

Confidential



Neo-Geo Hardware Specification

SNK

Neo-Geo Hardware Specification

Table of Contents

Neo-Geo Specification	Hardware - 1
Special Features of the "3D-LINE SPRITE"	Hardware - 3
Specification of Each Function	Hardware - 4
FIX	Hardware - 4
Background	Hardware - 4
3D-Line Sprite	Hardware - 4
Interrupts	Hardware - 5
Interrupt-1	Hardware - 5
Interrupt-2	Hardware - 5
Access to Line Sprite Controller (LSPC)	Hardware - 6
Address Map of the 68000	Hardware - 8
Address Map of the Z80	Hardware - 10
I/O Map of the Z80	Hardware - 10
Sound Function	Hardware - 10
Notes	Hardware - 10
3D Line Sprite	Hardware - 12
Vertical and Horizontal Positions	Hardware - 13
Example of the Number of Active Characters (ACT), Vertical Reduction (BIGV), and Horizontal Reduction (BIGH)	Hardware - 14
Address Mapping of the FIX Area (VRAM) (In the NTSC Mode)	Hardware - 15
Address Mapping of the FIX Area (VRAM) (In the PAL Mode)	Hardware - 16
Address Example of the 3D-Line Sprite	Hardware - 17
Address Mapping of the 3D-Line Sprite	Hardware - 18
Color Palettes	Hardware - 19
Data Format of the 3D-Line Sprite Character ROM	Hardware - 20
Data Format of the FIX Area Character ROM	Hardware - 21

Neo-Geo Specification

Dec. 1, 1989

Aug. 22, 1990

Jun. 18, 1991

- [1] CPU HD68HC000PS12 Clock: 12 MHz
- [2] Program ROM 2M Bytes (Maximum)
- [3] Work RAM 64K Bytes
- [4] Fix character 128K Bytes
- [5] 3D-line sprite character 128M Bits (Maximum)
- [6] VRAM 64K Bytes + 4K Bytes
- [7] Background 1 Color
- [8] Color 4096 colors out of 65536 colors simultaneous color display
(15 colors x 256 palettes)
16k bytes
- [9] RGB output, video output, and headphone output (stereo)
- [10] Can use memory card
- [11] Interrupts Vertical blanking interrupt and timer interrupt

[12] Sound

- | | |
|-----------------------|-------------|
| 1) 1 | YM-2610 |
| 2) 62K Bytes | Program |
| 3) 2K Bytes | Working RAM |
| 4) 16M Bytes (max.) | ADPCM-A |
| 5) 16M Bytes (max.) | ADPCM-B |
| 6) Z80A (clock 4 MHz) | CPU |

[13] The screen is displayed using LSPC or LSPC2.

When using LSPC2, switching between NTSC and PAL becomes possible through hardware (with software they can only be read).

In the PAL mode, the display area becomes 16 lines larger on the top and bottom; the vertical blanking lengthens only by 16 lines.

Special features of the "3D-LINE SPRITE"

1. One 3D-LINE SPRITE consists of 32 characters arranged vertically. Each character is a 16 by 16 dot array.
2. With 380 3D-LINE SPRITES, many enemies and ammunition can appear on the screen. Large characters can be easily displayed, and parts may be overwritten.
3. Characters are more freely distributed than are those produced using boards which have separate scroll and sprite patterns.
4. The image can be reduced in 256 levels vertically, and 16 levels horizontally.
5. The display range may be changed in steps (0, 1, 2, 3, ... 16, 32, 33) by specifying the number of active characters.
6. A 3D scrolling display is formed by forty-one 3D-LINE SPRITES.
In this configuration, set the number of active characters to 33. The picture size can be reduced up to half both vertically and horizontally.

Note: When the number of active characters is set to 33, the picture can be reduced by up to half vertically, and both ends of the picture will be connected together as a loop.

7. A maximum of 96 3D-LINE SPRITES can be placed horizontally.
(The arrangement is determined only by the vertical position and number of active characters.)
8. Various scenes such as rivers, are easily displayed by the automatic character switching display function.
(Switching character number bits 0-2)
9. Multiple 3D-LINE SPRITES can be moved simultaneously using the chain function.
10. Interrupts can be issued at specified intervals using timer interrupt, or when the scanning reaches an arbitrary point on the screen.
11. The display can be dimmed by using the shadow bit output.

Specification of Each Function

1) FIX ;Priority 1

1. 40 by 28 characters are displayed at a fixed location.
2. The size of each character is 8 by 8 pixels.
3. Up to 15 colors can be used per character.
4. There are 16 color pallettes. (4 bits)
5. The number of characters is 4096. (12 bits)

2) Background ;Priority 3

One color is specified from 65,536 colors.

3) 3D-LINE SPRITE ;Priority 2

1. Up to 380 3D-LINE SPRITES can be displayed simultaneously, but only 96 can be placed horizontally.
2. One 3D-LINE SPRITE consists of 32 vertical characters; each character being a 16 by 16 dot array.
3. The following parameters are specified for each 3D-LINE SPRITE:
 - 1] Vertical position (9 bits)
 - 2] Horizontal position (9 bits)
 - 3] Vertical reduction (8 bits) ;0FFH for maximum
 - 4] Horizontal reduction (4 bits);0FH for maximum
 - 5] Number of active characters (6 bits);No characters are displayed
;when this is set to 0.
;The product of 16 times this
;number will be the total
;number of dots that are
;displayed, except when "33"
;is selected.
 - 6] Chain bit (1 bit);When this bit is set to "1,"
;this 3D-LINE SPRITE is
;connected to the right side of
;the previous 3D-LINE
;SPRITE, and vertical
;position, horizontal position,
;vertical reduction, and
;number of active characters
;settings are ignored.

4. The following parameters are set for each character.
 - 1] Character number (16 bits)
 - 2] Color palette number (8 bits)
 - 3] Vertical flip (1 bit) ;"1" flips image vertically.
 - 4] Horizontal flip (1 bit) ;"1" flips image horizontally.
 - 5] Automatic character switching display of 4-sequence characters. (1 bit)
;"1" for active.
 - 6] Automatic character switching display of 8-sequence characters. (1 bit)
;"1" for active.

5. Specify the speed of the automatic character switching display. (8 bits)

;This value controls the speed
;for both 4-sequence and
;8-sequence characters.

6. Total number of characters. (65,536 is the maximum.)

7. Fifteen colors can be used per character.

8. There are 239 color palettes.

4) Interrupts

1. Interrupt-1
 - 1] Interrupt-vector address is 64H.
 - 2] Interrupt is triggered at the beginning of vertical blanking.
 - 3] Execute the following instructions when the interrupt occurs:
 MOV.W #4, 3C000CH ;For the next interrupt
 MOV.B D0, 300001H ;For the watchdog

2. Interrupt-2
 - 1] Interrupt vector address is 68H.
 - 2] Interrupt is triggered when the timer counter reaches 0.
 - 3] The following instruction is executed when the interrupt occurs:
 MOVE.W #2, 3C000CH

 - 4] After the CPU sets the Timer High register and Timer Low register, that data is set to the timer counter (32 bits) at the moment that the following events listed below occur. (The value of the timer counter is decremented by one every 167nS which is the time it takes to scan one pixel, and an interrupt is triggered when it reaches 0.)

- (1) When the Timer Low register is set.
 - (2) At the beginning of the horizontal blanking in the first line of vertical blanking.
 - (3) When the timer counter reaches 0.
- 5] User can set bits 4 to 7 of the Mode register to disable interrupts, or to change the initialization timing for the timer counter.
- 6] Timer High and Timer Low registers should never be set to 0.
- 7] To trigger interrupts for every N pixels, set the Timer register to N-1, and set bits 4 to 7 of the Mode register to 1001B⁴ (9 x H).
- 8] To trigger interrupts when the scanning reaches multiple arbitrary display locations, set bits 4 to 7 of the Mode register to 1101B (d x H). Then, in the interrupt handler routine, set the interval between the next interrupt and the following to the Timer register.

5) Access to Line Sprite Controller (LSPC)

1. Read and write operations to VRAM should be done via the LSPC register.
2. VRAM has one address for each word (16 bits), and all read/write operations are done in words. (Long words, and bytes are not allowed.)
3. LSPC has the following registers:
 - 1] Address register (write-only)
Contains the VRAM address (0 ~ 0FFFFH) of the next read/write operation.
 - 2] Write Data register (write-only)
Contains data to be transferred to VRAM.
When the CPU writes data to this register, it is passed to VRAM.
 - 3] Automatic Increment register (read/write)
Values range between 0 and 0FFFH.
This value is added to the Address register immediately after the data is written to the Write Data register.
 - 4] Mode Register

Bits 8-15:	Speed for automatic character switching display. Multiply value times 16mS for the actual timing interval.
Bit 3 = 1:	Stop automatic character switching display.
Bit 4 = 1:	Issues an interrupt when the timer counter reaches 0.
Bit 5 = 1:	When Timer Low register is set, Timer counter is simultaneously set.
Bit 6 = 1:	Timer counter is set at the beginning of the horizontal blanking of the first vertical blanking line.
Bit 7 = 1:	Timer counter is set to initial value when Timer counter becomes 0.

5] Read Register (read-only)

VRAM data is read from this register. (The value of the Address register doesn't change.)

6] Raster Vertical Position register (read only)

Bits 0-2: When the automatic character feature is active, the character number of the character displayed can be read from here.

Bit 3: NTSC/PAL mode select *** LSPC2 only ***
= 0 : PAL
= 1 : NTSC

Bit 7 : Read 1V

Bit 8 : Read 2V

Bit 9 : Read 4V

Bit 10 : Read 8V

Bit 11 : Read 16V

Bit 12 : Read 32V

Bit 13 : Read 64V

Bit 14 : Read 128V

Bit 15 : Read 256V (Set to "1" during display.)

7] Timer High register (write-only)

Upper 16 bits of data to be set to Timer counter.

8] Timer Low register (write-only)

Lower 16 bits of data to be set to Timer counter.

9] Interrupt Clear register (write-only)

Interrupt flags are cleared when bit is set to "1".

Bit 1 = 1: Clears the timer interrupt flag.

Bit 2 = 1: Clears the vertical blanking interrupt flag.

10] Timer Stop Switch (write-only) *** LSPC2 only ***

(Initial Value : Bit 0 = 1)

Bit 0 = 0: Timer counter is not stopped even when the PAL mode is selected.

Bit 0 = 1: Only when PAL mode is selected, timer counter is stopped for 32 horizontal lines (18,432 pixels).

4. Read and write operations to LSPC registers are executed with no wait cycle.
5. Read operation from VRAM should be done at least $1.4\mu\text{S}$ (16 CPU clock cycles) after the address register is changed.
6. When the value in the Address register is changed from the 0 - 7FFFH range to the 8000H - OFFFFH range, or from the 8000H - OFFFFH range to the 0 - 7FFFH range, this value must be written directly into the register, rather than using the automatic increment operation. At least $1.4\mu\text{S}$ (16 CPU clock cycles) is required before writing to the Address register after the Data register is

changed.

7. Write operation to Data or Address register should be done $1\mu\text{S}$ (12 CPU clock cycles) after the write operation is executed to a Data register.

Example :

MOVE D0, [A0]	;A0=3C0000H
MOVE D1, [A1]	;A1=3C0002H
ADDQ #8, D1	;instruction for 4 or more clock cycles
MOVE D1, [A1]	
MOVE #1234H, [A1]	;instruction for 12 or more clock cycles
MOVE #5678H, [A1]	;instruction for 12 or more clock cycles
MOVE #9ABC, [A1]	;instruction for 12 or more clock cycles

6) Address map of the 68000

1. Program ROM	0-0FFFFFH
2. System ROM	0C0000H-0C1FFFFH
3. Work RAM	100000H-10FFFFH
4. LSPC	
1] Address register	3C0000H (write, word)
2] Write Data register	3C0002H (write, word)
3] Auto Increment register	3C0004H (read/write, word)
4] Mode register	3C0006H (write, word)
5] Read Data register	3C0000H (read, word) or 3C0002H
6] Raster Vertical Position register	3C0006H (read, word)
7] Timer High register	3C0008H (write, word)
8] Timer Low register	3C000AH (write, word)
9] Interrupt Clear register	3C000CH (write, word)
10] Timer Stop Switch	3C000EH (write, word) **** LSPC2 only ****
5. Color Palette (2 banks)	400000H-401FFFFH (word-long)
6. Watchdog Timer	300001H (write, BYTE)
7. Output Port (write, byte)	
1] Set shadow bit to 0 for normal display	3A0001H
2] Set shadow bit to 1 for dimmed display	3A0011H
3] Vector switch (0-7FH) ON	3A0003H (Switch between 0-7FH and 0C00000H-0C0007FH)
4] Vector switch (0-7FH) OFF	3A0013H
5] Memory card 1/write enable	3A0005H

6] Memory card 2/write enable	3A0017H
7] Memory card 1/write disable	3A0015H
8] Memory card 2/write disable	3A0007H
9] Memory card/register select enable	3A0009H
10] Memory card/set to normal	3A0019H
11] Memory card/bank output (bit 2-0)	380011H
12] Select palette bank 0	3A000FH
13] Select palette bank 1	3A001FH
14] Controller Output (negative logic, open-collector)	380001H
Bit 0 : Bit 1 output from controller 1	
Bit 1 : Bit 2 output from controller 1	
Bit 2 : Bit 3 output from controller 1	
Bit 3 : Bit 1 output from controller 2	
Bit 4 : Bit 2 output from controller 2	
Bit 5 : Bit 3 output from controller 2	
nd code output	320000H (write, byte)
nd code input	320000H (read, byte)

8. Sound code output 320000H (write, byte)

9. Sound code input 320000H (read, byte)

10. Input Port

1] Player 1/top	(0 = ON)	300000H, bit 0
2] Player 1/bottom	(0 = ON)	300000H, bit 1
3] Player 1/left	(0 = ON)	300000H, bit 2
4] Player 1/right	(0 = ON)	300000H, bit 3
5] Player 1/trigger 1	(0 = ON)	300000H, bit 4
6] Player 1/trigger 2	(0 = ON)	300000H, bit 5
7] Player 1/trigger 3	(0 = ON)	300000H, bit 6
8] Player 1/trigger 4	(0 = ON)	300000H, bit 7
10] Player 1/start	(0 = ON)	380000H, bit 0
11] Player 1/select	(0 = ON)	380000H, bit 1
12] Player 2/top	(0 = ON)	340000H, bit 0
13] Player 2/bottom	(0 = ON)	340000H, bit 1
14] Player 2/left	(0 = ON)	340000H, bit 2
15] Player 2/right	(0 = ON)	340000H, bit 3
16] Player 2/trigger 1	(0 = ON)	340000H, bit 4
17] Player 2/trigger 2	(0 = ON)	340000H, bit 5
18] Player 2/trigger 3	(0 = ON)	340000H, bit 6
19] Player 2/trigger 4	(0 = ON)	340000H, bit 7
20] Player 2/start	(0 = ON)	380000H, bit 2
21] Player 2/select	(0 = ON)	380000H, bit 3
22] Memory card 1 insertion status (0 = inserted)		380000H, bit 4

23] Memory card 2 insertion status (0 = inserted)	380000H, bit 5
24] Write protect status detection (0 = write enable)	380000H, bit 6
25] NEO-GEO mode 0 : NEO-GEO 1: MULTI-VIDEO SYSTEM	380000H, bit 7

11. Memory card 800000H-0BFFFFFFH (word, long word)
(8 banks)

When 2K-byte RAM card is used, it is allocated to address
800000H - 800FFFFH, and bits 8 to 15 are ignored.

7) Address map of the Z80

1. Program ROM	0000H-07FFH
2. Work RAM	0F800H-FFFFH

8) I/O map of the Z80

1. Sound code input	0000H	(read)
2. Clear sound code input	0000H	(write)
3. NMI enable	0008H	(write)
4. NMI disable	0018H	(write)
5. Sound code output	000CH	(write)
6. YM2610	0004H-0007H	(read/write)

9) Sound function

1. The NMI can be activated and deactivated by writing sound code from the 68000. The NMI is disabled immediately after the system is reset.
2. The NMI flag is cleared when the sound code is read.
3. The sound command is read from 0000H but a sound acknowledge (or return data) is written to 000CH. On the 68000 side, the single location 320000H is written or read.
4. An interrupt can be triggered by the YM2610 timer.

10) Notes

1. After the CPU is reset and before the interrupt is enabled, the following instruction needs to be executed at least once every 100mS.
MOVE, B D0, 300001H
2. You must access the palette by word or long word. Access should be attempted only during the vertical blanking period. (Noise may appear on the screen if the access is attempted during the visible scanning period.)
3. Set the number of active characters to "0" to erase a 3D-LINE SPRITE.

4. After the CPU is reset, the following instruction needs to be executed immediately before the interrupt is issued.
MOVE, W #7, 3C000CH
This will clear the interrupt register.
5. The following initialization is also necessary after the CPU is reset.
 - [1] Write the transparency character (e.g. 0020H) into the VRAM address 0 to 3FH.
 - [2] Write 0000H into VRAM address 8200H.
6. When designing for NEO-GEO, important characters should not be placed within the left most and right most 16-dot areas, nor in the top and bottom 8-dot areas. These areas may not be visible on some television monitors.
7. For Multi-Video Systems, 8 dots on both the left and right sides should be masked by black characters, using the FIX display mode.
8. When the vertical reduction ratio of a 3D-LINE SPRITE is set to a value other than OFFH, the characters which are addressed by 1EH and 20H of that 3D-LINE SPRITE need to be transparent characters. Or, put a transparent dot in the locations one dot below 1EH and one dot above 20H.

3D-LINE SPRITE

16DOT

224dot
(NTSC)

256dot
(PAL)

16DOT

One character

Display Area

320DOT

512DOT

← 3D-LINE SPRITE

Coordinates

512DOT

Display area

Vertical and Horizontal Positions

0: Character

00
02
04
06
1A
1C
1E
20
3C
3E

→ Horizontal Position- Large

↑ Vertical Position- Large

Display Position When Vertical Position=0 and Horizontal Position=0

PAL

NTSC

NTSC

PAL

.3D-LINE SPRITE

Example of the Number of Active Characters (ACT), Vertical Reduction (BIGV), and Horizontal Reduction (BIGH)

 are display area

ACT=32 BIGV=0FFH BIGH=0FH	ACT=32 BIGV=7FH BIGH=0FH	ACT=33 BIGV=7FH BIGH=0FH	ACT=16 BIGV=0FFH BIGH=0FH	ACT=8 BIGV=7FH BIGH=0FH	ACT=4 BIGV=7FH BIGH=0FH	ACT=2 BIGV=3FH BIGH=0FH	ACT=2 BIGV=3FH BIGH=7
00 02	00 02	00 02	00 02	00 02	00 02	00 04	00 04
	1A 1C	1A 1C 1E 20			1C		OE
				3E 00			
	22 24	1E 20 22 24					
3C		3C					
3E	3C	3E	3E				

Address Mapping of the FIX Area (VRAM) (In the NTSC Mode)

7002	7022	7042		74C2	74E2
7003	7023	7043		74C3	74E3
7004	7024	7044		74C4	74E4
701C	703C	705C		74DC	74FC
701D	703D	705D		74DD	74FD

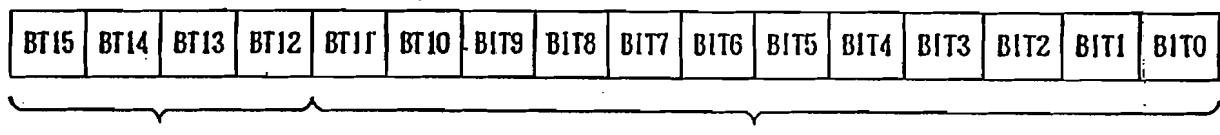


Color Palette Number

Character Number

Address Mapping of the FIX Area (VRAM) (In the PAL Mode)

7000	7020	7040		7400	74E0
7001	7021	7041		74C1	74E1
7002	7022	7042		74C2	74E2
701E	7E3E	705E		74DE	74FE
701F	703F	705F		74DF	74FF



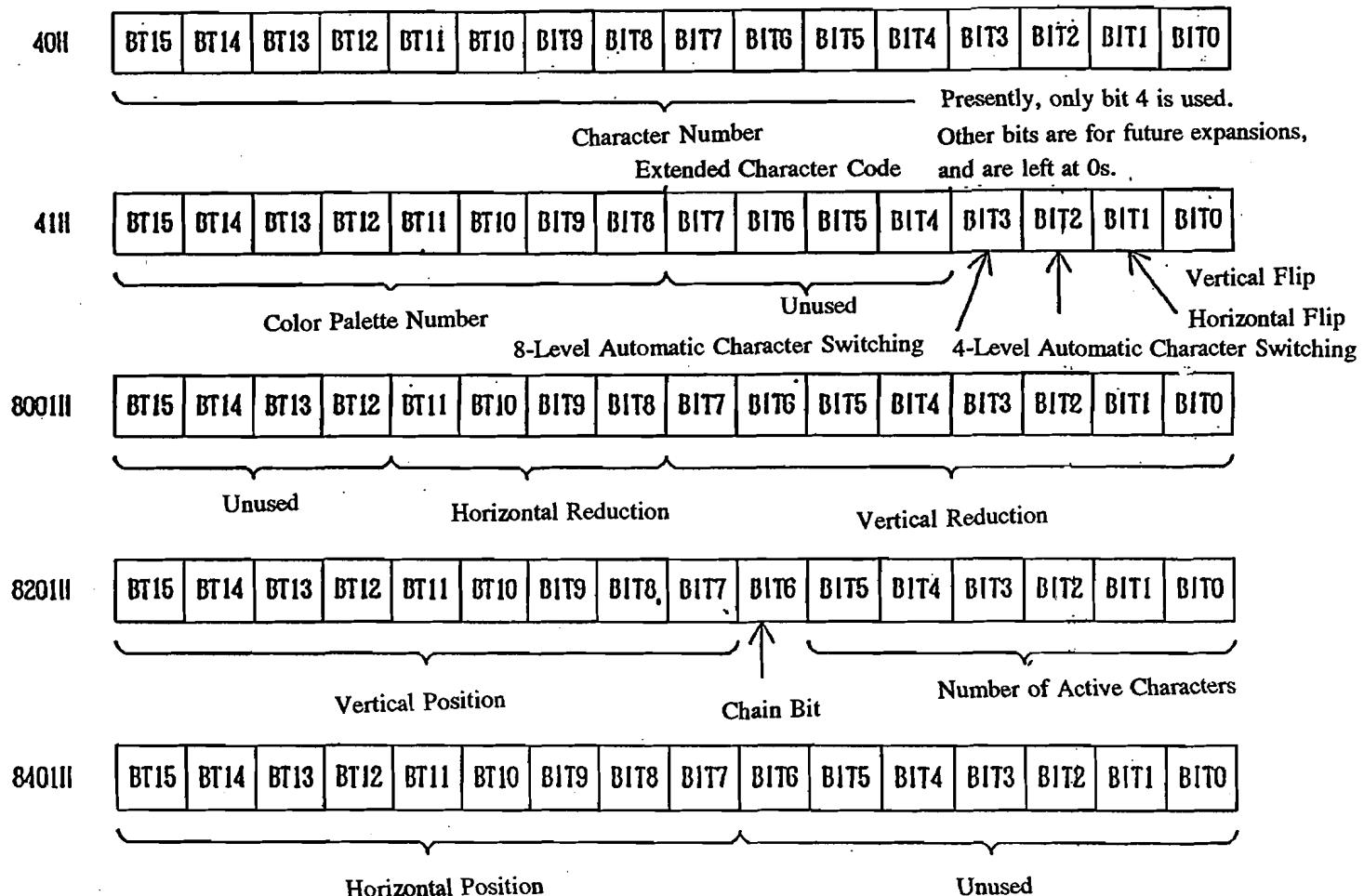
Color Palette Number

Character Number

3D-LINE SPRITE

◎

VRAM Address (Example)



