×
1	0	0	Tone generator 3	C ×

2nd byte

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

*		n					
0	×	F9	F8	F7	F6	F5	F4



Frequency division ratio with respect to the output frequency

### (3) Example of frequency setting

Consider an example in which the basic clock frequency is 3.579545 MHz and the desired frequency of 440 Hz is output from TONE 1. (corresponding to "A" on the musical scale)

#### a. Calculation of frequency division ratio n

$$\begin{aligned} n &= N/(32 \times f) \\ &= 3579545/(32 \times 440) \\ &\doteq 254.229 \end{aligned}$$

n is a 10-bit integer, hence the nearest integral value is 254.

Consequently, the frequency actually output is

$$\begin{aligned} f &= N/(32 \times n) \\ &= 3579545/(32 \times 254) \\ &\doteq 440.397 \text{ (Hz)} \end{aligned}$$

Here, the pitch error  $\Delta C$  is obtained according to the following equation.

$$\begin{aligned} \Delta C &= \{(f' - f)/f\} / ({}^{1200}\sqrt{2} - 1) \\ &\doteq \{(440.397 - 440)/440\} / ({}^{1200}\sqrt{2} - 1) \\ &\doteq (0.397/440) / 0.000578 \doteq 1.56 \end{aligned}$$

f: True frequency f': Actual frequency  ${}^{1200}\sqrt{2}$

#### b. Data sent to PSG

$$\begin{aligned} n &= 254 \\ &= 0011111110B \end{aligned}$$

1st byte

*	REG. ADDR.			n			
1	R2	R1	R0	F3	F2	F1	F0
D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	1	1	0

2nd byte

*		n					
0	×	F9	F8	F7	F6	F5	F4
D7	D6	D5	D4	D3	D2	D1	D0
0	×	0	0	1	1	1	1

### (4) Tone level setting

The frequency set by the tone generator is sent to the level setting section where the volume level is set. The level setting section is a programmable attenuator which enables the volume level to be set in 16 steps from 0 dB to 0FF according to a 4-bit attenuation value.

1st byte only

D7	D6	D5	D4	D3	D2	D1	D0
*	REG. ADDR.			ATT. DATA			
1	R2	R1	R0	A3	A2	A1	A0

R2	R1	R0	Control register allocation	HEX
0	0	1	Tone 1 attenuation	9×
0	1	1	Tone 2 attenuation	B×
1	0	1	Tone 3 attenuation	D×

## Attenuation

[db]	A3 A2 A1 A0	HEX	[db]	A3 A2 A1 A0	HEX
0	0 0 0 0	× 0	1 6	1 0 0 0	× 8
2	0 0 0 1	× 1	1 8	1 0 0 1	× 9
4	0 0 1 0	× 2	2 0	1 0 1 0	× A
6	0 0 1 1	× 3	2 2	1 0 1 1	× B
8	0 1 0 0	× 4	2 4	1 1 0 0	× C
1 0	0 1 0 1	× 5	2 6	1 1 0 1	× D
1 2	0 1 1 0	× 6	2 8	1 1 1 0	× E
1 4	0 1 1 1	× 7	OFF	1 1 1 1	× F

## [2] Noise generator

The noise generator consists of a noise generator circuit and a level setting section. The source of the noise supplied from the noise generator circuit is a shift register with EX-OR feedback. Each time the noise control register changes, the shift register is cleared.

The shift clock of this shift register is determined by four modes that are in turn determined by NF0 and NF1. If NF0 = NF1 = 0, for example, the shift clock becomes  $(N/32)/16$ . In this case, if FB = 0, this shift clock will be frequency-divided by 16, resulting in synchronous noise of a frequency of  $N/(32 \times 16 \times 16)$ . If FB = 1, the shift register will be driven by this shift clock with EX-OR feedback, resulting in the generation of white noise.