Address Mapping of the 3D-Line Sprite (VRAM) ;by low priority

	Character and Color	Reduction	Vertical Position	Horizontal Position
1	40H-7FH	8001H	8201H	8401H
2	80H-0BFH	8002H	8202H	8402H
3	0C0H-0FFH	8003H	8203H	8403H
.
.
.
379	5ECH0H-5EFHH	817BH	837BH	857BH
380	5F00H-5F3FH	817CH	837CH	857CH

Color Palettes

1. There are 256 color palettes.
2. Each palette can have 15 color codes selected from 65,537 colors.
3. Use color palette number 255 for the background.
4. Use color palette numbers 0-15 for the FIX areas.
5. Use color palette numbers 16-254 for the 3D-line sprites.
bit 4 - bit 7 30 line sprite color
bit 0 - bit 3 fix color

Addresses for the Color Codes

1) Color Palette Number 0	400000H-40001FH
	However, 400000H is 0 (transparent)
2) Color Palette Number 1	400020H-40003FH
3) Color Palette Number 2	400040H-40005FH
.	.
.	.
255) Color Palette Number 254	401FC0H-401FDFH
256) Background	401FFEH-401FFFFH

Color Codes

BIT0	BLUE1
BIT1	BLUE2
BIT2	BLUE3
BIT3	BLUE4
BIT4	GREEN1
BIT5	GREEN2
BIT6	GREEN3
BIT7	GREEN4
BIT8	RED1
BIT9	RED2
BIT10	RED3
BIT11	RED4
BIT12	BLUE0 (Will become a bit darker with 1)
BIT13	GREEN0
BIT14	RED0
BIT15	RGB-T

Data Format of the 3D-Line Sprite Character ROM

ADRS10H	ADRS10H	ADRS10H	ADRS10H	ADRS10H	ADRS10H	ADRS10H	ADRS10H	ADRS 0
BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7	
BIT8	BIT9	BIT10	BIT11	BIT12	BIT13	BIT14	BIT15	
BIT16	BIT17	BIT18	BIT19	BIT20	BIT21	BIT22	BIT23	
BIT24	BIT25	BIT26	BIT27	BIT28	BIT29	BIT30	BIT31	
ADRS 1H								ADRS 1
ADRS 12H								ADRS 2
ADRS 13H								ADRS 3
ADRS 14H								ADRS 4
ADRS 15H								ADRS 5
ADRS 16H								ADRS 6
ADRS 17H								ADRS 7
ADRS 18H								ADRS 8
ADRS 19H								ADRS 9
ADRS 1AH								ADRS 0AH
ADRS 1BH								ADRS 0BH
ADRS 1CH								ADRS 0CH
ADRS 1DH								ADRS 0DH
ADRS 1EH								ADRS 0EH
ADRS 1FH								ADRS 0FH

One character is 16 x 16 dot matrix and is composed of addresses 0-1FH (bits 7-0, bits 15-8, bits 23-16, and bits 31-24) of each 4 ROMs.

However, when using a 16-bit data ROM, it is composed of addresses 0-1FH (bits 15-0 and bits 31-16) for both ROMs.

Data Format of the FIX Area Character ROM

ADRS10H BIT3~0	ADRS10H BIT7~4	ADRS18H BIT3~0	ADRS18H BIT7~4	ADRS 0 BIT3~0	ADRS 0 BIT7~4	ADRS 8 BIT3~0	ADRS 8 BIT7~4
ADRS11H BIT3~0	ADRS11H BIT7~4	ADRS19H BIT3~0	ADRS19H BIT7~4	ADRS 1 BIT3~0	ADRS 1 BIT7~4	ADRS 9 BIT3~0	ADRS 9 BIT7~4
ADRS12H BIT3~0	ADRS12H BIT7~4	ADRS1AH BIT3~0	ADRS1AH BIT7~4	ADRS 2 BIT3~0	ADRS 2 BIT7~4	ADRS0AH BIT3~0	ADRS0AH BIT7~4
ADRS13H BIT3~0	ADRS13H BIT7~4	ADRS1BH BIT3~0	ADRS1BH BIT7~4	ADRS 3 BIT3~0	ADRS 3 BIT7~4	ADRS0BH BIT3~0	ADRS0BH BIT7~4
ADRS14H BIT3~0	ADRS14H BIT7~4	ADRS1CH BIT3~0	ADRS1CH BIT7~4	ADRS 4 BIT3~0	ADRS 4 BIT7~4	ADRS0CH BIT3~0	ADRS0CH BIT7~4
ADRS15H BIT3~0	ADRS15H BIT7~4	ADRS1DH BIT3~0	ADRS1DH BIT7~4	ADRS 5 BIT3~0	ADRS 5 BIT7~4	ADRS0DH BIT3~0	ADRS0DH BIT7~4
ADRS16H BIT3~0	ADRS16H BIT7~4	ADRS1EH BIT3~0	ADRS1EH BIT7~4	ADRS 6 BIT3~0	ADRS 6 BIT7~4	ADRS0EH BIT3~0	ADRS0EH BIT7~4
ADRS17H BIT3~0	ADRS17H BIT7~4	ADRS1FH BIT3~0	ADRS1FH BIT7~4	ADRS 7 BIT3~0	ADRS 7 BIT7~4	ADRS0FH BIT3~0	ADRS0FH BIT7~4

One character is an 8x8 dot matrix and is composed of addresses 0-1FH for one ROM.

Confidential

NEO-GEO SYSTEM PROGRAM

R&D Section, Technical Development Division
SNK Co., Ltd.

— 1 — OUTLINE OF SYSTEM PROGRAM

Two types of program ROMs exist for Neo-Geo. One is provided in the game cartridge (hereafter called the game program, or the game for short).

The other is built into the NEO-GEO unit (hereafter called the system program, or the system for short). The mapping for these is as follows:

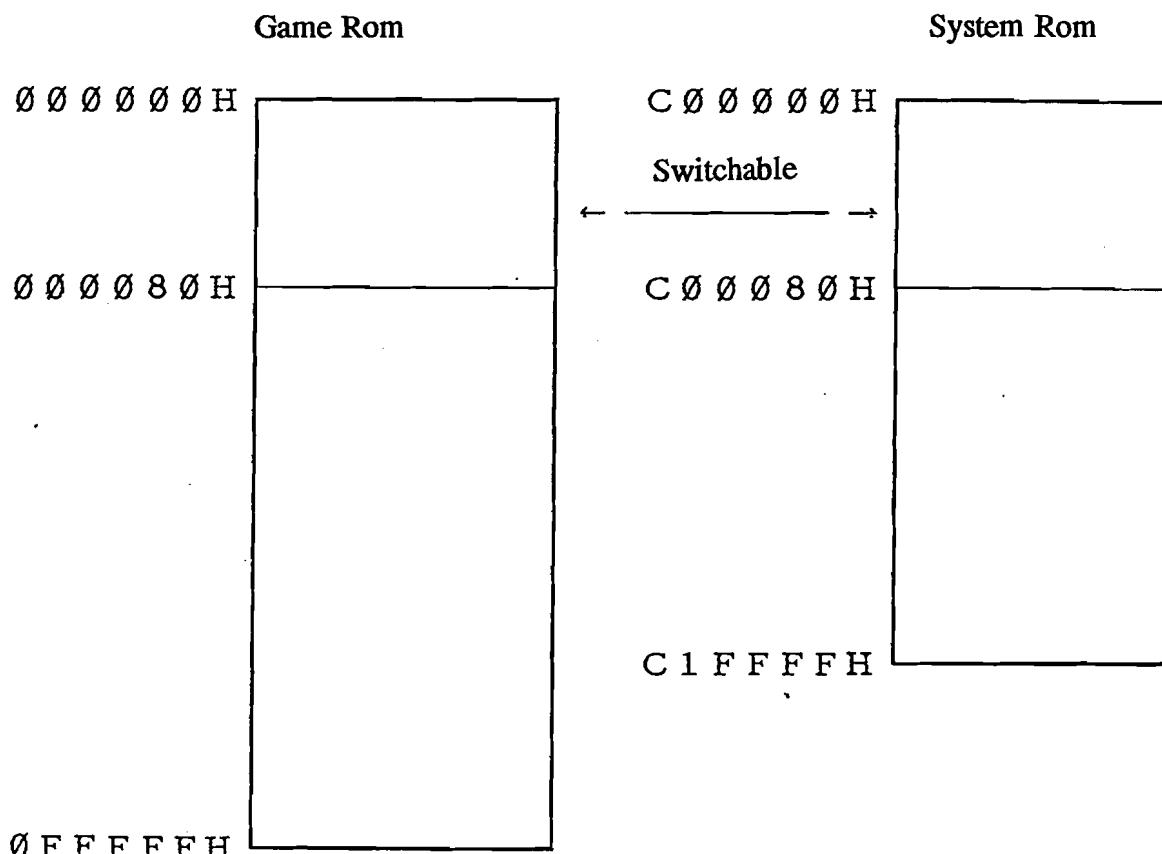


Figure 1-1

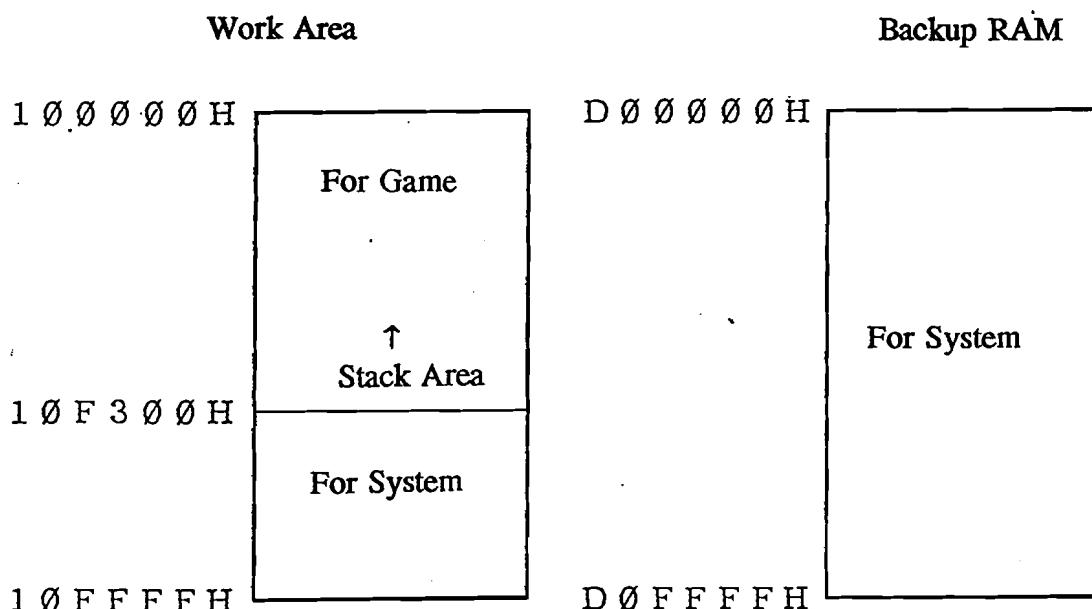
In this mapping, the areas between address 0 to 7FH can be switched using the software. The system side is selected when the system is reset.

The system program is generally composed of the following five parts:

- (1) Control over the entire game
- (2) Standard I/O
- (3) Memory-card interface
- (4) Simple monitoring program (for development)
- (5) Utility program

Furthermore, the NEO-GEO provides addresses 100000H-10FFFFH as a work area, out of which the addresses 10F300H-10FFFFH are reserved exclusively for use by the system program. Therefore, every game is to use addresses 100000H-10F2FFH.

For the multi-video system (hereafter called the MVS), D00000H-D0FFFFH is used as the backup RAM area. This is used exclusively by the system, and cannot be used for the game.



The complete program flow of Neo-Geo is basically as follows:

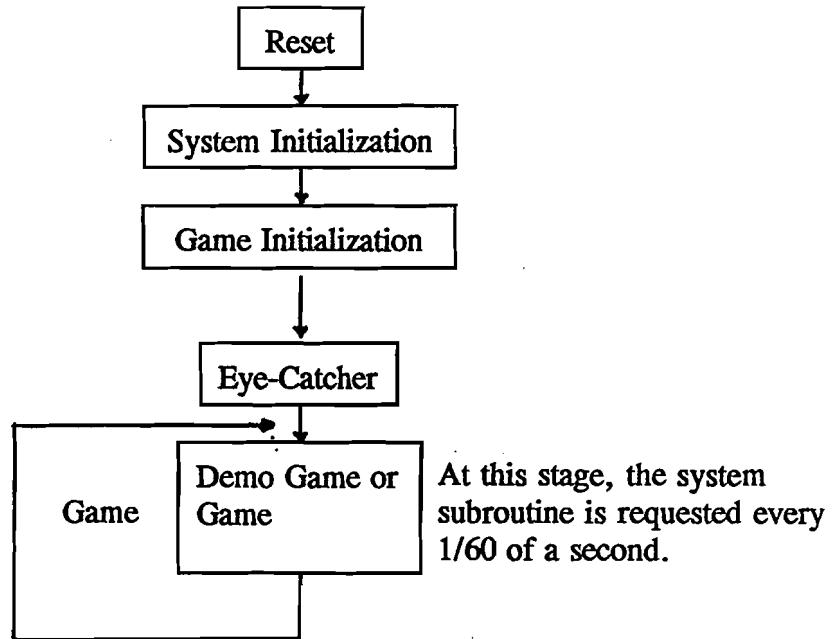


Fig.1-3

[1] System Initialization

As shown in Fig. 1-1, reset processing is done by the system, since the system side is selected for the exception-handling vector (address 0-7FH) when resetting.

At this point, the system work and different types of I/O are initialized, and it is determined whether or not the game cartridge has been loaded.

[2] Game Initialization

Out of the work area, only the backup area (described later) for each game is initialized. This is due to commercial-system (MVS) and home-unit compatibility. For the commercial machine, the hardware is shared among various games, and the status of work and V-RAM vary from game to game. Therefore, initialization is required every time a game is started.

[3] Eye-Catcher

A company logo or the NEO-GEO logo is displayed as an eye-catcher. If sprites for common Fix or the NEO-GEO logo are available, the eye-catcher procedure can be done through the system program.

[4] Game and Demo Game

Normally, a game demo or title display is shown (the system specifies which to show) for approximately 30 seconds. At this stage, if there is a specification or permission from the system to start the game, the game starts. A request is to be made every 1/60 second for the system subroutine [SYSTEM_IO]. It should return to the system program after the demo or end of the game.

--- 2 --- INTERFACE BETWEEN SYSTEM AND GAME

[1] Entries into the System Program from the Game

All entries into the system program from the game are made through the jump table (address C00402H). Because the system uses privileged instructions, be sure to put the system in the supervisor mode when entering the system program.
For the system subroutine, be careful, since the registers are not saved.

[2] Exception-Handling Vector

When the game program is being executed, the game side is selected for the exception-handling vector of address 0-7FH. At this point, handling some exceptions starts the system's simplified monitor. Therefore, define the addresses as follows:

No	Address	Name	Address
00	00	Reset SSP default values	10F300H
01	04	Reset PC default values	C00402H
02	08	Bus error (monitor start-up)	C00408H
03	0C	Address error	C0040EH
04	10	Incorrect command	C00414H
05	14	Division by 0	Defined by game
06	18	CHK command	Defined by game
07	1C	TRAPV command	Defined by game
08	20	Illegal privilege	C0041AH
09	24	Trace exception handling	C00420H
0A	28	No package command (1010)	Defined by game
0B	2C	No package command (1111)	Defined by game
0C-0E	30-38	Unused	C00426H
0F	3C	Uninitialized interrupt	C0042CH
10-17	40-5C	Unused	C00426H
18	60	Virtual interrupt	C00432H
19	64	Interrupt 1	Defined by game
1A	68	Interrupt 2	Defined by game
1B	6C	Interrupt 3 (unused)	Defined by game
1C	70	Interrupt 4 (unused)	Defined by game
1D	74	Interrupt 5 (unused)	Defined by game
1E	78	Interrupt 6 (unused)	Defined by game
1F	7C	Interrupt 7 (unused)	Defined by game

When the foreground monitor ICE (In-Circuit Emulator) is used, define the vectors according to the ICE monitor. (Otherwise, such problems as not being able to execute step by step may occur, causing difficulties in debugging.) But even in this case, define the addresses when placing the program in ROM.

Furthermore, vectors not used in the actual game are to be at No. 6 as unused, even if it is defined in the game.

[3] Game ID

In each game, the address 100H-18H is the ID region that the system refers to. Define the parameters, addresses, etc., according to the following:

ADDRESS	SIZE (byte)	DESCRIPTION									
100H	7		100H	101H	102H	103H	104H	105H	106H		
		ASC	N	E	O	-	G	E	O		
		HEX	4E	45	4F	2D	47	45	4F		
Cartridge Recognition Code 1 The game does not start unless this code has been set correctly.											
107H	1	0	System Version Set at 0 for the present.								
108H	2	Game Code ID Defines the game identification code (provided by SNK) (Note: 0000H is not allowed.)									
10AH	4	Program ROM Size (Example) For 4 M bits, 80000H for 16 M bits, 100000H These are defined in long-words.									
10EH 112H	4 2	Starting address of the work-backup area Size of the work-backup area Defined in long-words and words, the starting address and size of the system-backup area of the work area used for the game. Formula: 100000H <= [START] < [START + SIZE] = 10FC00H 0 < [SIZE] < = 1000H									

ADDRESS	SIZE (byte)	DESCRIPTION
114H	1	<p>0 = the common eye-catcher by the system to be used.</p> <p>This requires the common Fix, exclusive sprites and eye-catcher sounds. Furthermore, define the exclusive sprite bank number (the upper eight bits of the character code) in the following byte:</p> <p>1 = the game's original eye-catcher program</p> <p>2 = no eye-catcher</p>
115H	1	When 114H is 0, this defines the sprite bank (the upper eight bits of the character code) for the eye-catcher sprites.
116H	4	For Japan
11AH	4	For U.S.A.
11EH	4	For Europe (regions other than U.S.A.)
		Defines the starting addresses of data for titles, mode-select menu dips, etc., for each version, in long words. (Contents of data mentioned hereinafter.)
122H	6	USER
128H	6	PLAYER_START
12EH	6	DEMO_END
134H	6	COIN_SOUND
		Define the JMP commands for each program. (Contents of the programs described later)
182H	4	Defines the starting address of the Recognition Code 2 (in long words).

* Cartridge Recognition Code 2

Be sure to start with even addresses.

```
DC.W  Ø76ØØH,Ø4A6DH,ØØA14H,Ø66ØØH  
DC.W  ØØØ3CH,Ø2Ø6DH,ØØAØ4H,Ø3E2DH  
DC.W  ØØAØ8H,Ø13CØH,ØØØ3ØH,ØØØØ1H  
DC.W  Ø321ØH,ØØCØ1H,ØØØFFH,Ø671AH  
DC.W  Ø3Ø28H,ØØØØ2H,ØBØ2DH,ØØACEH  
DC.W  Ø661ØH,Ø3Ø28H,ØØØØ4H,ØBØ2DH  
DC.W  ØØACFH,Ø66Ø6H,ØB22DH,ØØADØH  
DC.W  Ø67Ø8H,Ø5Ø88H,Ø51CFH,ØFFD4H  
DC.W  Ø36Ø7H,Ø4E75H,Ø2Ø6DH,ØØAØ4H  
DC.W  Ø3E2DH,ØØAØ8H,Ø321ØH,ØEØ49H  
DC.W  ØØCØ1H,ØØØFFH,Ø671AH,Ø3Ø1ØH  
DC.W  ØBØ2DH,ØØACEH,Ø6612H,Ø3Ø28H  
DC.W  ØØØØ2H,ØEØ48H,ØBØ2DH,ØØACFH  
DC.W  Ø66Ø6H,ØB22DH,ØØADØH,Ø67Ø8H  
DC.W  Ø5888H,Ø51CFH,ØFFD8H,Ø36Ø7H  
DC.W  Ø4E75H
```

--- 3 --- GAME PROGRAM SPECIFICATION

The following terms will be used in the programming function descriptions that follow.

Title	Subroutine name
Address	Entry address
Function	Summary of subroutine contents.
Conditions	Conditions for entry
Input	Input parameters required at time of entry into the subroutine. B, W or L, following the address, stands for byte, word or long word, respectively.
Output	Output parameters returned at the exit of the subroutine.
Description	Extra notes on the functions performed by the subroutine.

Title	USER
Address	122H
Function	Execution of the game's main program
Input	<p>Command No. (0-3) for the USER_REQUEST (10FDAEH,B)</p> <p>Other settings:</p> <ul style="list-style-type: none"> * 68000 registers SR = 2700H A7 (SSP) = 10F300H Other registers: Undefined <p>* Work Area for the Game (100000-10F2FFH)</p> <p>Only for the backup area of each game. The values at the end of the game, etc., are already undefined. Everything is undefined at the time of command 0 start-up initialization.</p> <p>* I/O</p> <ul style="list-style-type: none"> Timer Interrupt: Not allowed. Automatic Character Switching speed: 64/60 sec. (4000H is set to 3C006H.) Vector: Switch to the game side. Sound Reset Code (03): Already sent. Other I/O: Undefined. <p>* Displays</p> <ul style="list-style-type: none"> Color Palette Banks: All 0s for both 0 and 1 Fix: All at 0020H. Sprite: Vertical Position (Active Characters)... all at 0 Horizontal Position... all out of screen Reduction Parameters... all at 0FFFH Character Codes & Attributes: All undefined.
Description	All game programs start when the system jumps to here (not a subroutine call). Jump to SYSTEM_RETURN (C00444H) when processing is over (when the game is complete, etc.). Also, call the SYSTEM_IO (C0044AH) every 1/60 second.
Description by User-Request Conditions	
No 0	<p>Start-up initialization</p> <p>USER_REQUEST = 0</p> <p>Initialize only the part defined (by the address 10EH-113H) as the backup area of the work area used for the game. Normally, use this area for high scores, rankings, etc. On the MVS, this command is called only once: when the cassette is inserted into the main board for the first time.</p> <p>Initialize the work area, excluding the backup area, screen displays, and I/O, etc. every time USER is entered.</p>

No 1	<p>Eye-catcher</p> <p>USER_REQUEST = 1</p> <p>A request is made only when the address 114H is 1. The call is not made at any other time, nor with the MVS. Only one request is to be made right after the home system is turned on.</p>
No 2	<p>Demo Game/Game</p> <p>USER_REQUEST = 2</p> <p>A demonstration of the title and the game is performed when a request is made. Jump to SYSTEM_RETURN when the demonstration is over and the game has yet not been started. Or, if the game has been started, jump to SYSTEM_RETURN after the game is over.</p>
No 3	<p>Title Display</p> <p>USER REQUEST = 3</p> <p>Only the game title is displayed. This command is requested only in the mode with the MVS-forced start. At this point, the SELECT_TIMER (10FDDAH, B) gives the remaining time in BCD, so please display this on the screen. When the time runs out, instructions come from the system to start the game. Therefore, there is no need for the game side to return to the SYSTEM_RETURN. Everything else is the same as in the Command 2 game demo. (Please note that if the Game Start Compulsion is not set for compulsion time, then the SELECT-TIMER is not used.)</p>

Title	PLAYER_START																								
Address	128H																								
Function	Game-Start Processing																								
Condition	A request is made, if the pressing of the start button is detected, with sufficient credit, in the SYSTEM IO. Or, a call is made when the MVS-forced start is past the time limit.																								
Input	<p>Starting Player in the START_FLAG (10FDD2H, B)</p> <p>START_FLAG</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>d7</td><td>d6</td><td>d5</td><td>d4</td><td>d3</td><td>d2</td><td>d1</td><td>d0</td></tr> <tr> <td>--</td><td>--</td><td>--</td><td>--</td><td>P4</td><td>P3</td><td>P2</td><td>P1</td></tr> <tr> <td colspan="8" style="text-align: right;">Start at "1"</td></tr> </table>	d7	d6	d5	d4	d3	d2	d1	d0	--	--	--	--	P4	P3	P2	P1	Start at "1"							
d7	d6	d5	d4	d3	d2	d1	d0																		
--	--	--	--	P4	P3	P2	P1																		
Start at "1"																									
Output	<p>Sets each bit of the START_FLAG to "1" (as they are) if the game can be started, or to "0" if not. (It is not allowed to change any bits from 0 to 1.) When the game has been started, set the applicable byte to "1" (game status) where the USER_MODE (10FDCAFH, B) is 2 (normal game status, PLAYER_MODE (10FDB6H-, 4 bytes)) during the demo.</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>1 Ø F D B 6 H</td><td>P 1</td></tr> <tr> <td>1 Ø F D B 7 H</td><td>P 2</td></tr> <tr> <td>1 Ø F D B 8 H</td><td>P 3</td></tr> <tr> <td>1 Ø F D B 9 H</td><td>P 4</td></tr> </table>	1 Ø F D B 6 H	P 1	1 Ø F D B 7 H	P 2	1 Ø F D B 8 H	P 3	1 Ø F D B 9 H	P 4																
1 Ø F D B 6 H	P 1																								
1 Ø F D B 7 H	P 2																								
1 Ø F D B 8 H	P 3																								
1 Ø F D B 9 H	P 4																								

Description	<p>A call is made when the system instructs the game to begin. (When starting the game from its own judgment, use the system subroutines CREDIT_CHECK and CREDIT_DOWN, described later.) With the START_FLAG, basically only the bit representing the player whose start button has been pressed becomes 1. For during the MVS demo (the USER_MODE at "1"), the following shows how it works:</p> <table border="1"> <thead> <tr> <th rowspan="2">COUNTRY_CODE (10FD83H,B)</th><th colspan="2">START_FLAG LOWER 4 BITS</th></tr> <tr> <th>P1 START</th><th>P2 START</th></tr> </thead> <tbody> <tr> <td>0=JAPAN</td><td>0001</td><td>0011</td></tr> <tr> <td>1=U.S.A.</td><td>0001</td><td>0010</td></tr> <tr> <td>2=EUROPE</td><td>0001</td><td>0011</td></tr> </tbody> </table> <p>This means: On pressing the P2 START key, Japanese and European specifications allow the two players to start the game simultaneously, while the U.S. specifications allow only player 2 to start the game. Therefore, for such games as sports games, where participation in the middle of a game is not possible, the U.S. specifications require preselection of the two-player game (refer to the sample program for these specifications). When the game is in progress, an extension of play time or the addition of lives is available at the player's option, and participation from the middle of play should be available. If these options are not possible, or if it is a game like Mah-Jong, which allows only one player to play, return after changing the applicable bit to "0."</p> <p>If any of the START FLAG bits is returned with 1, the system deducts credit. Therefore, be sure to perform the start processing, etc. Furthermore, be sure to keep the USER MODE at 2, or the game may be switched to another game.</p>				COUNTRY_CODE (10FD83H,B)	START_FLAG LOWER 4 BITS		P1 START	P2 START	0=JAPAN	0001	0011	1=U.S.A.	0001	0010	2=EUROPE	0001	0011
COUNTRY_CODE (10FD83H,B)	START_FLAG LOWER 4 BITS																	
	P1 START	P2 START																
0=JAPAN	0001	0011																
1=U.S.A.	0001	0010																
2=EUROPE	0001	0011																

Title	DEMO_END
Address	12EH
Function	Procedure to allow the demo to end early.
Condition	When a switch to another game is detected in the SYSTEM_IO during the demonstration (only MVS)
Description	In the MVS, switching to another game can be made by pressing the select button during the demo. This switching interrupts the demo, so if necessary store the items in the work backup area. The system returns itself to SYSTEM_RETURN. Therefore, do not directly jump to SYSTEM RETURN.

Title	COIN_SOUND
Address	134H
Function	Request to the Sound CPU(Z80) for the coin-deposit sound
Parameter	When the deposit of a coin or coins is detected in the SYSTEM_IO (MVS only)

[Explanation] Make a request to the Z80 according to the specifications of the particular game. Be sure to get a coin-deposit sound in any setting or scene.

[Note] **PLAYER_START**, **DEMO_END** and **COIN_SOUND** are all called from SYSTEM_IO.

[2] Data Referred to by the System

In conventional commercial games, the game's difficulty level, etc., was set using the DIP switch on the board. With the MVS, the setting is performed by the memory switch using backup memory --, which is called "mode-select menu." Every game needs to have data on default values, descriptions of items, and the contents of the modes. The starting addresses are 116H, 11AH and 11EH, which are defined in long words. (Refer to page 9.)

* Data Format

ADDRESS (OFFSET)	SIZE (BYTE)	CONTENT
+0	16	Game title
+16	16	Mode-select default
+32	12	Mode-select item/description of contents
+44	12	Mode-select item/description of contents
+56	12	Mode-select item/description of contents

(1) Game Title

Define the game title (16 bytes) in the common Fix code.

(2) Mode-Select Default

The mode-select menu can have up to 14 items. The beginning four items are reserved for specific purposes, but the remaining 10 items can be used freely for each game.

Item No.	Offset	Size	Content
1,2	+0,+2	WORD	Setting of (extended) play time, etc. The default values are set using BCD, with the minutes in the upper byte and seconds in the lower byte. Setting in seconds is possible in the lower byte up to 29 minutes and 59 seconds. If this is not used, specify 0FFFFH.
3	+4	BYTE	Set the remaining lives, etc., from 1 ~ 99 using binary code. Specify 0FFH if unused.
4	+5	BYTE	The default value of the extended play is set between 0 and 100 in binary, with 0 for no extended play, 1 to 99 for the extended-play limit, and 100 for unlimited extended plays. Specify 0FFH if unused.
5-14	+5+15	BYTE	This can be used freely for each game. The default value (0-0FH) is set in the upper four bits, and the content number (1-0FH) is set in the lower four bits. However, if the lower four bits are 0s, items beyond this point are not used.

(3) Description on Items and Contents of the Mode-Select Menu

This is the description on the items and contents for display when a change(s) in setting the modes is/are made on the screen. Define each in the 12-byte common Fix code. Define only the names for items 1 through 4, but do not define an item(s) that are not used. For the item 5 and above, define the item name, and the contents when set at "0," set at "1" and so on... up to 15.

[Setting Example 1]

For a game based on remaining lives

1	DC. B	' SAMPLE	GAME1	
2				
3	DC. W	0FFFFH	;ITEM1	(TIME)
4	DC. W	0FFFFH	;ITEM2	(TIME)
5	DC. B	3	;ITEM3	(REMAINING LIVES)
6	DC. B	100	;ITEM4	(EXTENDED PLAY)
7	DC. B	14H	;ITEM5	
8	DC. B	13H	;ITEM6	
9	DC. B	24H	;ITEM7	
10	DC. B	01H	;ITEM8	
11	DC. B	00H	;ITEM9	
12	DC. B	00H	;ITEM10	
13	DC. B	00H	;ITEM11	
14	DC. B	00H	;ITEM12	
15	DC. B	00H	;ITEM13	
16	DC. B	00H	;ITEM14	
17	DC. B	'HERO	';ITEM3	ITEM NAME
18	DC. B	'CONTINUE	';ITEM4	ITEM NAME
19				
20	DC. B	'DIFFICULTY	';ITEM5	ITEM NAME
21	DC. B	'EASY	';	0
22	DC. B	'NORMAL	';	1
23	DC. B	'HARD	';	2
24	DC. B	'VERY HARD	';	3
25				
26	DC. B	'BONUS	';ITEM6	ITEM NAME
27	DC. B	'NO BONUS	';	0
28	DC. B	'EVERY BONUS	';	1
29	DC. B	'SECOND BONUS	';	2
30				
31	DC. B	'BONUS RATE	';ITEM7	ITEM NAME
32	DC. B	'20000/10000	';	0
33	DC. B	'30000/10000	';	1
34	DC. B	'50000/30000	';	2
35	DC. B	'100000/500000	';	3
36				
37	DC. B	'DEMO SOUND	';ITEM8	ITEM NAME
38	DC. B	'WITH	';	0
39	DC. B	'WITH OUT	';	1

On-screen display of initialization status and set values

Sample Game 1
(Set value)

HERO	3	3
CONTINUE	NO LIMIT	100
DIFFICULTY	NORMAL	1
BONUS	EVERY BONUS	1
BONUS RATE	50000/30000	2
DEMO SOUND	WITH	0

Comments

- LINE 1 Specify the game title in 16 bytes.
LINES 3 & 4 Define OFFFFFH, since the game is based on remaining lives and time is not used.
- LINE 5 Set three lives
LINE 6 Unlimited number of extended plays
LINES 7-10 Default values and numbers of contents that can be set on items 5 to 8. For example, for item 5 (DIFFICULTY LEVEL), four types can be set from 0 to 3 (EASY, NORMAL, HARD, VERY HARD) with the default value as 1 (NORMAL).
- LINE 11 Since the lower four bits are at "0," items 9 through 14 are not used. Although these items are not used, space for data needs to be reserved.
- LINES 17-18 Since items 1 and 2 are not used, definition begins from item 3. But on items 1 to 4, only item names are to be defined.
- LINE 19 The name of item 5
LINES 20-24 Use words to express the contents of the set values from 0 to 3 respectively, for item 5. Four types of settings are possible, so four descriptions are necessary.

[Setting Example 2]

Game on a time-unit basis

1	DC. B	' SAMPLE	GAME2	
2				
3	DC. W	00330H	;ITEM1	(TIME)
4	DC. W	00300H	;ITEM2	(TIME)
5	DC. B	0FFH	;ITEM3	(REMAINING LIVES)
6	DC. B	14H	;ITEM4	(EXTENDED PLAY)
7	DC. B	03H	;ITEM5	
8	DC. B	04H	;ITEM6	
9	DC. B	01H	;ITEM7	
10	DC. B	00H	;ITEM8	
11	DC. B	00H	;ITEM9	
12	DC. B	00H	;ITEM10	
13	DC. B	00H	;ITEM11	
14	DC. B	00H	;ITEM12	
15	DC. B	00H	;ITEM13	
16	DC. B	00H	;ITEM14	
17				
18	DC. B	'PLAY TIME	;ITEM1	ITEM NAME
19	DC. B	'CONT. TIME	;ITEM2	ITEM NAME
20				
21	DC. B	'DIFFICULTY	;ITEM5	ITEM NAME
22	DC. B	'EASY	;	0
23	DC. B	'NORMAL	;	1
24	DC. B	'HARD	;	2
25	DC. B	'VERY HARD	;	3
26				
27	DC. B	'BONUS	;ITEM6	ITEM NAME
28	DC. B	'NO BONUS	;	0
29	DC. B	'EVERY BONUS	;	1
30	DC. B	'SECOND BONUS	;	2
31				
32	DC. B	'BONUS RATE	;ITEM7	ITEM NAME
33	DC. B	'20000/10000	;	0
34	DC. B	'30000/10000	;	1
35	DC. B	'50000/30000	;	2
36	DC. B	'100000/50000	;	3
37				
38	DC. B	'DEMO SOUND	;ITEM8	ITEM NAME
39	DC. B	'WITH	;	0
40	DC. B	'WITH OUT	;	1

On-screen display of initialization and set values

SAMPLE GAME 2

		(Set Value)
PLAY TIME	03 MINUTES 30 SECONDS	0330H
CONT. TIME	03 MINUTES 00 SECONDS	0300H
CONTINUE	NONE	0
DIFFICULTY	NORMAL	1
BONUS	NO BONUS	0
BONUS RATE	20000/10000	0
DEMO SOUND	WITH	0

The set values are transferred to the work GAME_DIP (10FD84H-, 16 bytes) when entering USER (122H). Therefore, refer to the values and carry out the applicable process for each game. The system makes a change(s) in the modes or transfers the set values to the work area, but does not process any of the contents whatsoever. Care must be taken in this respect. When it comes to the home system, there is no display of the mode-select menu, etc., so please set these modes freely for each game.

--- 4 --- SYSTEM PROGRAMS SPECIFICATION

[1] System Programs

Enter the following by the JMP instruction instead of subroutines. Be sure to always make the entry in the supervisor mode.

Title	SYSTEM_INT1
Address	C00438H
Function	System Level 1 Interrupt Program
Condition	When it is determined that it is in the system mode at the beginning of the level 1 interrupt. If bit 7 in the SYSTEM_MODE (10FD80H, B) is at "0," it is determined to be in the system mode, and if at "1" it is in the regular game mode.
Description	At the time of entering USER (122H), the 0-7FH exception handling vector from the game is used, and bit 7 in the SYSTEM_MODE is at 1—the game mode. When the system is in full operation, the vector from the system is used, and the bit 7 in the SYSTEM_MODE is at 0—the system mode. If the vector is mapped onto the emulation memory when switching vectors or debugging and vector switching is not possible, the vector from the game may be selected even in the system mode. In this case, jump to the SYSTEM_INT1 at the beginning of interrupt "1" for the game. Furthermore, the SYSTEM_INT1 is involved only in accessing the SYSTEM_IO, MESS_OUT (described later) and watchdog. Therefore, for games with long "no interrupt" times directly after entering USER, the game itself can allow interrupts by placing a "0" for bit 7 of SYSTEM_MODE at that time.

Title	SYSTEM_RETURN
Address	C00444H
Function	Returns from the game program to the system program.
Parameter	This subroutine should be called when all jobs are finished after USER entry.

[2] System Subroutines

There is no guarantee that the contents of the registers are preserved after the program exits a subroutine.

Title	SYSTEM_IO
Address	C0044AH
Function	Input and output operations from the system program.
Condition	This subroutine should be called every 1/60 of a second during the game program. When it is called outside of the interrupt-handling routine, and if the program needs to access the I/O besides the screen using the interrupt-handling routine, prohibit the interrupt request while SYSTEM_IO is executed, which takes about 300 micro seconds on average.
Description	<p>This subroutine executes the following:</p> <ol style="list-style-type: none">1 Senses joystick-input status.2 Detects coins inserted in the slot (MVS only). If required, COIN_SOUND(134H) is called.3 Checks the start button. If required, PLAYER_START(128H) is called.4 Checks the game selection (MVS only). If necessary, DEMO_END(12EH) is called and the game is selected.5 Timers used by the system are counted up or down. <p>To detect the coin input precisely, this routine needs to be called every 1/60 of a second. In general, it is recommended that this routine be called at the end of the interrupt-1 handler routine. However, if this is called from the main routine and not from the interrupt-1 handler routine due to the V-RAM output and such, be sure that the processing does not overflow. Since the coin-input detection routine also controls the credit status, that the game program does not have to execute any coin-related tasks.</p> <p>Status of the joystick is stored in the work area, INPUT_1(10FD94H), and the game routine can read the status from this work area rather than reading it directly from I/O.</p>

Title	CREDIT_CHECK
Address	C00450H
Function	The system verifies the credit status.
Input	In addresses CREDIT_DEC (10FDB0H, B) and CREDIT_DEC +1, put the number of credits (BCD, 0-99H) needed for player 1 and player 2 respectively.
Output	Unchanged if the content of CREDIT_DEC (10FDB0H, B) and CREDIT_DEC +1 is sufficient to start the game. Otherwise it's set to "0."
Description	Before the game program can start the game or add remaining lives without using PLAYER_START, the credit should be confirmed first using this subroutine. The game program can use this routine to simply check if the player can start the game, because it only checks the credits left.

Title	CREDIT_DOWN
Address	C00456H
Function	Credit deduction executed by the system.
Condition	After the system checks the credits by calling the CREDIT_CHECK routine.
Input	Place the exact returned values from CREDIT_CHECK for CREDIT_DEC and CREDIT_DEC +1.
Description	Credits are deducted by the value of CREDIT_DEC, and the system executes the operation in preparation for starting a game. Be sure to execute the processing to start the games in the game program.

Title	READ_CALENDAR																					
Address	C0045CH																					
Function	Reading from the calendar																					
Condition	MVS only																					
Output	<p>Current date and time data are stored in seven bytes, starting from the address label DATE_TIME (10FDD2H, seven bytes).</p> <table> <tr> <td>+0</td> <td>Year</td> <td>Last two digits of the year in BCD</td> </tr> <tr> <td>+1</td> <td>Month</td> <td>Two digits in BCD (1-12)</td> </tr> <tr> <td>+2</td> <td>Day</td> <td>Two digits in BCD (1-31)</td> </tr> <tr> <td>+3</td> <td>Week</td> <td>0 to 6, starting from Sunday, Monday,... Saturday.</td> </tr> <tr> <td>+4</td> <td>Hours</td> <td>Two digits BCD (0-23)</td> </tr> <tr> <td>+5</td> <td>Minutes</td> <td>Two digits BCD (0-59)</td> </tr> <tr> <td>+6</td> <td>Seconds</td> <td>Two digits BCD (0-59)</td> </tr> </table>	+0	Year	Last two digits of the year in BCD	+1	Month	Two digits in BCD (1-12)	+2	Day	Two digits in BCD (1-31)	+3	Week	0 to 6, starting from Sunday, Monday,... Saturday.	+4	Hours	Two digits BCD (0-23)	+5	Minutes	Two digits BCD (0-59)	+6	Seconds	Two digits BCD (0-59)
+0	Year	Last two digits of the year in BCD																				
+1	Month	Two digits in BCD (1-12)																				
+2	Day	Two digits in BCD (1-31)																				
+3	Week	0 to 6, starting from Sunday, Monday,... Saturday.																				
+4	Hours	Two digits BCD (0-23)																				
+5	Minutes	Two digits BCD (0-59)																				
+6	Seconds	Two digits BCD (0-59)																				
Explanation	This subroutine is only for the MVS. Use this for updating the ranking and beekeeping at certain intervals.																					

Title	FIX_CLEAR
Address	C004C2H
Function	Clears the fix display.
Output	Put 020H (opaque character) on the entire line at both sides, and puts 0FFH (transparent character) in the rest of the Fix display area. These two characters are defined as common Fix characters.

Title	LSP_1st
Address	C004C8H
Function	Initializes the line sprite.
Output	<p>Set the following values to line sprites 1 to 380:</p> <p>V-POSITION : 0 H-POSITION : Off the screen Reduction Ratio: 0FFFH</p> <p>However, the character codes and attributes are not changed.</p>

Title	MESS_OUT
Address	C004CEH
Function	Generic V-RAM output (explained later)

Title	CARD
Address	C00468H
Function	Access to the memory card (explained later in more detail)

Title	CARD_ERROR
Address	C0046EH
Function	Handles memory card errors. (explained later in more detail)

[3] System Work Area

An asterisk following the name indicates that the value is written by the game. Do not write into the others or undefined memory addresses.

Title	SYSTEM_MODE						
Address	10FD80H 1 byte						
Description	<p>Current software mode status:</p> <table style="margin-left: 40px;"> <tr> <td>bit 7 =</td> <td>0</td> <td>system mode</td> </tr> <tr> <td></td> <td>1</td> <td>game mode</td> </tr> </table> <p>The system is not triggered by the game-start button and game-selection button while it is in the system mode. (Therefore, <code>PLAYER_START</code> would not be called.) During the interrupt-handler routine of the game program (such as during game initialization), and when <code>PLAYER_START</code> cannot be called, the system can be in the system mode temporarily, but will return to the game mode as soon as possible.</p>	bit 7 =	0	system mode		1	game mode
bit 7 =	0	system mode					
	1	game mode					

Title	MVS_FLAG				
Address	10FD82H 1 byte				
Description	<p>Indicates if this system is for commercial or home use.</p> <table style="margin-left: 40px;"> <tr> <td>0</td> <td>Home use</td> </tr> <tr> <td>other values</td> <td>Commercial use</td> </tr> </table>	0	Home use	other values	Commercial use
0	Home use				
other values	Commercial use				

Title	COUNTRY_CODE						
Address	10FD83H 1 byte						
Description	<p>Country specification:</p> <table style="margin-left: 40px;"> <tr> <td>0</td> <td>Japan</td> </tr> <tr> <td>1</td> <td>USA</td> </tr> <tr> <td>2</td> <td>Europe (or countries other than Japan and USA)</td> </tr> </table> <p>This information is used to select the language to be used in the game and the type of coins to accept. (Refer to the <code>PLAYER_START</code> section about the coin system.)</p>	0	Japan	1	USA	2	Europe (or countries other than Japan and USA)
0	Japan						
1	USA						
2	Europe (or countries other than Japan and USA)						

Title	GAME_DIP
Address	10FD84H - 10FD93H 16 byte
Description	Game parameters of each game.