### (1) Noise generator circuit control

1st byte only

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

*	REG. ADDR.				SHIFT		
1	1	1	0	×	FB	NF1	NF0

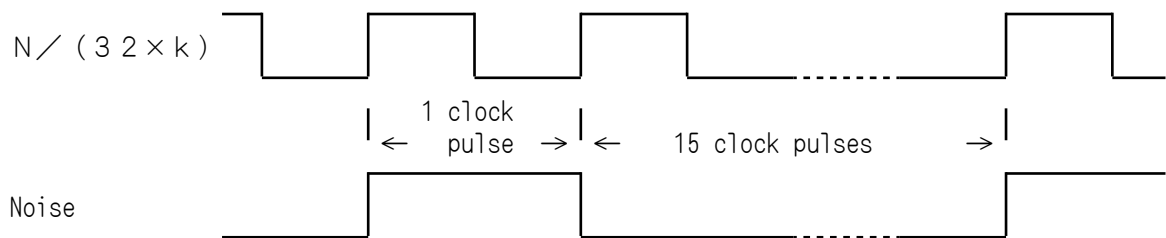
FB	Noise Generation
0	Synchronous Noise
1	White noise

NF1	NF0	Shift clock	k
0	0	$(N/32)/k$	1 6
0	1	$(N/32)/k$	3 2
1	0	$(N/32)/k$	6 4
1	1	Tone generator 3	

N : Basic clock

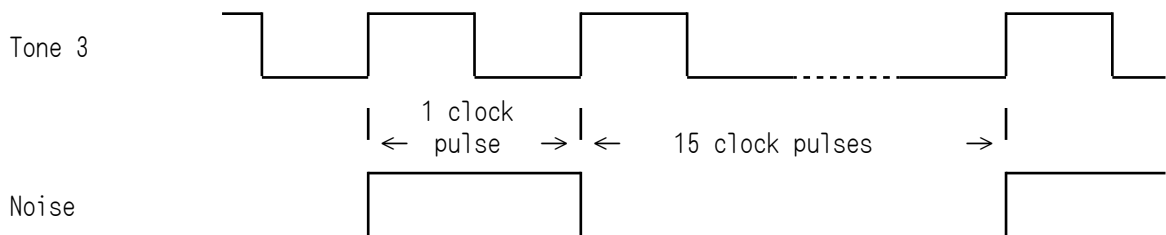
←At this time, the tone of the noise can be varied continuously

- a. Synchronous noise (FB = 0)  
When NF0 = NF1 = Value other than 1



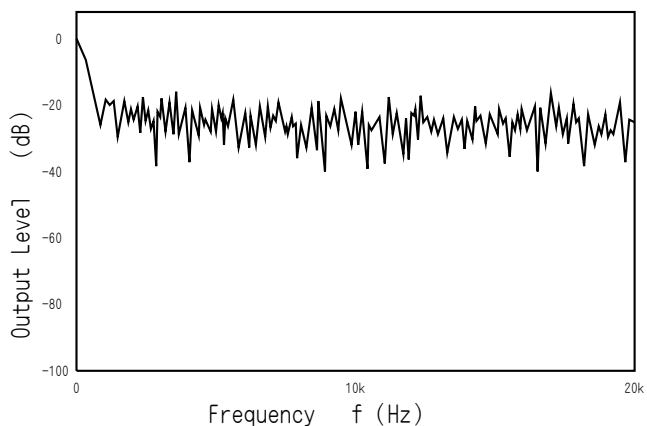
\* The noise frequency is  $(N/(32 \times k))/16 = N/(32 \times k \times 16)$

When NF0 = NF1 = 1 (Control by tone 3)



\* The noise frequency is  $(\text{frequency TONE 3})/16 = N/(32 \times n \times 16)$

- b. White Noise (FB = 1)  
Spectrum when NF0 = 0 NF1 = 1 n=1



- (2) Noise level setting  
1st byte only

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

*	REG. ADDR.				ATT. DATA		
1	1	1	1	A3	A2	A1	A0


\* The attenuation control is the same as for "Tone".

### [3] Register address feed

PSG uses the three bits R2 to R0 of the 1st byte to judge which control register the data has been sent from.

R2	R1	R0	Control register allocation	1st
0	0	0	Tone 1 frequency division ratio	8 ×
0	0	1	Tone 1 attenuation	9 ×
0	1	0	Tone 2 frequency division ratio	A ×
0	1	1	Tone 2 attenuation	B ×
1	0	0	Tone 3 frequency division ratio	C ×
1	0	1	Tone 3 attenuation	D ×
1	1	0	Noise generator circuit control	E ×
1	1	1	Noise attenuation	F ×

### [4] Correlation between the sound elements and PSG

Sound element	Physical element	Correlation with PSG
Pitch of sound	Frequency	[1] - (2) Tone frequency setting, [2] - (1) Noise generator circuit control
Tone	Harmonic components	This is mainly related to wave length. In the tone generation mode, the PSG can output three frequencies simultaneously from only a 50% duty pulse waveform. Consequently, by combining attenuation control with this mode, the harmonic components can be controlled. In the synchronous noise mode, a 6.25% duty pulse waveform.
Strength of tone	Amplitude	As described in [1] - (3) and [2] - (2), the attenuation of the three tones and noise can be controlled by 4-bit data.
Way in which sound is emitted.	Envelope 	The wave length at left can be realized by using external data to control each attenuation. This can be done in the range where the 4-bit attenuation data is rewritten and envelope sequence control performed at each step. The tone and noise frequencies can be controlled by the range in which component control can be performed.