Title	INPUT_1																																																																								
Address	10FD94H - 10FD99H 6 bytes																																																																								
Description	<p>Current status of Controller 1.</p> <p>+0 Controller 1 status 0 = No connection 1 = Normal controller 2 = Expanded controller 3 = Mah-jong controller 4 = Keyboard</p> <p>This value is set when the program jumps to SYSTEM_RETURN after the reset.</p> <p>+1 Controller 1 : Status from 1/60 second before +2 Controller 1 : Current status +3 Controller 1 : Active-edge flag +4 Controller 1 : Auto-repeat flag +5 Controller 1 : Repeat timer</p> <p>Values in addresses [INPUT_1 + 1] to [INPUT_1 + 4] are expressed in positive logic, and bit assignments are:</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>d7</td><td>d6</td><td>d5</td><td>d4</td><td>d3</td><td>d2</td><td>d1</td><td>d0</td></tr> <tr> <td>D</td><td>C</td><td>B</td><td>A</td><td>RIGHT</td><td>LEFT</td><td>BOTTOM</td><td>TOP</td></tr> </table>	d7	d6	d5	d4	d3	d2	d1	d0	D	C	B	A	RIGHT	LEFT	BOTTOM	TOP																																																								
d7	d6	d5	d4	d3	d2	d1	d0																																																																		
D	C	B	A	RIGHT	LEFT	BOTTOM	TOP																																																																		
Example:	<p>ON OFF </p> <p style="text-align: center;"><-16/60sec-> <-8/60sec-> <-8/60sec-></p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>+1</td><td>?</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>.</td><td>.</td><td>.</td><td>.</td><td>1</td><td>.</td><td>.</td><td>.</td><td>1</td></tr> <tr> <td>+2</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>.</td><td>.</td><td>.</td><td>.</td><td>1</td><td>.</td><td>.</td><td>.</td><td>1</td></tr> <tr> <td>+3</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>.</td><td>.</td><td>.</td><td>.</td><td>0</td><td>.</td><td>.</td><td>.</td><td>0</td></tr> <tr> <td>+4</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>.</td><td>.</td><td>.</td><td>.</td><td>0</td><td>1</td><td>0</td><td>.</td><td>0</td></tr> </table>	+1	?	0	1	0	1	1	0	1	1	.	.	.	1	+2	0	1	0	1	1	0	1	1	1	.	.	.	1	+3	0	1	0	1	0	0	1	0	0	.	.	.	0	+4	0	1	0	1	0	0	1	0	0	1	0	.	0
+1	?	0	1	0	1	1	0	1	1	.	.	.	1																																																								
+2	0	1	0	1	1	0	1	1	1	.	.	.	1																																																								
+3	0	1	0	1	0	0	1	0	0	.	.	.	0																																																								
+4	0	1	0	1	0	0	1	0	0	1	0	.	0																																																								

The value in address [INPUT_1 + 3] is set to "1" only when the status changes from OFF to ON. The value in address [INPUT_1 + 4] is set to "1" when the status changes from OFF to ON and at a fixed interval if the switch is kept ON. (After 16/60 second from the first active edge, and every 8/60 second thereafter.)

Because the auto-repeat flag monitors all the bits of the byte, if there is a change on another bit, it will immediately be set to "1," even before the time interval mentioned above elapses.

	<p>Even if the value of $[INPUT_1 + 0]$ (status) is "0" (no connection), all the values are set, assuming that a normal controller is connected, but there is no input if the controller status is 4 (keyboard).</p> <p>When the controller status is 3 (mah-jong), the switches are set as the follows:</p> <table border="1"> <thead> <tr> <th></th><th>d7</th><th>d6</th><th>d5</th><th>d4</th><th>d3</th><th>d2</th><th>d1</th><th>d0</th></tr> </thead> <tbody> <tr> <td>+1</td><td>-</td><td>G</td><td>F</td><td>E</td><td>D</td><td>C</td><td>B</td><td>A</td></tr> <tr> <td>+2</td><td>-</td><td>N</td><td>M</td><td>L</td><td>M</td><td>J</td><td>I</td><td>H</td></tr> <tr> <td>+3</td><td>-</td><td>-</td><td>-</td><td>Reach</td><td>Ron</td><td>Kan</td><td>Chi</td><td>Pong</td></tr> </tbody> </table> <p>These values are the raw data; the active-edge flags are set in the addresses starting from $INPUT_3$ with the same address offset (positive logic).</p> <p>These controller input values (addresses $[INPUT_1 + 1]$ to $[INPUT_1 + 5]$) are set by the SYSTEM_IO subroutine.</p>											d7	d6	d5	d4	d3	d2	d1	d0	+1	-	G	F	E	D	C	B	A	+2	-	N	M	L	M	J	I	H	+3	-	-	-	Reach	Ron	Kan	Chi	Pong
	d7	d6	d5	d4	d3	d2	d1	d0																																						
+1	-	G	F	E	D	C	B	A																																						
+2	-	N	M	L	M	J	I	H																																						
+3	-	-	-	Reach	Ron	Kan	Chi	Pong																																						

Title	INPUT_2, 3, 4
Address	10FD9AH, 10FDA0H, 10FDA6H
Description	<p>Status of controllers 2, 3, and 4 are set.</p> <p>The contents are the same as for INPUT_1.</p> <p>When the controllers are expanded, they are assigned as follows:</p> <p>INPUT_1 = controller 1A INPUT_2 = controller 1B INPUT_3 = controller 2A INPUT_4 = controller 2B</p> <p>When a controller for mah-jong is used, the active-edge flags of controllers 1 and 2 are set in the addresses INPUT_3 and INPUT_4 respectively.</p>

Title	INPUT_5, 6																			
Address	10FEE8H, 10FEEEH																			
Description	<p>When controllers 1 and 2 are normal controllers, INPUT_5 and INPUT_6 will have the same values as those in INPUT_1 and INPUT_2. When the controller is for mah-jong, the values are converted to those of the normal controller. The following is the table of conversion:</p> <table border="1"> <tr> <td>Mah-jong</td><td>Normal</td></tr> <tr> <td>A button</td><td>A button (bit 4)</td></tr> <tr> <td>B button</td><td>B button (bit 5)</td></tr> <tr> <td>C button</td><td>C button (bit 6)</td></tr> <tr> <td>D button</td><td>D button (bit 7)</td></tr> <tr> <td>E button</td><td>Joystick up (bit 0)</td></tr> <tr> <td>F button</td><td>Joystick down (bit 1)</td></tr> <tr> <td>G button</td><td>Joystick left (bit 2)</td></tr> <tr> <td>H button</td><td>Joystick right (bit 3)</td></tr> </table>		Mah-jong	Normal	A button	A button (bit 4)	B button	B button (bit 5)	C button	C button (bit 6)	D button	D button (bit 7)	E button	Joystick up (bit 0)	F button	Joystick down (bit 1)	G button	Joystick left (bit 2)	H button	Joystick right (bit 3)
Mah-jong	Normal																			
A button	A button (bit 4)																			
B button	B button (bit 5)																			
C button	C button (bit 6)																			
D button	D button (bit 7)																			
E button	Joystick up (bit 0)																			
F button	Joystick down (bit 1)																			
G button	Joystick left (bit 2)																			
H button	Joystick right (bit 3)																			

Title	INPUT_S																																			
Address	10FDACH 2 bytes																																			
Function	<p>Status of the start button and select button</p> <p>+0 Raw data +1 Active-edge flag Both are in positive logic.</p> <p>Bit format is as follows:</p> <table> <tr> <td>d7</td><td>d6</td><td>d5</td><td>d4</td><td>d3</td><td>d2</td><td>d1</td><td>d0</td><td></td></tr> <tr> <td>P4</td><td>P4</td><td>P3</td><td>P3</td><td>P2</td><td>P2</td><td>P1</td><td>P1</td><td></td></tr> <tr> <td>Select</td><td>Start</td><td>Select</td><td>Start</td><td>Select</td><td>Start</td><td>Select</td><td>Start</td><td></td></tr> </table>									d7	d6	d5	d4	d3	d2	d1	d0		P4	P4	P3	P3	P2	P2	P1	P1		Select	Start	Select	Start	Select	Start	Select	Start	
d7	d6	d5	d4	d3	d2	d1	d0																													
P4	P4	P3	P3	P2	P2	P1	P1																													
Select	Start	Select	Start	Select	Start	Select	Start																													
The select button is not sensed in MVS mode. (It is used to select games.)																																				

Title	USER_REQUEST
Address	10FDAEH 1 BYTE
Description	A command number is set here when USER (122H) is executed. 0 = Start-up initialization 1 = Eye-catcher 2 = Demonstration game 3 = Title only

Title	USER_MODE [*]
Address	10FDAFH 1 byte
Description	Set the current status of the game program with the game software. 0 = Start-up initialization, eye-catcher 1 = Title, game demo 2 = Game in progress Game selection is enabled only when the mode is "1" for the MVS. Make sure to change the mode to "2" when the game starts after the demo.

Title	CREDIT_DEC [*]
Address	10FDB0H 4 bytes
Description	Before calling CREDIT_CHECK and CREDIT_DOWN, set the necessary credits for each player here in one byte of BCD for each player. (Set 0 for the player who is not playing.) Use the exact returned value from CREDIT_CHECK when calling CREDIT_DOWN.

Title	START_FLAG
Address	10FDB4H 1 byte
Function	Player(s) who is starting the game is indicated when PLAYER_START (128H) is called. (Refer to PLAYER_START for details.) START_FLAG is valid only when PLAYER_START is called; the value is undefined otherwise. Do not use this value directly in the game program.

Title	PLAYER_MODE ["]
Address	10FDB6H 4 byte
Function	Set the current status of the players in the order of player 1, 2, 3, and 4. 0 = Never played a game 1 = Currently playing 2 = Continue option being displayed 3 = Game over

Title	DATE_TIME
Address	10FDD2H 7 byte
Description	The current date and time are set when READ_CALENDAR is called. (Refer to READ_CALENDAR for the data format.)

Name	CARD_COMMAND ["]
Address	10FDCC4H 1 byte
Name	CARD_ANSWER ["]
Address	10FDCC6H 1 byte
Name	CARD_START ["]
Address	10FDCC8H 1 long word
Name	CARD_SIZE ["]
Address	10FDCCCH 1 word
Name	CARD_FCB ["]
Address	10FDCEH 1 word
Name	CARD_SUB ["]
Address	10FDD0H 1 byte or 1 word
Function	Addresses listed above are used when CARD(C00462H) and CARD_ERROR(C00468H) are called.

The memory card functions are explained in detail in Section 6 "Memory Card," p.47.

Title	MESS_BUSY ["]
Address	10FDCC2H 1 byte
Function	If the value is not "0," MESS_OUT (generic V-RAM output routine) is skipped. Always put "0" before calling CARD_ERROR.

Title	MESS_POINT [*]
Address	10FDBEH 1 long word
Function	Set pointer for MESS_OUT. Refer to "Using MESS_OUT," p.37 for more details.

Title	SOUND_STOP
Address	D00046H 1 byte
Function	If this is set to "0" in the MVS, do not play any demo sound regardless of the settings in the mode-select menu (except for sound effects for coin insertion). This work area is located in the backup memory, and cannot be accessed by the system for home use.

--- 5 --- COMMON FIX CHARACTER CODE AND GENERIC V-RAM OUTPUT

[1] Common Fix character code

The data displayed by the system for mode-select menu and such, should use the following common Fix character code:

TYPE 1

Used for the game title and data title in the memory card

TYPE 2

Used exclusively for explaining the mode select menu

TYPE 3

Used for displaying all information on overseas versions

In the table, [SP] (20H) denotes the "SPACE" character. All three types have 0FFH assigned for the "end" code, so 0FFH cannot be used in the data.

One Kanji character of TYPE 2 occupies two bytes. (e.g. Kanji character [NAN] can be defined as 00H, 01H.)

Type 1

Used Exclusively for Japanese
MESS_OUT Output Command 9

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	が	ぎ	ぐ	げ	こ	ざ	じ	す	ぜ	ぞ	だ	ち	づ	で	ど	ば
10	び	ぶ	べ	ぼ	ば	び	ぶ	ペ	ぼ					.		
20	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	
30	Ø	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[]			
60	ガ	ギ	グ	ゲ	ゴ	ザ	ジ	ズ	ゼ	ゾ	ダ	チ	ヅ	デ	ド	バ
70	ビ	ブ	ベ	ボ	バ	ピ	ブ	ペ	ボ							
80	あ	い	う	え	お	か	き	く	け	こ	さ	し	す	せ	そ	た
90	ち	つ	て	と	な	に	ぬ	ね	の	は	ひ	ふ	へ	ほ	ま	み
A0	む	め	も	や	ゆ	よ	ら	り	る	れ	ろ	わ	を	ん	あ	い
B0	う	え	お	や	ゆ	よ	づ	。	。	一	「	」				
C0	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ	タ
D0	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ	ミ
E0	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ヲ	ン	ア	イ
F0	ウ	エ	オ	ヤ	ユ	ヨ	ツ	。	。	一	、	。				

Type 2 Used Exclusively for the Mode Select Menu
 MESS_OUT Output Command 8

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	難	易	度	時	間						分	秒	有			
10	◀	▶	▲	▼	無	設	定				持	機	數			
20	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	
30	Ø	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
50	P	Q	R	S	T	U	V	W	X	Y	Z	[]	█	█	█	█
60	ø	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z		遊	び		
80	あ	い	う	え	お	か	き	く	け	こ	さ	し	す	せ	そ	た
90	ち	つ	て	と	な	に	ぬ	ね	の	は	ひ	ふ	へ	ほ	ま	み
A0	む	め	も	や	ゆ	よ	ら	り	る	れ	ろ	わ	を	ん	あ	い
B0	う	え	お	や	ゆ	よ	つ	、	。	一	「	」	方	説		
C0	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ	タ
D0	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ	ミ
E0	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ヲ	ン	ア	イ
F0	ウ	エ	オ	ヤ	ユ	ヨ	ツ	、	。	一	、	。	明			

Type 3

Used Exclusively for English
 MESS_OUT Output Command 8

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00
10	◀	▶	▲	▼
20	SP	!	"	#	\$	%	&	'	()	*	,	-	.	/	.	.
30	Ø	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	.	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[]	.	.	.
60	.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z
80	あ	い	う	え	お	か	き	く	け	こ	さ	し	す	せ	そ	た
90	ち	つ	て	と	な	に	ぬ	ね	の	は	ひ	ふ	へ	ほ	ま	み
A0	む	め	も	や	ゆ	よ	ら	り	る	れ	ろ	わ	を	ん	あ	い
B0	う	え	お	や	ゆ	よ	つ	、	。	一	「	」
C0	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ	タ
D0	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ	ミ
E0	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ヲ	ン	ア	イ
F0	ウ	エ	オ	ヤ	ユ	ヨ	ツ	、	。	一	、	。

[2] Using MESS_OUT (Generic V-RAM Output)

All messages displayed on the screen by the system must use MESS_OUT. MESS_OUT is called during interrupt 1 (SYSTEM_INT1) by the system, and even for the game, if this is called at an appropriate area, it can be used.

The output method is as follows: the addresses of output data are stored in the MESS_BUFFER (10FF00H), and when MESS_OUT is called these are output all at once.

(1) Output Data Format

There are two types of data output with MESS_OUT: a series of control commands and the actual output data accessed by those commands. The control command is made up of one word, the lower byte is used for the command number, and the upper byte or the next (long) word is used for the parameters. Since the control commands are made in word units, the starting address must start from an even address.

Command 0	Command End Code This means that a series of control commands has ended. Example DC.W 0 -- End of Command
Command 1	Data Format This specifies the format of the data to be output. Command upper byte d0 0=End code specification 1= Data size specification d1 0=Byte data 1= Word data Next word With byte data Upper byte Upper byte data that is commonly output Lower byte End code or data size With word data End code or data size

Command
1
(Continued)

Example 1: DC.W 0001H, 12FFH

Byte data
End code specification
Command 1

In this case, the output data is made up of only the lower byte, and the upper byte is 12H as specified. Also, FFH is the data end code. Therefore, FFH cannot be placed in the lower byte.

Example 2: DC.W 0101H, 3410H

Byte data
Data size specification
Command 1

In this case, the output data is made up of only the lower byte, and the upper byte is 34H as specified. Also, the data size is constant at 16 bytes, and H through FFH cannot be placed in the lower byte.

Example 3: DC.W 0201H, 0000H

Word data
End code specification
Command 1

In this case, the output data is made up in words, so the upper byte can be changed in word units.

Example 4: DC.W 0301H, 0020H

Word data
Data size specification
Command 1

The data size is constant at 32 bytes.

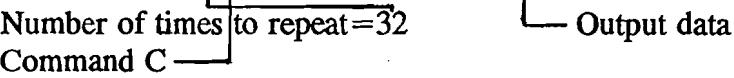
The format set here is valid until it is specified again using command 1, so it is not necessary to specify this for every output if they are in the same format.

Command 2	<p>The setting of the automatic increment amount Command upper byte: Increment amount (Character expansion to 16 bits and output to 3C0004H)</p> <p>Example: DC.W 2002H</p> <p>Automatic increment 20H</p> 
Command 3	<p>Output V-RAM address The next word is output directly to 3C0000H.</p> <p>Example: DC.W 3, 7102H</p> <p>Command 3</p> 

Command	<p>Output data address</p> <p>The starting address of the output data is specified in the next long word. (If word data was specified in command 1, please start from an even address.)</p> <p>The data is actually output with this command.</p> <p>Example 1: DC.W <u>1, FFH</u></p>  <table border="0"> <tr> <td style="vertical-align: top; padding-right: 20px;">DC.W 4</td> <td>Command 4</td> </tr> <tr> <td style="vertical-align: top; padding-right: 20px;">DC.L MESSAGE1</td> <td>Starting address of data</td> </tr> </table> <p>.....</p> <p>MESSAGE1: Output data</p> <table border="0"> <tr> <td style="vertical-align: top; padding-right: 20px;">DC.B</td> <td>'SAMPLE,' FFH</td> </tr> </table> <p>Example 2: DC.W <u>301H, 3</u></p>  <table border="0"> <tr> <td style="vertical-align: top; padding-right: 20px;">DC.W 4</td> <td>Command 4</td> </tr> <tr> <td style="vertical-align: top; padding-right: 20px;">DC.L MESSAGE2</td> <td>Starting address of data</td> </tr> </table> <p>.....</p> <p>MESSAGE2: Output data</p> <table border="0"> <tr> <td style="vertical-align: top; padding-right: 20px;">DC.W</td> <td>130H, 1131H, 3033H</td> </tr> </table>	DC.W 4	Command 4	DC.L MESSAGE1	Starting address of data	DC.B	'SAMPLE,' FFH	DC.W 4	Command 4	DC.L MESSAGE2	Starting address of data	DC.W	130H, 1131H, 3033H
DC.W 4	Command 4												
DC.L MESSAGE1	Starting address of data												
DC.B	'SAMPLE,' FFH												
DC.W 4	Command 4												
DC.L MESSAGE2	Starting address of data												
DC.W	130H, 1131H, 3033H												

Command 5	<p>Address increment amount Add the amount specified in the next word to the V-RAM address output previously, and this is used as the output address. (Output to 3C0000H)</p> <p>Example:</p> <table border="0"> <tbody> <tr> <td>DC.W</td><td>1, 0FFH</td><td>Data format Byte data End code 0FFH</td></tr> <tr> <td>DC.W</td><td>3, 7202H</td><td>V-RAM address (Fix character)</td></tr> <tr> <td>DC.W</td><td>102H</td><td>Automatic increment 1 (Fix character vertical output)</td></tr> <tr> <td>DC.W</td><td>4</td><td></td></tr> <tr> <td>DC.L</td><td>DATA1</td><td>Data output</td></tr> <tr> <td>DC.W</td><td>5, 20H</td><td>one line to the right</td></tr> <tr> <td>DC.W</td><td>4</td><td></td></tr> <tr> <td>DC.L</td><td>DATA2</td><td></td></tr> <tr> <td>DC.W</td><td>5, 41H</td><td>two lines to the right and one line down</td></tr> <tr> <td>DC.W</td><td>4</td><td></td></tr> <tr> <td>DC.L</td><td>DATA3</td><td></td></tr> <tr> <td>DC.W</td><td>0</td><td>Command end code</td></tr> </tbody> </table>	DC.W	1, 0FFH	Data format Byte data End code 0FFH	DC.W	3, 7202H	V-RAM address (Fix character)	DC.W	102H	Automatic increment 1 (Fix character vertical output)	DC.W	4		DC.L	DATA1	Data output	DC.W	5, 20H	one line to the right	DC.W	4		DC.L	DATA2		DC.W	5, 41H	two lines to the right and one line down	DC.W	4		DC.L	DATA3		DC.W	0	Command end code
DC.W	1, 0FFH	Data format Byte data End code 0FFH																																			
DC.W	3, 7202H	V-RAM address (Fix character)																																			
DC.W	102H	Automatic increment 1 (Fix character vertical output)																																			
DC.W	4																																				
DC.L	DATA1	Data output																																			
DC.W	5, 20H	one line to the right																																			
DC.W	4																																				
DC.L	DATA2																																				
DC.W	5, 41H	two lines to the right and one line down																																			
DC.W	4																																				
DC.L	DATA3																																				
DC.W	0	Command end code																																			
Command 6	<p>Resume the output of data previously output</p> <p>Example:</p> <table border="0"> <tbody> <tr> <td>DC.W 4</td> <td></td> <td></td> </tr> <tr> <td>DC.L DATA4</td> <td>Data output (1)</td> <td></td> </tr> <tr> <td>DC.W 5, 20H</td> <td>Address change</td> <td></td> </tr> <tr> <td>DC.W 6</td> <td>Output the rest (2)</td> <td></td> </tr> <tr> <td>.....</td> <td></td> <td></td> </tr> </tbody> </table> <p>DATA4:</p> <table border="0"> <tbody> <tr> <td>DC.B 0, 1, 2, 3, 4, <u>0FFH</u> (1)</td> <td style="text-align: right;">End code</td> </tr> <tr> <td>DC.B 5, 6, 7, 0FFH (2)</td> <td></td> </tr> </tbody> </table>	DC.W 4			DC.L DATA4	Data output (1)		DC.W 5, 20H	Address change		DC.W 6	Output the rest (2)				DC.B 0, 1, 2, 3, 4, <u>0FFH</u> (1)	End code	DC.B 5, 6, 7, 0FFH (2)																		
DC.W 4																																					
DC.L DATA4	Data output (1)																																				
DC.W 5, 20H	Address change																																				
DC.W 6	Output the rest (2)																																				
.....																																					
DC.B 0, 1, 2, 3, 4, <u>0FFH</u> (1)	End code																																				
DC.B 5, 6, 7, 0FFH (2)																																					