(next page)

Relation between musical interval and frequency division ratio for scale divided equally into 12 parts (basic clock: 3.579545 MHz)

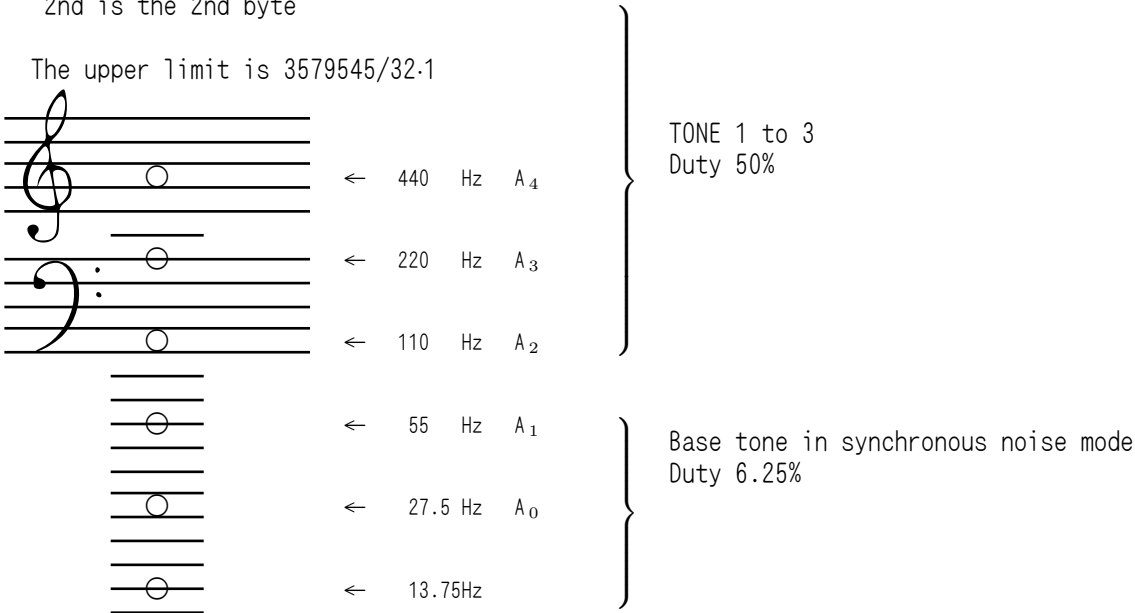


Musical interval	Frequency division ratio	H E X		PSG output [Hz]	Actual frequency [Hz]
		1 st	2 nd		
A 2	1 0 1 7	X 9	3 F	1 0 9 . 9 9 1	1 1 0 . 0 0 0
A #2	9 6 0	X 0	3 C	1 1 6 . 5 2 2	1 1 6 . 5 4 1
B 2	9 0 6	X A	3 8	1 2 3 . 4 6 7	1 2 3 . 4 7 1
C 3	8 5 5	X 7	3 5	1 3 0 . 8 3 2	1 3 0 . 8 1 3
C #3	8 0 7	X 7	3 2	1 3 8 . 6 1 3	1 3 8 . 5 9 1
D 3	7 6 2	X A	2 F	1 4 6 . 7 9 9	1 4 6 . 8 3 2
D #3	7 1 9	X F	2 C	1 5 5 . 5 7 8	1 5 5 . 5 6 3
E 3	6 7 9	X 7	2 A	1 6 4 . 7 4 4	1 6 4 . 8 1 4
F 3	6 4 1	X 1	2 8	1 7 4 . 5 1 0	1 7 4 . 6 1 4
F #3	6 0 5	X D	2 5	1 8 4 . 8 9 4	1 8 4 . 9 9 7
G 3	5 7 1	X B	2 3	1 9 5 . 9 0 4	1 9 5 . 9 9 8
G #3	5 3 9	X B	2 1	2 0 7 . 5 3 4	2 0 7 . 6 5 2
A 3	5 0 8	X C	1 F	2 2 0 . 1 9 9	2 2 0 . 0 0 0
A #3	4 8 0	X 0	1 E	2 3 3 . 0 4 4	2 3 3 . 0 8 2
B 3	4 5 3	X 5	1 C	2 4 6 . 9 3 4	2 4 6 . 9 4 2
C 4	4 2 8	X C	1 A	2 6 1 . 3 5 7	2 6 1 . 6 2 6
C #4	4 0 4	X 4	1 9	2 7 6 . 8 8 4	2 7 7 . 1 8 3
D 4	3 8 1	X D	1 7	2 9 3 . 5 9 8	2 9 3 . 6 6 5
D #4	3 6 0	X 8	1 6	3 1 0 . 7 2 5	3 1 1 . 1 2 7
E 4	3 3 9	X 3	1 5	3 2 9 . 9 7 3	3 2 9 . 6 2 8
F 4	3 2 0	X 0	1 4	3 4 9 . 5 6 5	3 4 9 . 2 2 8
F #4	3 0 2	X E	1 2	3 7 0 . 4 0 0	3 6 9 . 9 9 4
G 4	2 8 5	X D	1 1	3 9 2 . 4 9 5	3 9 1 . 9 9 5
G #4	2 6 9	X D	1 0	4 1 5 . 8 4 0	4 1 5 . 3 0 5
A 4	2 5 4	X E	0 F	4 4 0 . 3 9 7	4 4 0 . 0 0 0
A #4	2 4 0	X 0	0 F	4 6 6 . 0 8 7	4 6 6 . 1 6 4
B 4	2 2 6	X 2	0 E	4 9 4 . 9 6 0	4 9 3 . 8 8 3
C 5	2 1 4	X 6	0 D	5 2 2 . 7 1 5	5 2 3 . 2 5 1
C #5	2 0 2	X A	0 C	5 5 3 . 7 6 7	5 5 4 . 3 6 5
D 5	1 9 0	X E	0 B	5 8 8 . 7 4 2	5 8 7 . 3 3 0
D #5	1 8 0	X 4	0 B	6 2 1 . 4 5 0	6 2 2 . 2 5 4
E 5	1 7 0	X A	0 A	6 5 8 . 0 0 5	6 5 9 . 2 5 5
F 5	1 6 0	X 0	0 A	6 9 9 . 1 3 1	6 9 8 . 4 5 6
F #5	1 5 1	X 7	0 9	7 4 0 . 8 0 1	7 3 9 . 9 8 9
G 5	1 4 3	X F	0 8	7 8 2 . 2 4 4	7 8 3 . 9 9 1
G #5	1 3 5	X 7	0 8	8 2 8 . 6 0 0	8 3 0 . 6 0 9
A 5	1 2 7	X F	0 7	8 8 0 . 7 9 5	8 8 0 . 0 0 0
A #5	1 2 0	X 8	0 7	9 3 2 . 1 7 4	9 3 2 . 3 2 8
B 5	1 1 3	X 1	0 7	9 8 9 . 9 2 0	9 8 7 . 7 6 7
C 6	1 0 7	X B	0 6	1 0 4 5 . 4 2 9	1 0 4 6 . 5 0 2
C #6	1 0 1	X 5	0 6	1 1 0 7 . 5 3 4	1 1 0 8 . 7 3 1
D 6	9 5	X F	0 5	1 1 7 7 . 4 8 4	1 1 7 4 . 6 5 9
D #6	9 0	X A	0 5	1 2 4 2 . 8 9 9	1 2 4 4 . 5 0 8
E 6	8 5	X 5	0 5	1 3 1 6 . 0 1 1	1 3 1 8 . 5 1 0
F 6	8 0	X 0	0 5	1 3 9 8 . 2 6 2	1 3 9 6 . 9 1 3
F #6	7 6	X C	0 4	1 4 7 1 . 8 5 4	1 4 7 9 . 9 7 8
G 6	7 1	X 7	0 4	1 5 7 5 . 5 0 6	1 5 6 7 . 9 8 2
G #6	6 7	X 3	0 4	1 6 6 9 . 5 6 6	1 6 6 1 . 2 1 9
A 6	6 4	X 0	0 4	1 7 4 7 . 8 2 7	1 7 6 0 . 0 0 0
A #6	6 0	X C	0 3	1 8 6 4 . 3 4 9	1 8 6 4 . 6 5 5
B 6	5 7	X 9	0 3	1 9 6 2 . 4 7 3	1 9 7 5 . 5 3 3