Command 7	<p>Directly define the output data</p> <p>Example:</p> <table border="0" style="width: 100%;"> <tr> <td style="text-align: right; vertical-align: top;">DC.W 7</td><td style="text-align: right; vertical-align: top;">Direct data definition</td></tr> <tr> <td style="text-align: right; vertical-align: top;">DC.B '<u>ABCD</u>', <u>OFFH</u>, 0</td><td></td></tr> <tr> <td colspan="2" style="text-align: center; padding-top: 10px;"> </td></tr> <tr> <td colspan="2" style="text-align: center; padding-top: 10px;"> One-byte dummy so that the next command will be an even address (This is necessary if the total data size is odd, but not necessary if the size is even.) </td></tr> </table>	DC.W 7	Direct data definition	DC.B ' <u>ABCD</u> ', <u>OFFH</u> , 0				One-byte dummy so that the next command will be an even address (This is necessary if the total data size is odd, but not necessary if the size is even.)																
DC.W 7	Direct data definition																							
DC.B ' <u>ABCD</u> ', <u>OFFH</u> , 0																								
One-byte dummy so that the next command will be an even address (This is necessary if the total data size is odd, but not necessary if the size is even.)																								
Command 8	<p>Fix 8x16 dot-matrix character horizontal output</p> <p>The top-level output upper-byte data is placed in the upper byte of the command. The lower-output byte data is placed from the next byte on. (The end code is OFFH.)</p> <p>The bottom level will output the data in the first address following the upper byte.</p> <p>Example:</p> <table border="0" style="width: 100%;"> <tr> <td style="text-align: right; vertical-align: top;">DC.W 3, 7101H</td> <td style="text-align: right; vertical-align: top;">Output address</td> </tr> <tr> <td style="text-align: right; vertical-align: top;">DC.W 108H</td> <td style="text-align: right; vertical-align: top;">Command 8</td> </tr> <tr> <td colspan="2" style="text-align: center; padding-top: 10px;"> </td> </tr> <tr> <td colspan="2" style="text-align: center; padding-top: 10px;"> V-RAM after output </td> </tr> <tr> <td style="text-align: center; vertical-align: middle; padding-bottom: 10px;"> ADDRESS DATA </td><td style="text-align: center; border: 1px solid black; padding: 5px;"> 7010H 0130H </td><td style="text-align: center; border: 1px solid black; padding: 5px;"> 7030H 0131H </td><td style="text-align: center; border: 1px solid black; padding: 5px;"> 7050H 0132H </td><td style="text-align: center; border: 1px solid black; padding: 5px;"> 7070H 0133H </td></tr> <tr> <td style="text-align: center; vertical-align: middle; padding-top: 10px;"> ADDRESS DATA </td><td style="text-align: center; border: 1px solid black; padding: 5px;"> 7011H 0230H </td><td style="text-align: center; border: 1px solid black; padding: 5px;"> 7031H 0230H </td><td style="text-align: center; border: 1px solid black; padding: 5px;"> 7051H 0232H </td><td style="text-align: center; border: 1px solid black; padding: 5px;"> 7071H 0233H </td></tr> <tr> <td colspan="5" style="text-align: center; padding-top: 20px;"> If the output data size is odd, one byte must be placed after the end code as a dummy. </td></tr> </table>	DC.W 3, 7101H	Output address	DC.W 108H	Command 8			V-RAM after output		ADDRESS DATA	7010H 0130H	7030H 0131H	7050H 0132H	7070H 0133H	ADDRESS DATA	7011H 0230H	7031H 0230H	7051H 0232H	7071H 0233H	If the output data size is odd, one byte must be placed after the end code as a dummy.				
DC.W 3, 7101H	Output address																							
DC.W 108H	Command 8																							
V-RAM after output																								
ADDRESS DATA	7010H 0130H	7030H 0131H	7050H 0132H	7070H 0133H																				
ADDRESS DATA	7011H 0230H	7031H 0230H	7051H 0232H	7071H 0233H																				
If the output data size is odd, one byte must be placed after the end code as a dummy.																								

Command 9	<p>Fix 8x16 dot-matrix Japanese character horizontal output The format is the same as command 8. However, codes 0-1FH are changed to hiragana with voiced sound symbols, and code 60H-7FH is changed to katakana with voiced sound symbols. (Exclusive for common Fix TYPE 1) The automatic incrementing function is set in 20H for commands 8 and 9.</p>
Command A and B	<p>Subcommand calling (A) and return (B) The pointer is moved to the address specified by the long word after command A, and command B returns the pointer to the original location. Up to four levels of these subcommands can be nested. However, command 0 (end command) cannot be used within the subcommands.</p> <p>Example:</p> <pre> DC.W 0AH Call DC.L SUB_COMMAND DC.W 0 SUB_COMMAND: DC.W 7 DC.B 'XYZ', 0FFH DC.WS 0BH Return </pre>
Command C	<p>Continuous output of the same data The data specified in the next word is output continuously the number of times specified by the upper byte of the command.</p> <p>Example:</p> <pre> DC.W 200CH, 0020H </pre>  <p>0020H is output continuously 32 times.</p>

Command D	<p>Data increment output</p> <p>The data specified in the next word is continuously output by being incremented the number of times specified by the upper byte of the command. Only the lower byte is incremented for the data, and the upper byte is not increased.</p> <p>Example:</p> <p>DC.W 200DH, <u>1180H</u></p> <p>The data is output in the sequence 1180H, 1181H, 1182H, ..., 119FH.</p>
--------------	--

(2) Output Data Set

Set the starting address of the output data (control command) in long word from the address indicated by **MESS_POINT** (10FDBEH, L). Then, increment **MESS_POINT** by four. If **MESS_OUT** is going to be used for interrupts, set **MESS_BUSY** (10FDC2H, B) to a value other than 0 so that processing does not coincide. By doing this even if **MESS_OUT** has been called, processing can be skipped. Also, even when simply accessing the V-RAM, if **MESS_OUT** is called, the V-RAM address may change. Be sure to use **MESS_BUSY**.

MESS_POINT points to **MESS_BUFFER** (10FF00H) directly after the output for **MESS_OUT** has been finished (when no data is set), and since the buffer size is 256 bytes, be sure not to exceed this size.

When 0 is set for the control-command address, a control command can be placed from the next address in the **MESS_BUFFER**.

Sample Program

```
ADDQ.B #1,MESS_BUSY ;busy flag set
MOVE.L MESS_POINT,A0 ;pointer get
MOVE.L #0000H,(A0)+ ;buffer direct command
MOVE.W #0003H,(A0)+ ;COMMAND 3
MOVE.W #7318H,(A0)+ ; output v-ram address
MOVE.W #0301H,(A0)+ ;COMMAND 1
MOVE.W #0001H,(A0)+ ; 1 word data
MOVE.W #0007H,(A0)+ ;COMMAND 7
MOVE.W 100000H,(A0)+ ; output data
MOVE.W #0,(A0)+ ;COMMAND 0 (command end)
MOVE.L #MESSAGE1,(A0)+ ;next command address set
MOVE.L A0,MESS_POINT ;pointer set
SUBQ.B #1,MESS_BUSY ;busy flag clear
```

MESSAGE1:

```
DC.W 0001H ;COMMAND 1
DC.W 00FFH ; byte data,end code type
            ; data hi-byte 0,end code 0FFH
DC.W 2002H ;COMMAND 2
            ; auto increment 20H
DC.W 0003H ;COMMAND 3
DC.W 7024H ; output v-ram address
DC.W 0004H ;COMMAND 4
DC.L MS1   ; output data address
DC.W 0005H ;COMMAND 5
DC.W 0001H ; 1 line down
DC.W 0006H ;COMMAND 6
            ; data continue output
DC.W 0005H ;COMMAND 5
DC.W 0001H ; 1 line down
DC.W 0007H ;COMMAND 7
DC.B 'MESSAGE 3 '
DC.B 0FFH   ; output data (8 byte)
DC.B 0       ; data end code
            ; dummy for even address
```

```

DC.W 0005H ;COMMAND 5
DC.W 0001H ; 1 line down
DC.W 0108H ;COMMAND 8
                ; data hi-byte 1 & 2
DC.B 'ABCDE',0FFH ; output data & end code
DC.W 0005H ;COMMAND 5
DC.W 0001H ; 1 line down
DC.W 0109H ;COMMAND 9
                ; data hi-byte 1 & 2
DC.B 0,1,2,3,4,0FFH ; output data & end code
DC.W 000AH ;COMMAND A
DC.L SUB_MESS ; sub command address
DC.W 000AH ;COMMAND A
DC.L SUB_MESS ; sub command address
DC.W 000AH ;COMMAND A
DC.L SUB_MESS ; sub command address
DC.W 000AH ;COMMAND A
DC.L SUB_MESS ; sub command address
DC.W 0000H ;COMMAND 0
                ; command end code

MS1:
DC.B 'MESSAGE 1',0FFH ;output data
DC.B 'MESSAGE 2',0FFH ; 1st output data
                        ; 2nd output data

SUB_MESS:
DC.W 280CH ;sub command
DC.W 0020H ;COMMAND C
DC.W 100DH ; 0020H 40 times output
DC.W 0500H ;COMMAND D
DC.W 000BH ; output 0500H,0501H,...,050FH
                ;COMMAND B
                ; sub command return

```

If an error is detected during data output for MESS_OUT (such as an overflow of the command number), then the output is stopped at that point.

--- 6 --- MEMORY CARD

[1] Game Data Structure on the Card

The game data on the memory card is composed of the following four items:

Game Number	(Game ID placed in 108H)
Sub-Number	(16 types (0 through 15) for each game)
Data Title	(20 bytes)
Game Data	(64 byte units)

The memory card is divided into directories, FAT and data, as with floppy disks, and these are controlled by the system.

The game number and subnumber are placed in the directory. Multiple data can be placed for one game using subnumbers (for example, data for one-player games and two-player games).

The game data is controlled in page units, and one page is 64 bytes. However, the data title takes up 20 bytes for the first page, so only 44 bytes can be used for data. (The full 64 bytes can be used from the second page onward.)

[2] Memory Card Access

Accessing the memory card can be done by setting the parameters in the workspace for the card from 10FDC4H, and calling CARD (C00468H). Please do not access the card directly from the game besides special cases.

The command result is returned in CARD_ANSWER (10FDC6H, B). "0" means that the command has been successfully executed, and other values signify that some error has occurred, and the value represents the type of error.

Command	Format	
Input	CARD_COMMAND byte	0
This formats the memory card. All data in the card will be lost.		

Command	Data search	
Input	CARD_COMMAND byte	1
	CARD_FCB word	Game number
Output	CARD_SUB word	If bit n is "0," the data does not exist for sub-number n of the game number given by CARD_FCB. "1" means the data exists.
	Example: Call CARD_FCB by setting the parameter as 1234H, and if the result CARD_SUB is 011000010000101B, the data in the subnumbers 0, 2, 7, 13, and 14 exists for game number 1234H.	

Command	Data Load	
Input	CARD_COMMAND byte	2
	CARD_FCB word	Game number
	CARD_SUB byte	Subnumber
	CARD_START 1 word	Data-load address
	CARD_SIZE word	Load data size
Output	The smaller size is selected between the data size on the card and the size specified by CARD_SIZE. The game data is loaded up to the smaller size selected, from the address indicated by CARD_START. The game title is set in the beginning 20 bytes.	

Command	Data Save	
Input	CARD_COMMAND byte	3
	CARD_FCB word	Game number
	CARD_SUB byte	Subnumber
	CARD_START 1 word	Data-save address
	CARD_SIZE word	Save data size
	<p>Set the data title in common Fix code (TYPE1 for domestic and TYPE3 for overseas) for the beginning 20 bytes of save data. This is used for directory displays, etc., so please try to show the game title and the necessary data.</p> <p>(For example,</p> <ul style="list-style-type: none"> "Baseball VS. 1P" "Baseball VS. 2P" "Baseball Tournament" etc.) <p>The starting address and data size must be even values for data load and data save.</p>	

Command	Data Delete	
Input	CARD_COMMAND byte	4
	CARD_FCB word	Game number
	CARD_SUB byte	Subnumber
	This deletes the specified data.	

Command	Reading the data title	
Input	CARD_COMMAND byte	5
	CARD_START 1 word	Data title/set address
	CARD_SIZE word	Directory number
Output	If CARD_SIZE is n, the data title in the nth game data in the card directory is set in the 20 bytes beginning with the address indicated by CARD_START. Other output is as follows:	
	CARD_FCB word	Game number
	CARD_SUB byte	Subnumber
	CARD_SIZE word	+1
	All data titles on the card can be observed by first setting "0" to the CARD_SIZE and by calling this routine until CARD_ANSWER reaches 82H (no data).	

Command	User Name Save	
Input	CARD_COMMAND byte	6
	CARD_START 1 word	User name address
Command	User Name Load	
Input	CARD_COMMAND byte	7
	CARD_START 1 word	User name/load address
	The user name (16 bytes) can be specified for every memory card. (All are 20H during formatting.) The domestic format is Fix code TYPE1 and the overseas format is TYPE3. Saving and loading is performed from the address indicated in CARD_START.	

Note About Data Loading

The system does not check the game data, so please use the data after checking the contents for each game. Please do not allow for errors to occur from data corruption or improper changes made by the user. We prohibit running the program directly on the loaded data and on the memory card.

[3] Error Codes

The result of the call made to CARD is returned in CARD_ANSWER (10FDC6H, B).

CODE	Description
00H	Normal completion
80H	The card has not been inserted.
81H	The card has not been formatted.
82H	The specified data does not exist.
83H	FAT irregularity
84H	Data full (insufficient data area). When saving data, if the same game number and sub-number already exist, the old data is deleted after the new data is saved.
85H	Write disable. Or, a ROM card.

[4] Error Processing

Some of the error handling for errors caused by the card may be performed by the system by calling CARD_ERROR (C0046EH). Call CARD_ERROR by sending the exact parameters (including CARD_ANSWER) returned from the card.

The result of error handling is prompted with CARD_ANSWER. If "0" is returned, the previous command had been successfully executed after error handling, and other values denotes that the command cannot not be executed.

During error handling the system handles the screen display, so set MESS_BUSY (10FDC2H, B) to "0." The screen goes back to the previous display after the return from error handling.

** Details of Error Handling **

ERROR CODE	CARD _ COMMAND							
	Ø	1	2	3	4	5	6	7
8 0 H	-	-	-	-	-	-	-	-
8 1 H	-	-	A	B	-	-	B	-
8 2 H	-	-	-	-	-	-	-	-
8 3 H	-	-	A	B	-	-	-	-
8 4 H	-	-	-	C	-	-	-	-
8 5 H	A	-	A	A	A	-	A	A

- A This displays the error status.
- B After verifying that the card can be formatted, the card is formatted and the command is executed. (However, for commercial-use machines, formatting is done automatically without confirmation.)
- C All data titles in the card are displayed, and allowing the user to select the data for deletion to obtain more memory. Then, the data is saved. (This procedure can be canceled.)
- Nothing is performed.

If the same data exists when the card is full and the size of the data to save is the same as the already existing data, save the new data after deleting the old data. This is to prevent the players from having to perform extra operations for error handling, as much as possible.

--- 7 --- SOUND PROGRAM

The program specification for the sound CPU (Z80) can be done freely, but there are some sound codes that are standard specifications.

CODE 1 Slot-Switching

Cancel all sounds and enter an infinite loop in the work RAM after enabling the NMI. Return "1" to the 68000 right before entering the infinite loop.

The system waits for "1" to be returned from the Z80, and slot switching is done. Returning a "1" to the 68000 is prohibited if a sound code other than "1" is received. (Always return a code other than "1." If "1" is left carelessly, slot-switching is done before the Z80 is ready.)

Code 3 Z80 Reset

Perform the same process as the hardware reset. Complete the reset process within 100m seconds.

Code 2 Eye-Catcher Sound

This is used only when the eye-catcher is done by the system. (When the main CPU's memory location 114H is "0") Allow this to be executed in any condition once code 3 has been received.

[Note]

Allow codes 1 and 3 to be received with NMI as much as possible. In that case, the next NMI is not executed unless the Z80 executes RETN, so be sure to execute RETN. Also, allows these two codes to be executed in any condition.

--- 8 --- USING THE MONITOR PROGRAM

A simple monitor function is provided in the system ROM for development. The monitoring is started up from specific-exception handling. (Refer to game ID for details.) In particular, monitoring starts for the bus error when jumper J1 on the board is short-circuited.

Monitor Screen

Error Name			
Access Information	Memory Access Address	Operation Word	
Only for bus errors and address errors			
D0	D1	D2	D3
D4	D5	D6	D7
A0	A1	A2	A3
A4	A5	A6	A7 (SSP)
USP	SR	PC	
Monitor Command			

Monitor Commands

DISPLAY MEMORY	Displays the memory contents
DISPLAY V-RAM	Displays the V-RAM contents
MODIFY MEMORY	Modifies the memory
MODIFY V-RAM	Modifies the memory
MODIFY REGISTER	Modifies the registers
TRACE	Trace mode (step-by-step execution)
RUN	Runs the program (exits the monitor mode)

The command switches by using button A, and the command is executed with button C. The horizontal and vertical movements of the joystick change the parameters. The memory is accessed in words.

--- 9 --- POINTS TO NOTE WHEN CREATING A GAME

1. Call SYSTEM_IO every 1/60 of a second.
2. Most I/O-related processing are performed by the system, so please do not access the I/O directly with the game software except for special cases. The addresses that are not defined by hardware may be phantom addresses, so be sure never to use these addresses.
3. Be sure never to write anything in the system work area (10F300H-10FFFFH) except for the area to be set by the game software and the backup RAM (D00000H-D0FFFFH).
4. Be careful in using flags for the game starting processes, since these processes are especially known to be problematic.
5. Please do not place any important displays in the 32 dots at both the left and right sides and eight dots at both the top and bottom on the screen. (These areas may not be displayed, depending on the monitor.)

--- 10 --- SAMPLE PROGRAM

```
ORG    0

DC.L  10F300H      ;Reset SSP
DC.L  0C00402H      ;Reset PC
DC.L  0C00408H      ;Bus Error=monitor entry
DC.L  0C0040EH      ;Address Error
DC.L  0C00414H      ;Illegal Instruction
DC.L  ZD_ENTRY      ;Zero Divide
DC.L  CHK_ENTRY      ;CHK Instruction
DC.L  TARPV_ENTRY      ;TRAPV Instruction

DC.L  0C0041AH      ;Privilege Violation
DC.L  0C00420H      ;Trace
DC.L  L1010_ENTRY      ;Line 1010 Emulator
DC.L  L1111_ENTRY      ;Line 1111 Emulator
DC.L  0C00426H      ;Unassigned
DC.L  0C00426H      ;
DC.L  0C00426H      ;
DC.L  0C0042CH      ;Uninitialized Interrupt

DC.L  0C00426H      ;Unassigned
DC.L  0C00426H      ;

DC.L  0C00432H      ;Spurious Interrupt
DC.L  INT1          ;Interrupt 1
DC.L  INT2          ;Interrupt 2
DC.L  INT3          ;Interrupt 3
DC.L  INT4          ;Interrupt 4
DC.L  INT5          ;Interrupt 5
DC.L  INT6          ;Interrupt 6
DC.L  INT7          ;Interrupt 7
```

ORG	100H	
DC.B	'NEO-GEO', 0	; (100H) ID 1
DC.W	6789H	; (108H) Game Code
DC.L	800000H	; (10AH) Rom Size (4Mbit)
DC.L	100000H	; (10EH) Backup Start
DC.W	1000H	; (112H) Backup Size
DC.B	0	; (114H) Use System.Eye-Catch
DC.B	0	; (115H) Eye-Catch ch hi-byte
DC.L	JAPAN_DATA	; (116H) Soft Dip Data
DC.L	USA_DATA	; (11AH) "
DC.L	EUROPE_DATA	; (11EH) "
JMP	USER	; (122H)
JMP	PLAYER_START	; (128H)
JMP	DEMO_END	; (12EH)
JMP	COIN_SOUND	; (134H)
ORG	182H	
DC.L	ID2	
ID2:		
DC.W	07600H, 04A6DH, 00A14H, 06600H	
DC.W	0003CH, 0206DH, 00A04H, 03E2DH	
DC.W	00A08H, 013C0H, 00030H, 00001H	
DC.W	03210H, 00C01H, 000FFH, 0671AH	
DC.W	03028H, 00002H, 0B02DH, 00ACEH	
DC.W	06610H, 03028H, 00004H, 0B02DH	
DC.W	00ACFH, 06606H, 0B22DH, 00AD0H	
DC.W	06708H, 05088H, 051CFH, 0FFD4H	
DC.W	03607H, 04E75H, 0206DH, 00A04H	
DC.W	03E2DH, 00A08H, 03210H, 0E049H	
DC.W	00C01H, 000FFH, 0671AH, 03010H	
DC.W	0B02DH, 00ACEH, 06612H, 03028H	
DC.W	00002H, 0E048H, 0B02DH, 00ACFH	
DC.W	06606H, 0B22DH, 00AD0H, 06708H	
DC.W	05888H, 051CFH, 0FFD8H, 03607H	
DC.W	04E75H	

JAPAN_DATA:

DC.B 0CAH,0EDH,76H,0E8H,20H,03H,0F9H,0E0H
DC.B 20H,20H,20H,20H,20H,20H,20H,20H
; Sample Game ,

DC.W 300H ;Play time 3 min
DC.W 300H ;Continue Play 3 min
DC.B 3 ;Hero 3
DC.B 100 ;Continue non-limit

DC.B 03H ;Bonus Type
DC.B 14H ;Bonus Rate
DC.B 38H ;Difficulty
DC.B 02H ;Demo Sound
DC.B 00H ;Not used
DC.B 00H
DC.B 00H
DC.B 00H
DC.B 00H
DC.B 00H
DC.B 00H

DC.B 0C8H,0B7H,0F9H,0E0H,06H,07H,08H,09H
DC.B 20H,20H,20H,20H ; Game Time

DC.B 0C9H,0EDH,0D2H,0EFH,0D5H,0F4H,0F9H
DC.B 06H,07H,08H,09H,20H ; Extended Play Time

DC.B 0AH,0BH,0CH,0DH,0EH,0FH
DC.B 20H,20H,20H,20H,20H,20H ; Remaining Lives

DC.B 0C9H,0EDH,0D2H,0EFH,0D5H,0F4H,0F9H
DC.B 20H,20H,20H,20H,20H ; Continue

DC.B 'BONUS TYPE '
DC.B 'NO BONUS '
DC.B 'EVERY BONUS '
DC.B 'SECOND BONUS'

```

DC.B    'BONUS RATE '
DC.B    '20000/10000 '
DC.B    '30000/10000 '
DC.B    '50000/30000 '
DC.B    '100000/50000'

DC.B    00H,01H,02H,03H,04H,05H
DC.B    20H,20H,20H,20H,20H,20H ; Difficulty Level
DC.B    'LEVEL 1 '
DC.B    'LEVEL 2 '
DC.B    'LEVEL 3 '
DC.B    'LEVEL 4 '
DC.B    'LEVEL 5 '
DC.B    'LEVEL 6 '
DC.B    'LEVEL 7 '
DG.B    'LEVEL 8 '

DC.B    0D2H,0F7H,0E7H,0CAH,0C2H,0EDH,0D3H,0F7H
DC.B    20H,20H,20H,20H           ; Demo Sound
DC.B    0EH,0FH,0A7H,20H,20H,20H
DC.B    20H,20H,20H,20H,20H,20H ; with
DC.B    14H,15H,8BH,20H,20H,20H
DC.B    20H,20H,20H,20H,20H,20H ; without

```

USA_DATA:

EUROPE_DATA:

```

DC.B    'SAMPLE GAME '

DC.W    300H          ;Play time 3 min
DC.W    200H          ;Continue Play 3 min
DC.B    3              ;Hero 3
DC.B    0              ;Continue without

DC.B    03H          ;Bonus Type
DC.B    14H          ;Bonus Rate
DC.B    38H          ;Difficulty
DC.B    02H          ;Demo Sound
DC.B    00H          ;Not used
DC.B    00H

```

DC.B	00H
DC.B	'PLAY TIME'
DC.B	'CONT. TIME'
DC.B	'HERO'
DC.B	'CONTINUE'
DC.B	'BONUS TYPE'
DC.B	'NO BONUS'
DC.B	'EVERY BONUS'
DC.B	'SECOND BONUS'
DC.B	'BONUS RATE'
DC.B	'20000/10000'
DC.B	'30000/10000'
DC.B	'50000/30000'
DC.B	'100000/50000'
DC.B	'DIFFICULTY'
DC.B	'LEVEL 1'
DC.B	'LEVEL 2'
DC.B	'LEVEL 3'
DC.B	'LEVEL 4'
DC.B	'LEVEL 5'
DC.B	'LEVEL 6'
DC.B	'LEVEL 7'
DC.B	'LEVEL 8'
DC.B	'DEMO SOUND'
DC.B	'WITH'
DC.B	'WITH OUT'

```

        MOVEQ.L #0,D0          ; Get request number
        MOVE.B USER_REQUEST,D0
        ADD.W D0,D0
        ADD.W D0,D0
        MOVE.L REQUEST_VECTOR(PC,D0.W),A0
        JMP A0

REQUEST_VECTOR:
        DC.L    POWER_ON      ;Power on initialize
        DC.L    SYSTEM_RETURN ;Eye-Catch not used
        DC.L    GAME           ;Demo game,game
        DC.L    TITLE          ;title only

POWER_ON:
        JSR    INIT_RANKING ;Ranking initialize
        JMP    SYSTEM_RETURN

GAME:
        JSR    INIT_GAME     ;Display & work initialize
        JMP    DEMO_MAIN     ;Demo game main

TITLE:
        JSR    INIT_GAME     ;Display & work initialize
        JMP    TITLE_MAIN    ;Title display main

INIT_GAME:
        ADDQ.B #1,MESS_BUSY   ;MESS_OUT disable
        BCLR.B #7,SYSTEM_MODE ;To system mode
        MOVE.W #2000H,SR      ;Interrupt all enable
        JSR    DISPLAY_INIT   ;Display all initialize
        JSR    WORK_INIT      ;Work area all initialize

        SUBQ.B #1,MESS_BUSY   ;MESS_OUT enable
        BSET.B #7,SYSTEM_MODE ;To game mode
        RTS

```

Confidential



Neo-Geo Development Tool Catalogue

- Neo-Geo Development Tool List

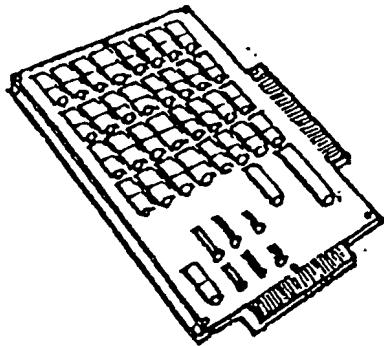
Product Name	Reference
1. Development Software PCB (ROM Type)	For both home and commercial use/Character 128 M/Can use 1 M or 4 M EPROM
2. Development Software PCB (IC Card Type)	For both home and commercial use/Character 128 M/Uses 4 M or 32 M IC card.
3. Sample Cassette Board (for Commercial-Use)	Mass-produced type/Can use 16 M EPROM * Note: Product availability may be affected by stock on hand. Contact SNK for availability information.
4. Sample Cassette Board (for Home-Use)	Mass-produced type/Can use 16 M EPROM * Note: Product availability may be affected by stock on hand. Contact SNK for availability information.
5. Development Neo-Geo Main Unit	Can develop for both home and commercial use/One controller, AC adapter, and AV cable are accessories.
6. Controller	Controller exclusively for Neo-Geo
7. RGB 21-Pin Cable	RGB output cable exclusively for Neo-Geo main unit
8. Memory Card (for Games)	2 K-byte (commercial product JEIDA Version 4.1 SRAM card can also be used) Game play data storage card
9. Art Box (Character Development Unit)	RGB cable for the monitor as an accessory * The mouse is not an accessory. (A mouse for PC98 can be used; DSUB 9-pin.)
10. SCSI Hard Drive (For Character Development)	SCSI cable for the Art Box is an accessory.
11. Sound Program (for Neo-Geo Development)	Software for PC 9800

* Not available from SNK.

- Functions

Development Software PCB (ROM Type)	This is the development board, which uses the EPROM. It corresponds to all programs, characters, and sounds. Note: Refer to the "Development EPROM Example" for the EPROM to be used.
Development Software PCB (IC Card Type)	This is the development board, which uses an IC card. It corresponds to all programs, characters, and sounds. Note: Refer to the "Development IC card Example" for the IC card to be used.
Development Neo-Geo Main Unit	This is a product for which the CPU (68,000) of the home-use Neo-Geo board is used as a pin socket, and for which the ICE can be used. (ICE: In-Circuit Emulator)
Art Box (Character Development Unit)	This is a design tool for character development. The hard drive is used as an external memory device. RS232C, RGB output, bus mouse, Centronics connector, and IC card output are the I/Os. (JEIDA Version 3, 4 can be used as the IC card output.)
RGB 21-Pin Cable	RGB cable exclusively used for the Neo-Geo

- Description of Development Tools

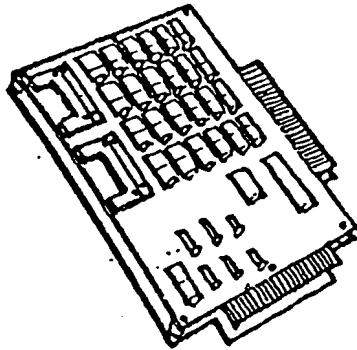


1. Development Software PCB (ROM Type)

This is the development board, which uses EPROMs*. It corresponds to all programs, characters, and sounds, and is used for both home and commercial use. It allows up to 128 M characters.

It can use 1 M or 4 M EPROM*.

* Not available from SNK.



2. Development Software PCB (IC-card type)

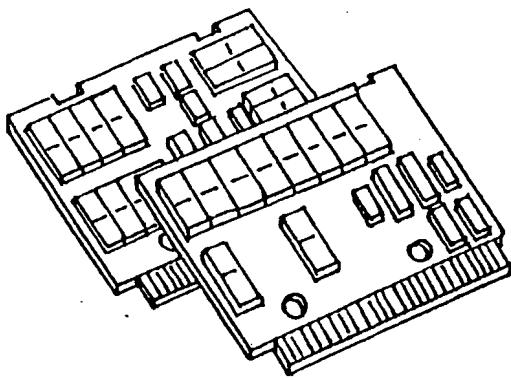
This is the development board, which uses the IC card*. It corresponds to all programs, characters, and sounds, and is used for both home and business. It allows up to 128 M characters.

It can use 4 M and 32 M IC card*.

(Can use JEIDA Version 3 or 4)

Note: Only the PCM uses the 4M EPROM*.

*Not available from SNK.

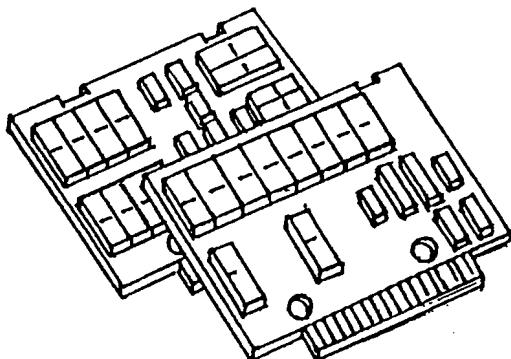


3. Sample Cassette Board (Business-use)

Used for location testing.

Mass-produced type, can use 16 M EPROM*.

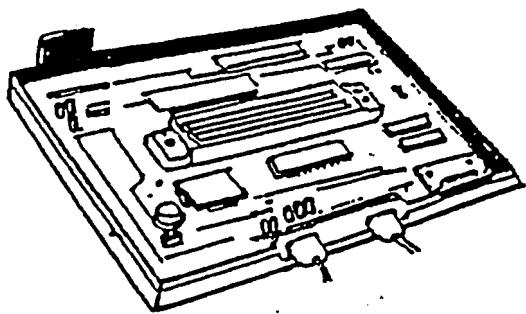
*Not available from SNK.



4. Sample Cassette Board (Home-Use)

Mass-produced type. Can use 16 M EPROM*.

*Not available from SNK.



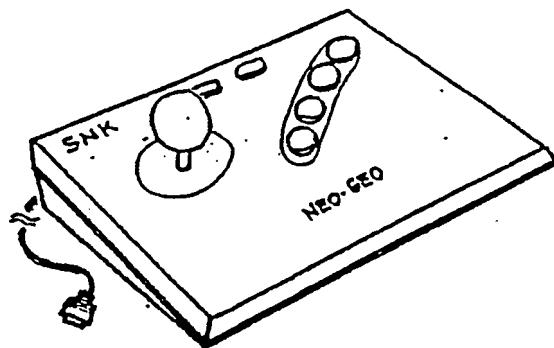
5. Development Neo-Geo Main Unit

This is a product for which the CPU (Z80, 68000) of the home-use Neo-Geo board is used as a pin socket, and for which the ICE can be used.

It is equipped with a development system ROM Possible for different country versions, commercial-and home-use check.

Can develop for both home and commercial use.
One controller, AC adapter, and AV cable are accessories.

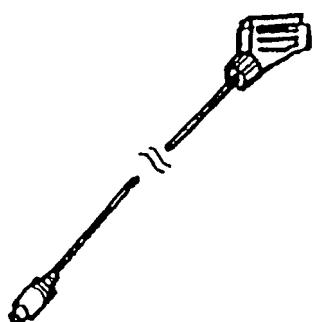
* The RGB cable for the monitor is attached.



6. Controller

Neo-Geo main-unit controller

* Two controllers are recommended for development of dual player games.



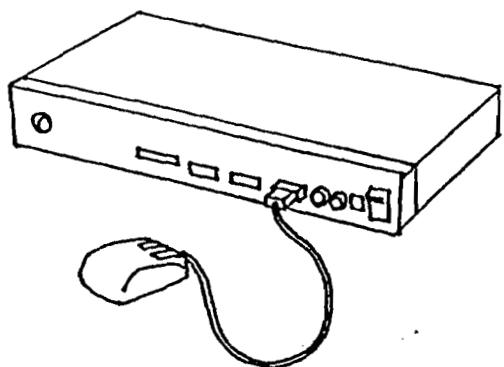
7. RGB 21-Pin Cable

RGB Cable exclusively for the Neo-Geo



8. Memory Card (for Games)

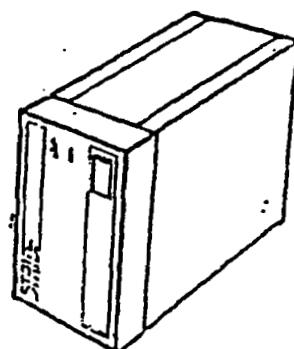
This is an IC card for game play data storage.
2 K bytes (Commercial product JEIDA Ver.
4.1 SRAM card can also be used.)



9. Art Box (Character Development Unit)

This is a design tool for character development. RS232C, RGB output, bus mouse, Centronics, and IC card output (can use JEIDA Version 3 or 4) are used as I/Os.

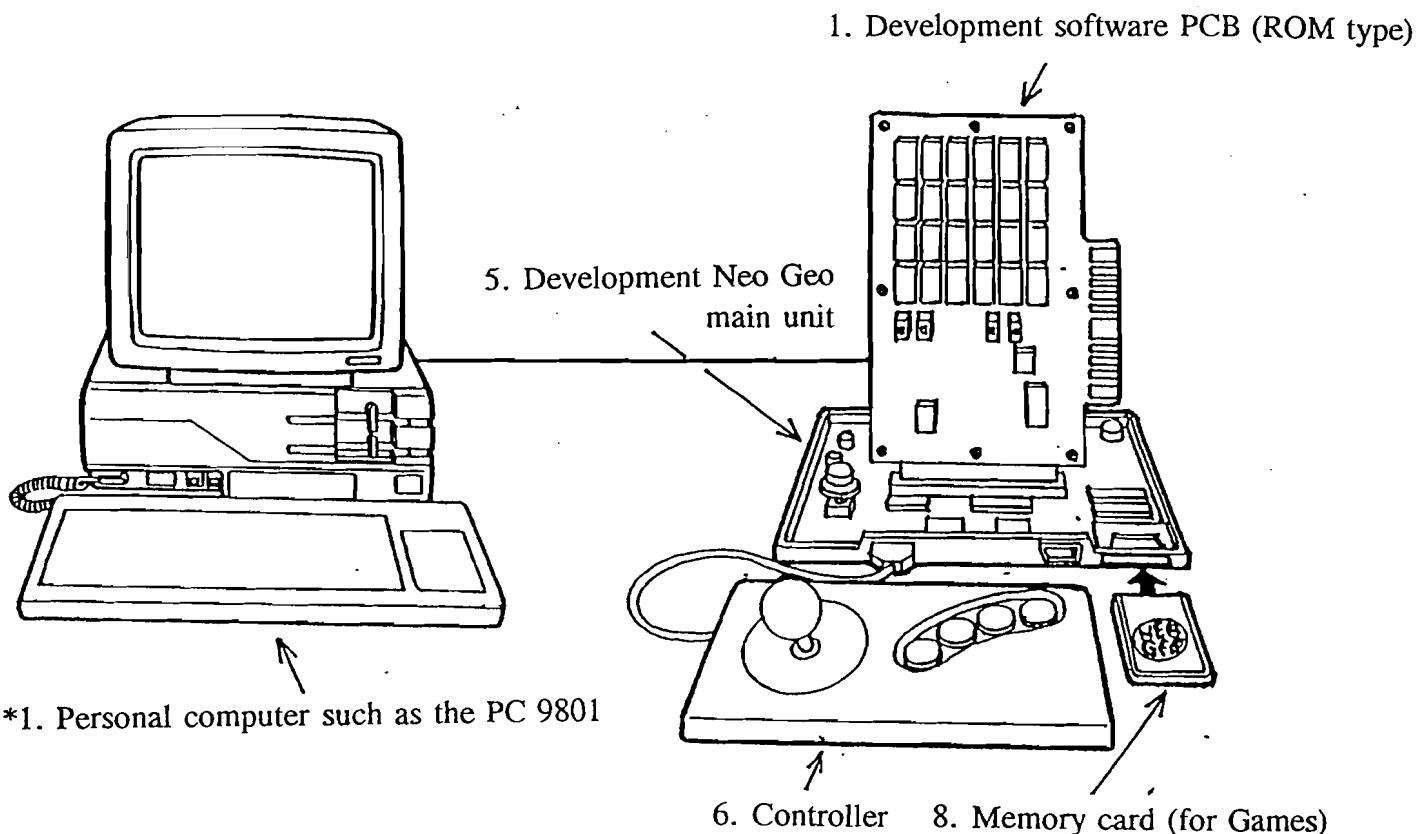
The RGB cable for the monitor is accessorized.



10. SCSI Hard Drive (for Character Development)

This is used as the external memory device of the Art Box.

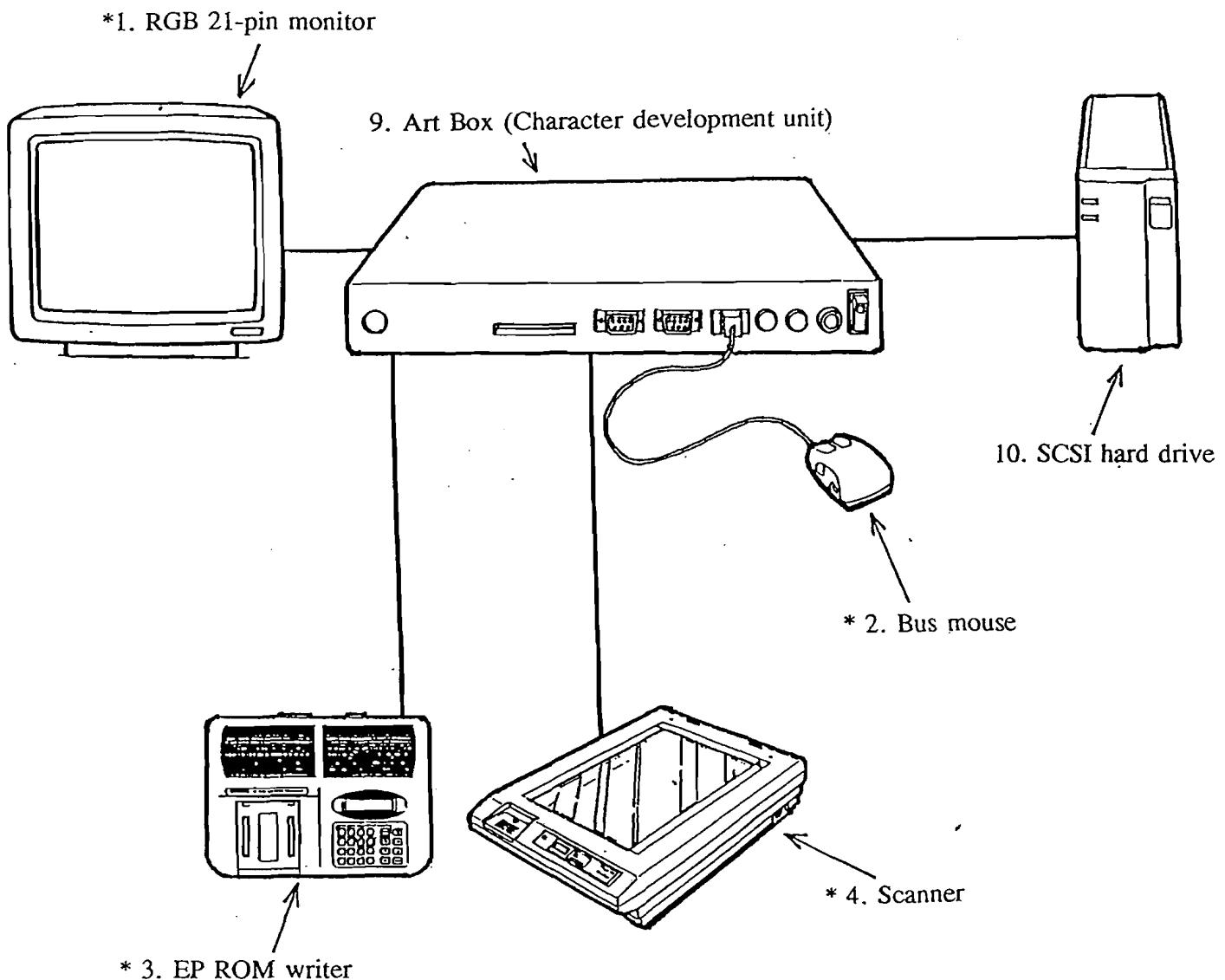
- Software Development Tool Structural Diagram



- Devices Necessary for Development

Program Development	<ul style="list-style-type: none"> * 1. Personal computer such as PC 9801, etc. * 2. 12 MHz 68000 ICE (In-Circuit Emulator) * 3. RGB 21-pin monitor * 4. PAL monitor (for PAL specification checking) <such as the Toshiba 21 System Core FS> * 5. EPROM writer (better if it can accept the 16 M EPROM) <Recommended:AF-9705, by Ando Electronics>
Sound Development	<ul style="list-style-type: none"> * 1. Personal computers, such as the PC9801, etc. * 2. Z80 ICE (In-Circuit Emulator)

- Structural Diagram of Character Development Tool



- Devices Necessary for Development

Character Development	<ul style="list-style-type: none"> * 1. RGB 21-pin monitor * 2. Bus mouse (400 dpi for the PC-9800) * 3. EPROM writer (better if it can accept the 16 M EPROM) <Recommended: AF-9705, by Ando Electronics> * 4. Scanner (Recommended: Epson GT-6000> * 5. PAL monitor (for checking PAL specifications) <Recommended: Toshiba 21 System Core FS>
-----------------------	--

* We do not market these products.

* We also do not market the IC card used for the PCB development software (IC card type).

Product Development Component Specifications

- An Example of EPROM for Development
 - ◆ Development Software PCB (ROM Type) X3002 Series

	Number of Mega Bytes	Manufacturer	Model	Reference
FIX	1 M	Toshiba	TC571001 or TC571000 (N-JEDEC) (JDEC)	Can switch between JDEC and N-JEDEC
Character	1 M	Toshiba	TC571001 or TC571000	
	4 M	Toshiba	TC574000	
Program	1 M	Toshiba	TC571001 or TC571000	Can switch between JDEC and N-JEDEC
	4 M	Toshiba	TC574000	
Music	1 M	Toshiba	TC571001 or TC571000	Can switch between JDEC and N-JEDEC
	4 M	Toshiba	TC574000	
PCM	1 M	Toshiba	TC571001 or TC571000	Can switch between JDEC and N-JEDEC
	4 M	Toshiba	TC57400	

* Not available from SNK.

Note: Differences of "JEDEC" and "N-JEDEC"

	OE	A16
JEDEC	24-pin	2-pin
N-JEDEC	2-pin	24-pin

OE = Output Enable

Refer to the cross reference on a separate sheet for details.

◆ Sample Cassette Board

	Number of Mega Bytes	Type	Manufacturer	Model	Reference
FIX	1 M	-	Toshiba	TC571001	N-JEDEC 200ns
Character	8 M	-	Toshiba	TC578200	200ns
	16 M	-	Toshiba	TC5716200	200ns
Program	4 M	4096 Type	Toshiba	TC574096	150ns
	4 M	4096 Type	Toshiba	TC574200	150ns
	8 M	-	Toshiba	TC578200	150ns
	16 M	-	Toshiba	TC5716200	150ns
Music	1 M	-	Toshiba	TC571000	JEDEC 200ns
PCM	1 M	-	Toshiba	TC574200	200ns
	8 M	-	Toshiba	TC578200	200ns
	16 M	-	Toshiba	TC5716200	200ns

* Not available from SNK.