- \*  $f = 3579545 / (32 \times n)$  n: 10-bit frequency division Maximum frequency is  $n = 1$
- \* In the previous table, musical interval was calculated on the basis of 440 Hz as concert pitch.
- \* Regarding HEX, the X part is as follows:

Tone 1 → 8  
 Tone 2 → A  
 Tone 3 → C  
 1st is the 1st byte  
 2nd is the 2nd byte

The upper limit is  $3579545 / 32.1$



If a frequency of no greater than that generated by the tone generator is output, the outputs shown in the table below will be obtained due to the synchronous noise mode of the noise generator section. (Basic clock: 3.579545 MHz)

Musical interval	Frequency division ratio	H E X (TONE3)		PSG output [Hz]	Actual frequency [Hz]
		1 st	2 nd		
C 0	4 2 8	C C	1 A	1 6 . 3 3 5	1 6 . 3 5 2
C #0	4 0 4	C 4	1 9	1 7 . 3 0 5	1 7 . 3 2 4
D 0	3 8 1	C D	1 7	1 8 . 3 5 0	1 8 . 3 5 4
D #0	3 6 0	C 8	1 6	1 9 . 4 2 0	1 9 . 4 4 5
E 0	3 3 9	C 3	1 5	2 0 . 6 2 3	2 0 . 6 0 2
F 0	3 2 0	C 0	1 4	2 1 . 8 4 8	2 1 . 8 2 7
F #0	3 0 2	C E	1 2	2 3 . 1 5 0	2 3 . 1 2 5
G 0	2 8 5	C D	1 1	2 4 . 5 3 1	2 4 . 5 0 0
G #0	2 6 9	C D	1 0	2 5 . 9 9 0	2 5 . 9 5 7
A 0	2 5 4	C E	0 F	2 7 . 5 2 5	2 7 . 5 0 0
A #0	2 4 0	C 0	0 F	2 9 . 1 3 0	2 9 . 1 3 5
B 0	2 2 6	C 2	0 E	3 0 . 9 3 5	3 0 . 8 6 8
C 1	2 1 4	C 6	0 D	3 2 . 6 7 0	3 2 . 7 0 3
C #1	2 0 2	C A	0 C	3 4 . 6 1 0	3 4 . 6 4 8
D 1	1 9 0	C E	0 B	3 6 . 7 9 6	3 6 . 7 0 8
D #1	1 8 0	C 4	0 B	3 8 . 8 4 1	3 8 . 8 9 1
E 1	1 7 0	C A	0 A	4 1 . 1 2 5	4 1 . 2 0 3
F 1	1 6 0	C 0	0 A	4 3 . 6 9 6	4 3 . 6 5 4
F #1	1 5 1	C 7	0 9	4 6 . 3 0 0	4 6 . 2 4 9
G 1	1 4 3	C F	0 8	4 8 . 8 9 0	4 8 . 9 9 9
G #1	1 3 5	C 7	0 8	5 1 . 7 8 7	5 1 . 9 1 3
A 1	1 2 7	C F	0 7	5 5 . 0 5 0	5 5 . 0 0 0
A #1	1 2 0	C 8	0 7	5 8 . 2 6 1	5 8 . 2 7 0
B 1	1 1 3	C 1	0 7	6 1 . 8 7 0	6 1 . 7 3 5
C 2	1 0 7	C B	0 6	6 5 . 3 3 9	6 5 . 4 0 6
C #2	1 0 1	C 5	0 6	6 9 . 2 2 1	6 9 . 2 9 6
D 2	9 5	C F	0 5	7 3 . 5 9 3	7 3 . 4 1 6
D #2	9 0	C A	0 5	7 7 . 6 8 1	7 7 . 7 8 2
E 2	8 5	C 5	0 5	8 2 . 2 5 1	8 2 . 4 0 7
F 2	8 0	C 0	0 5	8 7 . 3 9 1	8 7 . 3 0 7
F #2	7 6	C C	0 4	9 1 . 9 9 1	9 2 . 4 9 9
G 2	7 1	C 7	0 4	9 8 . 4 6 9	9 7 . 9 9 9
G #2	6 7	C 3	0 4	1 0 4 . 3 4 8	1 0 3 . 8 2 6

\* Set E3H for control of the noise generator circuit (synchronous noise mode).  
 $f = \text{TONE 3 frequency} / 16 = 3579545 / (32 \times n \times 16)$

PSG Manual END