Note: Differences of "JEDEC" and "N-JEDEC"

	OE	A16
JEDEC	24-pin	2-pin
N-JEDEC	2-pin	24-pin

OE = Output Enable

Refer to the cross reference on a separate sheet for details.

■ 256K/512K bit EPROM/OTP/mask ROM

Component	32K × 8			64K × 8		
	EPROM	OTP	MROM	EPROM	OTP	MROM
Pins	28	28	28	28	28	28
TOSHIBA	TC57256AD	TC54256AP	TC53257	TC57512AD	TC54512AP	
AMD	Am27C256	Am27C256		Am27C512	Am27C512	
Fujitsu	MBM27C256A	MBM27C256AP	MB83256P	MBM27C512	MBM27C512P	MB83512P
Hitachi	HN27C256	HN27C256	HN623257	HN27C512G	HN27512P	
Intel	27C256	P27C256		27C512	P27C512	
Mitsubishi	MSM27C256AK	MSM27C256AP		MSM27CS12AK	MSM27CS12AP	
NEC	#PD27C256	#PD27C256	#PD23C256P	#PD27C512D	#PD27C512C	
OKI	MSM27C256	MSM27256ZB	MSM53256P	MSM27512	MSM27512B	
Sharp	LH57256J	LH57256D	LH53259			LH53514
TI						

■ 1M-bit EPROM/OTP/mask ROM

Component	128K × 8			128K × 8			64K × 16		
	EPROM	OTP	MROM	EPROM	OTP	MROM	EPROM	OTP	MROM
Pins	32	32	32	32	32	28	40	40	40
TOSHIBA	TC571000	TC541000	TC531001C	TC571001	TC541001	TC531000C	TC57H1024	TC54H1024	TC531024
AMD	Am27C010	Am27C010		Am27C100	Am27C100		Am27C1024	Am27C1024	
Fujitsu	MBM27C1001	MBM27C1001P		MBM27C1003	MBM27C1003P	MB831003	MBM27C1024	MBM27C1024P	
Hitachi	HN27C101AG	HN27C101AP/APP	HN62321A	HN27C301AG	HN27C301AP/APP	HN62321	HN27C1024HG		
Intel	27C010						27C210		
Matsushita							MN231001		
Mitsubishi	MSM27C10K	MSM27C10R/PP		MSM27C10K	MSM27C10P/PP	MSM231000	MSM27C102K	MSM27C102P/PP	
NEC	#PD27C101AD	#PD27C101AC	#PD23C100E	#PD27C1000AD	#PD27C1001AC	#PD23C1000	#PD27C1024	#PD23C1024	
OKI	MSM271000	MSM271000ZB	MSM531001A			MSM531000	MSM271024	MSM271024ZB	
Sharp	LH571001J	LH571001	LH530800	LH571000J	LH571000	LH531000A			
TI	TMS27C010						TMS27C210		

■ 2M-bit mask ROM

Component	256K × 8
EPROM/OTP/MROM	MROM
Pins	32
TOSHIBA	TC532000A
AMD	
Fujitsu	MB832000A
Hitachi	HN62302B
Intel	
Matsushita	
Mitsubishi	
NEC	
OKI	MSM532001A
Sharp	LH532300
TI	

■ 4M bit EPROM/OTP/mask ROM

Component	512K × 8			512K × 8 / 256K × 16			256K × 16	
	EPROM	OTP	MROM	EPROM	OTP	MROM	EPROM	OTP
Pins	32	32	32	40	40	40	40	40
TOSHIBA	TC574000	TCS44000	TC534000	TC574200	TC544200	TCS34200	TC574096	TCS44096
AMD	Am27C040	Am27C040		Am27C400	Am27C400		Am27C4096	Am27C4096
Fujitsu	MBM27C4001		MB834000			MB834200	MBM27C4096	
Hitachi	HN27C4001G		HN62324			HN62424	HN27C4096G	
Intel	27C040			27C400			27C240	
Matsushita			MN234002			MN234000		
Mitsubishi	MSM27C401K	MSM27C401P	MSM23401			MSM23400	MSM27C402K	MSM27C402P
NEC	μPD27C4001DZ	μPD27C4001CZ	μPD23C4001	μPD27C4000DZ		μPD23C4000	μPD27C4096	
OKI	MSM27C401		MSMS34001A			MSMS34002A		
Sharp			LHS34300			LHS34500		
TI	TMS27C040							

■ 8M/16M bit EPROM/Mask ROM

Component	1M × 8	1M × 8 / 512K × 16		2M × 8 / 1M × 16	
	MROM	EPROM	MROM	EPROM	MROM
Pins	32	42	42/44	42	42/44
TOSHIBA	TC538000A	TCS78200	TC538200A	TC5716200	TCS316200A TCS316200B TCS316210B
AMD					
Fujitsu			MB838200		MB831620
Hitachi	HN62308B		HN62408		HN624016
Intel		27C800			
Matsushita	MN238002		MN238000		MN2316000
Mitsubishi	MSM23801		MSM23800		MSM23160
NEC			μPD23C8000		μPD23C16000
OKI	MSMS38001A		MSMS38002A		
Sharp	LHS38100		LHS38000		LHS316000
TI					

- An Example of the IC Card for Development
- ◆ Development Software PCB (IC Card Type) X3002 Series <P1-C2>

	Number of Mega Bits	Display on the Card Surface	IC Card Version	Reference
FIX	1 M	128	JEIDA Ver. 4	Can use both Versions 3 and 4. *
	4 M	512	JEIDA Ver. 3	
Character	32 M	4 M	JEIDA Ver. 4	Capacity switching is possible via the slide switch (SW1 and SW2).
Program	4 M	512	JEIDA Ver. 3	Cannot use the Version 4 card for the program. Access time: 150 ns *
Music	1 M	128	JEIDA Ver. 4	
	4 M	512	JEIDA Ver. 3	Can use both Versions 3 and 4. *

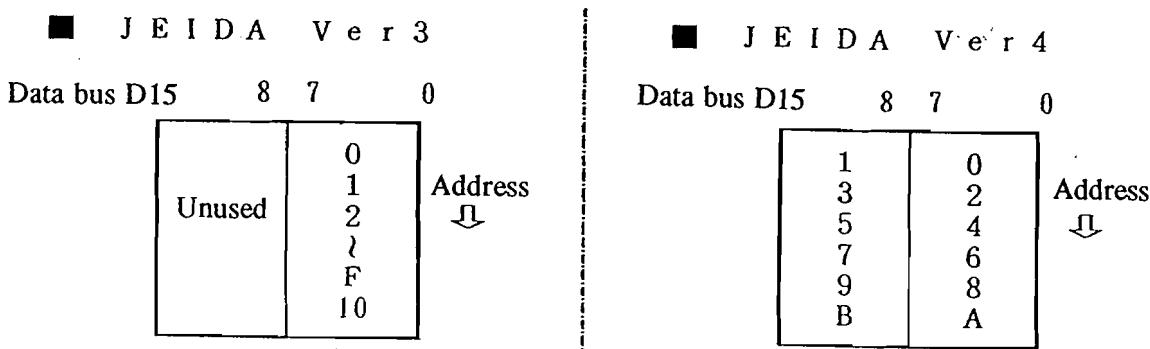
* The JEIDA Ver. 3 IC cards are not marketed to the public.
 Versions after JEIDA Ver. 4 IC cards can be used instead of version 4.

- ◆ The data region for the PCM is exclusive to "M EPROM (JEDEC)."

	Number of Mega Bytes	Manufacturer	Model	Remarks
PCM	4 M	Toshiba	TC574000	Less than 200ns

* We do not market these products.

Note: Although versions 3 and 4 can be used for "music" and "fix," the following points must be noted.



- ◆ Regarding the access time of the JEIDA Version 4 IC Card

Generally it is "200 nanoseconds."

We recommend "200 ns," although there is the "250 ns" type.

◆ Regarding the IC Card Reader for the JEIDA Version 4 IC Card

The other IC cards (not the programs) can only be read with this development board.

Please purchase the "IC Card Reader for Version 4," which is being marketed.

Recommended Products

Manufacturer: Adtech Systems

Product Name: RAM-ZO

Type: AMI-2F

→ Corresponds to flash memory.

◆ Regarding the IC Card Capacity Display

Generally, the capacity is displayed in bytes.

Example:

Card Display	When Converted to Bits	
512 K Bytes	4096 K Bits => 4M Bits	Multiplies the card display eight times
2 M Bytes	16 M Bits	,
4 M Bytes	32 M Bits	

♦ When Using as Characters

* C1 and C2 are pairs Increase by two

Example) For 64M-bit characters (Ver. 4)

Developing using four cards
 $64 \div 4 = 16$ M bits

Developing using two cards
 $64 \div 2 = 32$ M bits

Thus, 16 M bits per one card

Thus, 32 M bits per one card

$$16 \div 8 = 2\text{M bytes}$$

$$32 \div 8 = 4 \text{ M bytes}$$

Use four "2 M byte" IC cards.

Use two "4 M byte" IC cards.

- Examples of Sound Development Utility Devices

Mixer	Connects between devices and adjusts the sound quality.
Patch Bay	Used along with the mixer, and smoothes out connections between devices.
Synthesizer	Used as sound source for instrumental sounds and effects.
Sampler	Used as sound source for instrumental sounds and effects. Also, short recording, editing, and processing are performed.
Sound Processor	Processes sound, such as echoes, etc.
Sound Effect Library	50-60 set CDs Various sound effects are recorded (includes the licensing fee).
Audio System	Power amps, speakers, DAT, microphone, etc.

* Not available from SNK.

The
future is
now



Neo-Geo/MVS Software Development Planner's Manual

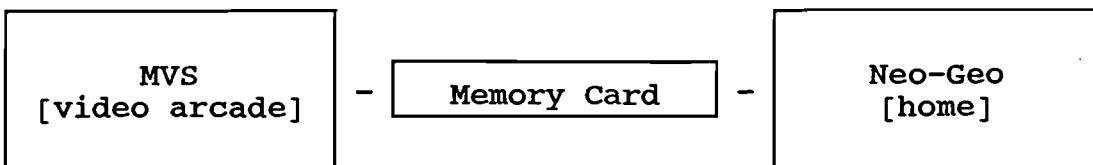
SNK

Neo-Geo/MVS Game Development Process

- The Neo-Geo and MVS game software packages at our company are developed through the following process. Please read this section as a reference.
(100 M software was used to determine the production period, and the following is an average example.)
1. Draft an original plan for the project (approximately 30 days to draft the plan).
 2. Planning meeting: Define the plan's details and determine the submission deadline for the plan.
 3. Develop the plan and test characters, then proceed to a design review (30-60 days for development).
 4. Planning meeting: Review each step in the program's production process, including characters and sounds, and submit on the deadline.
 5. Start character development (180-240 days after start of the development).
 6. Start program development (210-270 days after start of the development).
 7. Start sound development (60-120 days after start of the development).
 8. Character and sound check: Perform corrections and changes as necessary.
 9. Game-detail check and company location -> Perform corrections and changes as necessary.
 10. Start the debugging process (start about 60 days prior to the final deadline date).
 11. Location testing: Perform corrections and changes as necessary after the final check of each item.
 12. Complete the character and sound master ROM: Submit the ROMs to SNK.
 13. Complete the program master ROM. Submit the ROM to SNK.
 14. Perform the debugging process for approximately a week, even after submitting the final ROM revision.

The entire process takes approximately 10-14 months.

Issues to be considered while making the original game plan



- What is Neo-Geo?

The Neo-Geo system allows individuals to play games which are usually installed in video arcades. Individuals can rent larger capacity hardware at rental stores, or they can purchase the hardware and software. A memory card, which is compatible with the MVS, is used to store the game information, thus the player can continue the game at home after leaving the video arcade.

- What is MVS (Multi Video System)?

MVS is designed for coin operation usage. MVS contains one or more program cartridges which differ in size from the consumer cartridges. There are two types of MVS: one has multiple cassettes and the other type is equipped with a single cassette. Players using the multiple slot system can choose which game to play by pushing the Game Select Button (described later).

[Domestic distribution]

- * board with 1 slot (capable of connecting to a JAMMA unit)
- * board with 2 slots (for the special upright unit)
- * board with 4 slots (for the special table or upright unit)
- * board with 6 slots (for the special table or upright unit)

[Overseas distribution]

- * board with 1 slot
- * board with 2 slots (for the special upright unit for the upright unit exclusively used for MVS)
- * board with 4 slots (for the special upright unit)
- * board with 6 slots (for the special upright unit)

Note: "Slot" means a game cartridge receptacle.

Note: The above configurations were available as of February 1991; what is currently available may differ.

- What is a Memory Card? (Details to be described later)

A memory card stores information, such as the player's status and area, enabling the player to continue play at home (using Neo-Geo) after leaving the video arcade (using MVS). Therefore every game should be compatible with the Memory card. (Although the game should be playable without a Memory card.)

- Differentiating the Neo-Geo and MVS games

When designing a game, consider including some features to distinguish the Neo-Geo version from the MVS version (see the examples below). The purpose is to give some incentive to users to rent Neo-Geo; please design in such special features in order to raise the entertainment value of this machine to consumers.

Examples:

- Make the Neo-Geo's standard difficulty level different from the MVS version's level. Allowing the Neo-Geo difficulty level to be fully selectable raises player interest.
- Include more varieties and longer demonstration images in Neo-Geo, since the length of play does not have to be cut down so much for home. Extra ideas, which can not be included in the game for MVS because of the limit of play time, may be incorporated in the game for Neo-Geo.
- The number of bonus or continued games should be limited in Neo-Geo. (In that case, please display the bonus or continued games left.)

- Game Content, Language, and Symbol Considerations

The Neo-Geo and MVS units operate in a wide variety of home and arcade locations. The players have a wide age range, especially for the Neo Geo home unit, which is used by families. When designing the game project, the following factors must be considered during the design. A recent game trend is the introduction of ultra-violent games which are designed to use shock value to gain audience. This trend has caused regrettable actions to occur in terms of public acceptance of games and exposing a younger age range to violence beyond their years. SNK Corp. requests that the game designers use good judgement in their game design. Though there is no set guidelines, please refer to the following below as a gauge in the design stage.

1. Avoid scenes of spurting blood or ultra violence for the game actions, even in secret moves.
2. Symbols of adult consumption such as liquor bottles and cigarettes or large wads of money require special consideration.
3. Avoid religious symbols of any form.
4. Copyrighted symbols and trademarks or brand names are not recommended, unless there is a brand tie-in for the game basis. Similarly, avoid using famous people's names or likenesses due to possible invasion of privacy.
5. Profanity and anti-social language are to be avoided.
6. Avoid using sound effects or music that is borrowed without permission of the authors/composers.

As a general rule, the most important thing to remember is to give a good play value without resorting to shock tactics. Comedy in good taste is a very good method to add gaming value to any game. If the whole family can laugh together at a game with a sense of humor, it provided a more enjoyable experience without being offensive.

Brief Game Flow (MVS Only)

- The MVS unit game developed by our company flows as shown below. The screens marked with a ★ must be developed, but the ones marked with a ♦ are optional, depending on each project. (Since the following is strictly basic, please consult with our company before development or refer to our games for actual processes.)
- The software dip unit settings are standardized, as are the hardware dip. (Please refer to the sections on software dip and hardware dip.)

Power ON

.....

Eye-catching Neo-Geo display
(displayed by the system) ← If the credit is 0, loop here

Display the FBI mark (U.S. version only)
(displayed by the system)

Title demo (★) is displayed from slot 1 in order

Game demo display (★)

Insert credit

Forced start countdown display (★)
(if game start compulsion is set)

Start button ON : if there are multiple slots, the game to be played can be selected.
Forced start countdown : with the game select button time out

How to play display (♦)

Memory card load screen (only when there is a memory card)

Start play

.....

Naming (♦)

Continue display (★) — To continue —

Not to continue

.....

Memory card save screen (★)

Game over display (★)

Don't forget your memory card! (displayed by the system)

Performance of Neo-Geo/MVS
(Provided herein are specifications needed for software developers)

• Display Method

■ **Sprite** One sprite character consists of 16 dots x 16 dots, and 32 sprite characters, vertically arranged, form into one line sprite. Neo-Geo is made up of 32-character line sprites (of 16 dots x 512 dots each) with back characters, such as backgrounds, also comprised entirely of line sprites.

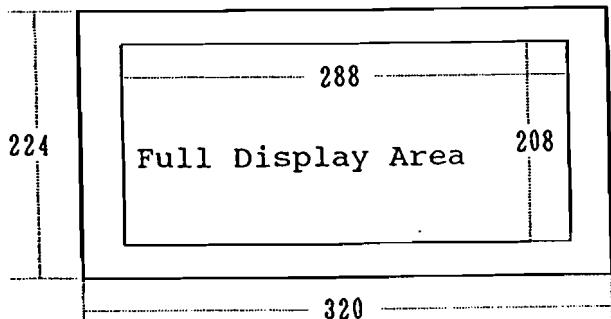
380 line sprites are all prioritized. In actual programming, these line sprites are allotted for such items as the main characters, enemies, and backgrounds.

A maximum of 380 line sprites can be displayed; however, should over 96 line sprites be horizontally displayed at one time, line sprites from the highest priority will start missing. Every program, therefore, must be worked out with this in mind. (This precaution must be taken only when more than 96 lines are displayed horizontally.)

■ **FIX** One character consists of 8 x 8 dot matrix, and scrolling (dot-by-dot movement) is not possible. This is given a higher priority than the sprite, and this cannot be changed. In program development, this is used to display the Neo-Geo system display, scores, etc.

• Screen Size (Maximum display area on the monitor)

The screen size is 320 dots (20 characters) horizontally x 224 dots (14 characters) vertically.



(Caution) Some monitors may not provide the full display of 320 x 224 (partly not visible); therefore, it is safer to set such important display items as the main character's life gauge, etc. within the 288 x 208 area. (288 x 208 is what we consider as the safe area, and it is suggested that you use your own judgment for the display positions.)

- Palettes

One palette consists of 16 colors with 0 as blank for 0-F. There are up to FF (256 units) palettes, 4,096 out of 65,536 colors can be displayed at one time.

In the 0-FF palette number range, it is possible to have 0-F for the FIX and the remaining 10-FF for the sprite. (The 0-FF palettes for the FIX are usable for the sprite as well, but the palettes for the sprite are not usable for the FIX.)

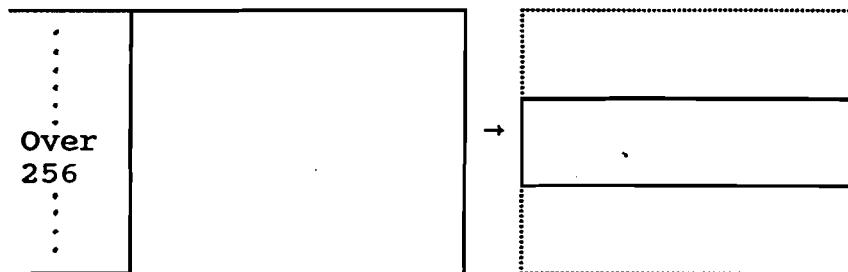
In some programs FF may even be insufficient, so the palette number can be expanded to a certain number of units. (This varies from program to program.)

- Functions

The MVS and Neo-Geo hardware have functions such as reduction, auto-action, vertical (V) and horizontal (H) flip in character units, and lowering of the color tone.

■ Reduction

The sprite can be reduced from 100% to 0%. The reductions can be done in 256 levels vertically and 16 levels horizontally. (The reason that there are only 16 levels horizontally is that a one-line sprite consists of 16 dots horizontally.)



(Note)

If an item is reduced more than 256 dots (16 characters) vertically, one dot of the character below will be pulled up when displayed (refer to the left diagram). When reducing an item that is more than 256 dots, it is necessary to take measures such as leaving the lowest dot blank. (Please perform these measures and ideas during development.)

■ Auto-Action

By using this for items that are displayed as constant action in loops, such as rivers and spectators, the CPU processing load can be reduced. This can only be used with sprites, and the number of actions can be selected from two types four and eight step and settings from 1/60 to 256/60 seconds can be set as the action speed.

(Note) The parsing method of the character for auto-action differs between the selection of four step actions and eight step actions. For four actions, the first character of each action is placed in each character bank where the lower digit is 0, 4, 8, and F, then the second character, third character, and so on, are parsed. For eight step actions, they are placed in each character bank where the lower digit is a 0 or 8, and are parsed in the order of action. (Even if the first and third actions are entirely the same, they are treated as completely different characters for auto-action characters.)

■ Character Flip Vertical and horizontal flips can be performed in character units for the sprite.

■ Decreasing the Tone of the Screen

The tone of the game screen can be decreased (a little bit).

■ Communication Function

The communication function is not included as the basic function of the hard drive, but it becomes possible by adding a communication board to the cassette side. Please take note when planning, that there are restrictions in amount of data that can be sent or received during communication.

• Sound Function

A Yamaha YM2610 is used as the sound-source chip. This chip can simultaneously produce three sounds: ADPCM (speech synthesis), four seven-sound FM sound source, and SSG (PSG compatible).

■ ADPCM-A and ADPCM-B

There are two types of ADPCMs used for speech synthesis, and the number of sounds, etc., differs.

	Number of Sounds	Sampling Rate	Data ROM	Number of Seconds That Fits in 4M ROM
ADPCM A	6 Sounds	18.5 KHz	4M Mask ROM	Approximately 1 Minute
ADPCM B	1 Sound	2.0-55.5 KHz	4M Mask ROM	Maximum of 2 Minutes If the Quality is Decreased.

The sound quality improves as the value of the sampling rate is increased for the "ADPCM-B" shown above, but this would require more memory. But if the sampling rate is low, it would require less memory, but the sound quality would decrease. The tuning can be changed by changing this value during replay. For example, if an instrumental sound, etc., is recorded, it can be used as the BGM.

The "ADPCM-B" corresponds to both BGM and sound effects. Depending on the plan, it can be used for sound effects or BGM, especially as BGM instruments.

■ FM Sound Source Four sounds can be produced simultaneously by the FM sound source.

■ SSG Sound Source This sound source is compatible with the PSG, and three low-frequency sounds can be produced, which can be mixed with noises.

■ Static Setting for Stereo

This sound-source chip is stereo, but it can only output to Lch, Rch, or L+Rch. So, when setting sounds in more detail, two of the same sounds should be used and adjustments must be made for each level setting.

■ Sound Arrangement

The sound arrangement depends on each plan. The diagram below is an arrangement for when our program is used:

BGM	ADPCM-B 3 sounds, FM 4 sounds	ADPCM-B 1 sound
Sound Effects	ADPCM-B 3 sounds, SSG 3 sounds 1 type	

■ Volume In order for it to be compatible with the MVS, please follow our specifications for volume adjustment.

Regarding the System ROM

A system ROM that controls the game system is built into the Neo-Geo/MVS. The following detail controls are handled in different regions by the system ROM.

- Types of System ROM

MVS	Japan/U.S.A./Southeast Asia/Europe
Neo-Geo	Japan/U.S.A./Europe/Asia

- MVS Regional Specification (The details of each item are described on the next page.)

Specification	Japan	U.S.A.	Southeast Asia (Korea)	Europe
Language (Display Method)	Japanese	English	English	English
Coin 1	Both P1 & P2 Coin counter 1	P1 only Coin counter 1	Both P1 & P2 Coin counter 1	P1 only Coin counter 1
Coin 2	Both P1 & P2	P2 only Coin counter 2	Both P1 & P2 Coin counter 2	P2 only Coin counter 1*
Coin 3	Unused	Unused	Unused	P1 only Coin counter 2
Coin 4	Unused	Unused	Unused	P2 only Coin counter 2
Credit/Coin Setting	Separate coin slots 1 and 2	Same for coin slots 1 and 2	Separate for coin slots 1 and 2	Separate for coin slots 1 and 2
Credit/ LED Display	Same for P1 & P2	One for P1 & one for P2	Same for P1 & P2	One for P1 & one for P2
FBI Display	None	Yes	None	None
Continue Service	None	Yes	None	None
Start Button 1	P1 start (note) 2	P1 start (note) 2	P1 start (note) 2	P1 start (note) 2
Start Button 2	P1 and P2 start	P2 start (note) 2	P1 and P2 start	P2 start (note) 2
Game Select	After coin insertion	Before coin insertion	After coin insertion	After coin insertion
Cross-Hatch Color	Red	Sky Blue	Blue	Yellow
Forced Start	Possible	None	Possible	Possible
Current Version	Version 5.0	Version 5.0	Version 5.0	Version 5.0

*: If DIP No. 2 is off for hard dip in the European version and "Coins 1 and 2 only (Coins 3 and 4 will not be used)" is set, then the box will be changed as follows

↓

P2 only
coin counter 2

- Descriptions of terms from the previous table

Regarding "coins"

- ▷ Input of coins 1-4 can be accepted, but "same for P1 & P2" means they share the same LED credit display.

Ex.) An insertion of one coin means one credit for coin 1, and the insertion of two coins means one credit for coin 2. There is only one LED credit display, and it shows a value of two when one coin is inserted in coin 1 and two coins inserted in coin 2.
 - △ "P1 only" and "P2 only" means they have independent credit LED displays. The point to note in this specification is that the credit rate for coin 1 and coin 2 will be the same. So, the number of coins for each credit will not vary, such as one coin insertion for a coin-1 credit and two coin insertions for a coin-2 credit.
 The hardware is equipped to deal with the input of coins of different values into coin 3 and coin 4. (However, dip switch 2 must be on.)
- Example) Player 1 = Coin 1 (¥100) / Coin 3 (¥50)
 Player 2 = Coin 2 (¥100) / Coin 4 (¥50)
- Note) Coin 1 and coin 2 do not have the above meanings for the U.S. version. Coin 1 indicates "coin to start the game" and coin 2 indicates "coin to continue game during the continued countdown."
 (This is described later in the section, "Continue Service.")

Regarding the "Coin Counter"

- ▷ There are two types of coin counters, and one counter is allocated for one type of coin. "Both P1 & P2" means that coin 1 uses counter 1 and coin 2 uses counter 2. "P1 only" and "P2 only" means that coin 1 or coin 2 uses counter 1 and coin 3 or coin 4 uses counter 2.

Regarding "Credit LED Display"

- ▷ "Both P1 & P2" means there is only one credit LED display, but "P1 only" or "P2 only" means that the credit LED displays are independent for players 1 and 2.

Regarding the "FBI Display"

- ▷ As one method to prevent copying, the U.S. version displays the "Drug Prevention" announcement and the FBI logo.

Regarding "Continue Service"

- A regular number of coins is required when starting a game, but for the U.S. version, when coin(s) are inserted during the continue countdown, the game can be continued with fewer coins than the regular amount.

Example) Coin 1 = One credit for two coins
Coin 2 = One credit for one coin

For the above settings:

Coin 2 is the number of coins required for one credit at the start of a game, and coin 2 is the number of coins required for one credit during the continued countdown.

Regarding the "Start Button"

- With "Both P1 & P2," start button 1 allows only player 1 to start or continue a game. Start button 2 allows players 1 and 2 to start at the same time if there are two or more credits. After the game starts, start button 2 can be used for player 2 to continue a game.

With "P1 only" and "P2 only," the start buttons are independent, while start button 1 relates only to player 1 and start button 2 relates only to player 2.

Regarding "Cross-Hatch Colors"

- The region type of the system ROM can be identified by the color displayed around the cross-hatch during the test screen.

Regarding the "Forced Start"

- After the coin has been inserted, the game will start automatically after a fixed duration, without pressing the start button. This is a standard setting and can be changed with the mode-select menu (described later).

Regarding the Hard Dip (MVS Only)

The unit's condition can be controlled with the hard dip with the MVS (along with the control using soft dip).

The hard dip is located on the MVS system PCB, the location varying, depending on the PCB model.

All settings are read and set on power up, except for switches 7 & 8 which affect the game mode immediately.

Switch Number	Item	Details
1	Setting Mode	Turn this on, and when the power is turned on the soft dip screen is displayed.
2	Coin System	Off = two-coin system status
		On = four-coin system status/ When coin 3 and coin 4 are used
3	Mah-Jong Computer Panel	Turn this on when connecting the mah-jong computer panel.
4	Communication Control	This is used as the unit identification number during communication.
5		There are four types: "4-OFF and 5-OFF," "4-ON and 5-OFF," "4-ON and 5-ON," and "4-OFF and 5-ON."
6	Communication Control	Turn this on to enable the communications mode.
7	Free Play	The game can be played without credit if this is turned on.
8	Screen Stop	The screen stops when this is turned on.

About the Mode Select Menu (MVS only)

With the MVS, each game's parameters are set using the Parameter Selection menu. This menu also sets some of the parameters used in the operation of the game machine itself, in conjunction with DIP switches.

Mode Select Menu- Operation

- When the test switch located inside the unit is pressed while the MVS system's power is on, or switch #1 is set in the hard dip, the following menu will appear on the display.
- When the menu is displayed, use the 1P joystick to select an item which needs to be set or verified, then push button A to go to that item. The display will then show the selected page.

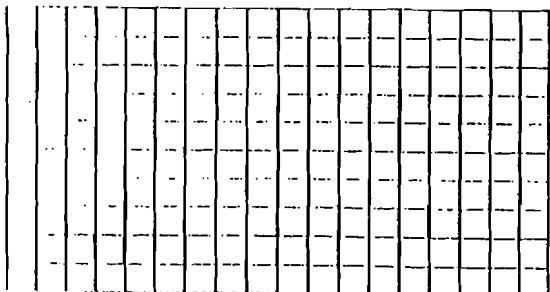
```
→ Hardware Test
Hardware (DIP) Switch Set
Parameter Select Software
Income Records
Password
Calendar Set
Exit
dd/mm/yyyy
```

* The display screens of domestic and foreign versions differ slightly.

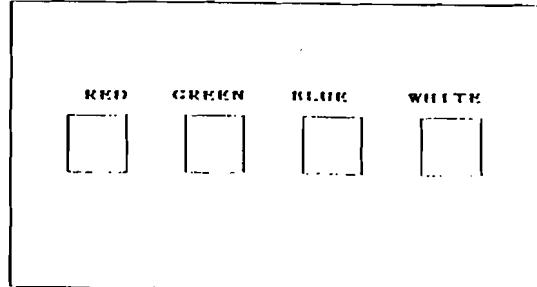
Mode Select Menu - Hardware Test

Items on the Hardware Test change when the player-1 button is pressed. An item especially useful for development is the "Memory Card Test or Backup RAM Clear" displays. Power the PCB down to exit from the Hardware Test page.

[1] Crosshatch



[2] RGB Test



[3] DIP Switch Status Display [4] Sound Test

I/O CHECK		DIP 12345678
UP	0	00000000
DOWN	0	
LEFT	0	TEST 0
RIGHT	0	COIN2 0
PUSHIA	0	COIN1 0
PUSHB	0	SERVICE0
PUSHC	0	
PUSHD	0	BL. TROT LK02
START	0	5 88 22
SLECTO	0	

SOUND TEST	
SOUND OFF	
RIGHT	
LEFT	
CENTER	

[5] Memory Card Test and Backup RAM Clear

INSERT MEMORY CARD---

BACKUP CLEAR

OK = PUSH A, B, C BUTTON

* On this menu, the Memory Card is tested to ensure that it works properly; in addition, the contents of the Backup RAM in the MVS unit can be cleared.

When the Memory Card is inserted, the screen reads :
"MEMORY CARD TEST OK"
When no Memory Card is inserted, it reads :
"INSERT MEMORY CARD"

[Memory Card Test]

Proper functioning of the Memory Card is tested. Please be advised that the data contained in the Memory Card becomes corrupted during this test if the Memory Card has any data in it.

[Backup RAM Clear]

When buttons A, B, and C are pressed simultaneously, the system initializes various parameters stored in the backup RAM of the MVS. Please be careful when performing this operation since all of the following are cleared by this procedure: unit parameters, software parameters, the calendar, and the income records.

[6] Calendar Set

Calendar Set
Current date and time mm/dd/yyyy hh:mm:ss
New date and time Use button A and joystick to select. Use button B to set. mm/dd/yyyy hh:mm:ss

Note:

The internal calendar may be altered slightly when the income statistics, described later, are collected. Please keep this in mind when the system is used for field trials.

Mode Select - DIP Switch Status

Displays the status of DIP switches on the main board.

	1	2	3	4	5	6	7	8	
Mode	0	OFF
Controller	.	.	0	Normal
Communication	.	.	.	0	0	0	.	.	OFF
Bonus Play	0	.	OFF
Stop Mode	0	OFF

0 = OFF, 1 = ON

Software Parameter Set

On this page, the system parameters and software control parameters (such as the difficulty level and bonus points) are checked and set for each game. Confirm that these values are correctly set while debugging the system as described in "Debug" section later, since these parameters directly affect the game.

When "EXIT" is selected, the system is reset and the screen is changed.

```
→ System Parameters
Slot 1      game title..
Slot 2      game title .
Slot 3      game title .
Slot 4      game title .
Slot 5      game title .
Slot 6      game title..
EXIT
```

- * After "Software Parameter Set" is selected, the display will show a list of selections as shown on the left. Please select either "Unit Parameters" or one of the games from the list.
- These items may vary depending on the number of slots and the number of games in the machine.

Display the maximum number of letters possible for the software title.

Software Parameters - Unit Parameters

The system's software parameters of the unit are set and checked.

→ COIN-1: 1 COIN = 1 CREDIT
 COIN-2: 1 COIN = 2 CREDIT
 GAME SELECTION : Only when player has credit
 Time before game start : 30 seconds
 Demo sound : Set by each game

* Use the joystick to select the item to be changed and press button A or B to set. Push button C to return to the previous page.

Item	Contents	Default
Credits for coin-1	Number of credits per coin inserted into coin slot 1	1 credit
Credits for coin-2	Number of credits per coin inserted into coin slot 2	2 credit
Game Selection Method	Whether the player can select the game when there is no credit left, or the selection can be made only when there is some credit left (multiple slot system).	Only when the player has credit
Automatic start	Select whether to start the game automatically after the time limit, or to wait until the Start button is pushed (game start compulsion, from 1-59 seconds).	30 seconds
Demo Sound ON/OFF	This sets whether there will be demo sound for the entire unit. If this parameter is set to "OFF", no sound is played even if the game's sound parameter is turned "ON".	Set by each game

Game Parameters

Each game installed in the system uses software parameters which are checked and set on this page. The contents of the parameters differ from game to game.

- * The maximum of 15 items can be set by each software; each item can have 15 options. (See the following examples for details.)
- * Please note that, of these 15 items, items numbered 1 to 4 can also be controlled in a more detailed fashion by the system ROM.

For example, the time can be set to minutes and seconds precision level. However the items that can be set by the system are limited depending on the system's configuration.

Items that can be controlled by the system (The same parameters set in items 1 to 4 may also be set using menu items 5 to 15, however, items 5 to 15 only have 15 options for each item.)

1	Play Time	0 minutes 0 seconds - 59 minutes 59 seconds
2	Continue Play Time Limit	0 minutes 0 seconds - 59 minutes 59 seconds
3	Number of Lives (e.g. planes)	1 - 99
4	Number of Continue Plays	None, No Limit, 1 - 99
	Examples of items which can be set by each software. (Contents of each item is up to the developers.)	
5	Difficulty Level	<p>Minimum 4 levels, maximum 15 levels.</p> <p>Sample difficulty levels (for 8 levels): 1 : very easy 2 : easy 3 : a little easy 4 : average 5 : a little difficult 6 : difficult 7 : very difficult 8 : extremely difficult</p>
6	Bonus Setting	<p>No bonus / limit two / no bonus limit</p> <p>Note: When "limit two" is set, player gets up to two bonuses, one bonus each time the score reaches a bonus score level. "no limit" increases the player bonuses for unlimited number of times.</p> <p>Example : When the bonus score setting is at 1000/2000 1st bonus : 1000 · 2nd bonus : 1000 + 2000 (System gives no more bonuses if "limit two" is set.) 3rd bonus : 1000 + (2x2000) 4th bonus : 1000 + (3x2000), and so on.</p>
7	Bonus Score	15 levels maximum
8	Playing Instructions	yes / no
9	Demo Sound	yes / no
10	Credit Display	yes / no
11	Free	
12	Free	
13	Free	
14	Free	
15	Free	

(Shooting game example)

NAM 2020	
→ CONTINUE PLAY	NO LIMIT
DEMO SOUND	YES
NUMBER OF PLANES	3
DIFFICULTY LEVEL	4
BONUS SETTING	LIMIT TWO
BONUS SCORES	1000/2000

(Sports game example)

HYPER BASEBALL	
→ PLAY TIME	03:00
CONTINUE PLAY TIME	04:00
CONTINUE SETTING	YES
DEMO SOUND	YES
PLAYING INSTRUCTIONS	YES
CALLED GAME	YES

Note: A maximum of 8 items can be displayed in one screen page in the above examples. When 9 or more items are used, place "NEXT" in line 8 and design the second page to be displayed when the player moves the cursor to the bottom of the page using the joystick. The first page appears again when the cursor is moved to the top of the second page.

(An example using 9 or more items)

First Page

Second Page

QUIZ BIG SEARCH

→ NUMBER OF LIVES	3
CONTINUE	NO LIMIT
DEMO SOUND	YES
PLAYING INSTRUCTIONS	YES
DIFFICULTY LEVEL	4
BONUS SETTING	LIMIT TWO
BONUS SCORES	1000/2000
NEXT	

DISPLAY CORRECT ANSWER	YES
CREDIT DISPLAY	YES

Income Records

For the MVS, various data such as the unit's total income and the income and playing time of each of the games are displayed and verified on this page. Please be careful when you execute the Backup RAM clear, since these income records are stored in the backup RAM, and they will be deleted when the RAM is cleared.

This page is mainly used by the operator, and it is also useful for the income checking during location testing.

→ UNIT/COIN
UNIT/PLAY
SLOT-1 game title
SLOT-2 game title ..	.
SLOT-3 game title ..	.
SLOT-4 game title ..	.
SLOT-5 game title ..	.
SLOT-6 game title

..... Unit's income and play data are displayed.
..... Income and play data for each game are displayed.

Income Records - UNIT/COIN

The unit's income records are displayed in the following order: "Weekly Data", "Monthly data, January through June", "Monthly data, July through December". Each category contains the following data: "Income from coin slot 1", "Income from coin slot 2", and "number of service credits."

Income Records - Unit/Play

The unit's game records are displayed in the following order: "Weekly Data", "Monthly data, January through June", and "Monthly data, July through December". Each category has data on "Number of games played", "continue games", and "Average time."

Income Records - Income, by game title

Game statistics of each game (such as "number of games played", "continue games", and "average time") are displayed.

Note: The income data might be altered when software cartridges are changed. Please execute the backup RAM clear after the change.

Password Set

* This is not directly related to the software development.

O O O O

Push button D to set.

When the memory card is used, be sure to set the password in the memory card as well.

Mode Select - Calendar Set

Year, month, day, and time is set in the system.

About Memory Card

By having a memory card to store the player's score, area, etc. in a game, the player can continue, (using the NEOGEO system), the rest of the game played at the arcade (MVS). The memory card is a selling strength of the Neo-Geo system; therefore, every game should be programmed with memory card capability.

- Specifications of Memory Card (Information needed for programs)

The memory card has "27 pages," and each page consists of 64 bytes. The data at the head of the page is used for the title display and uses 20 bytes, leaving 44 bytes on each page.

Although multiple-page use is possible for one game, please try not to use too many pages (to avoid situations where there is only one title in a memory card).

Consulting with a programmer regarding this matter is recommended in developing a program.

- Card Format

A memory card, if not yet formatted, can be formatted when it is saved or when the memory card utility menu is displayed on the monitor screen (as explained below).

- Memory Card Utility Menu (Neo-Geo unit only)

When a memory card is set in the Neo-Geo system and then the system is reset with the buttons A, B, C and D pressed down simultaneously, the memory card utility menu is displayed. In this mode, the user name can be registered in the memory card, or data copies can be performed, etc.

The operation here is to be performed according to the instructions displayed on the screen.

Memory Card Utility		
→ 1 Format card
2 Display data name		.
3 Data copy		. Select with the lever and
4 Erase Data		. choose with the button.
5 Register user's name		.
6 End

- Regulations for Saving Data

The data saving method and points are up to each program developer; however, the following fundamental display regulations must be strictly met:

■ Common Regulations

At the saving point, display "MEMORY CARD SAVE" and the option of "YES" or "NO." If the complete phrase does not fit, it is acceptable to just display "SAVE" and "YES" or "NO."

The basic position of the selection is set at "NO," selections can be made with the lever and the selection can be chosen with button A.

If the selection is "YES," "DATA SAVE OK" is displayed to indicate to the player that it has been completed. If the entire phrase cannot be displayed within the display frame, the "SAVE OK" display is acceptable.

If the selection is "NO," that is the end of the "MEMORY CARD SAVE" procedure.

As a rule only for the MVS, do not show any displays regarding the memory card unless the card is set in the unit.

Place a time limit for the foregoing selection of "YES" or "NO."
(Within 10 seconds)

- Data Loading

The data loading method and points are up to each program developer. However, the following fundamental regulations must be strictly observed:

- Common Regulations

At the loading point, display "MEMORY CARD LOAD" and the option of "YES" or "NO." If the entire phrase cannot be displayed within the display frame, just display "LOAD" and "YES" or "NO."

The basic position of the selection is set at "YES," selection is to be done using the lever and to be chosen using button A.

If the selection is "YES," "DATA LOAD OK" is displayed to indicate to the player that the procedure is completed. If the entire phrase cannot be displayed within the display frame, the "LOAD OK" display is acceptable. At this point, make an effort to display the contents of the data to be loaded.

(Example) 2nd AREA-START POWER UP LEVEL-4

If the selection is "NO," that is the end of the "MEMORY CARD LOAD" procedure.

As a regulation for the MVS, make sure not to display any items regarding the memory card unless the card is set in the unit.

Put a time limit for the foregoing selection of "YES" or "NO." (Within 10 seconds)

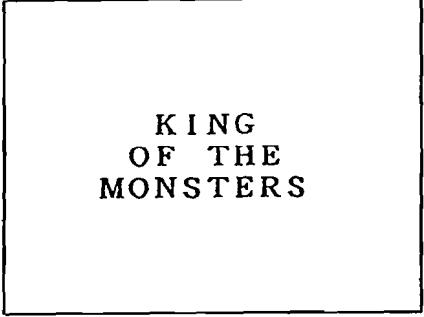
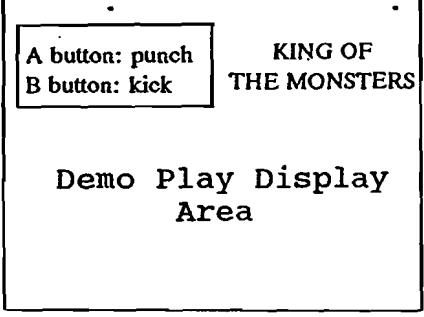
Note

Such words as "SAVE" and "LOAD" can be replaced by other words within the game context, if the meaning is easily understood, such as "retrieve progress scroll" for an RPG game, etc.).

Regarding Demo Screens

- The demo screen for the MVS (title, game demo, and ranking) should be approximately 30 seconds.

Example: King of the Monsters

	Screen	Details	Time (Aprox.)										
1 Title Screen		<ul style="list-style-type: none"> As long as it is within the time limits, title presentation (display) can be developed freely for each game. 	5										
2 Game Demo	<p>Operating instructions Title (Note) 1 (Note) 1</p>  <p>A button: punch KING OF B button: kick THE MONSTERS</p> <p>Demo Play Display Area</p>	<ul style="list-style-type: none"> Display a reduced-size title on the game demo. (The display position and title size must be determined by each developer.) Try to display the operating instructions, although it depends on the game's contents (not compulsory). However, when displaying the operating instructions, be sure to match the movements on the screen with the lever and button movements. 	20										
3 Ranking Screen	 <p>RANKING</p> <table border="1"> <tr><td>1</td><td>[Redacted]</td></tr> <tr><td>2</td><td>[Redacted]</td></tr> <tr><td>3</td><td>[Redacted]</td></tr> <tr><td>4</td><td>[Redacted]</td></tr> <tr><td>5</td><td>[Redacted]</td></tr> </table>	1	[Redacted]	2	[Redacted]	3	[Redacted]	4	[Redacted]	5	[Redacted]		5
1	[Redacted]												
2	[Redacted]												
3	[Redacted]												
4	[Redacted]												
5	[Redacted]												

(Note) 1 We recommend using fixed characters, since the "operating instructions" and "title" displayed during the demo screen may need excess sprites, depending on the game.

* Make sure to include a game demo.

* For home use, there aren't any particular provisions. (May elaborate on the creation of the title screen or story of the game.)

- Messages to be included during the demo screen (1 to 3)
- MVS (Commercial-Use)

<For Japan and Southeast Asia>

Number of credits		Message	
0		INSERT COIN This message should be displayed as long as no coin is inserted during game demonstration. (1 to 3)
1		PUSH P1 TO START Display these after returning the screen to the title display. (1)
2 ~		PUSH P1 OR P2 TO START..

<For USA and Europe>

Number of credits			Message	
1P	2P			
0	0		INSERT COIN This message should be displayed as long as no coin is inserted during game demonstration. (1 to 3)
1~	0		PUSH P1 TO START
0	1~		PUSH P2 TO START	.. Display these after returning the screen to the title display. (1)
1~	1~		PUSH P1 OR P2 TO START

- "INSERT COIN" display

[In Japan and southeast Asia]

While two players are playing simultaneously, and the game ends for only one player, display "INSERT COIN" for that player at the same time as the continue countdown display. Display credits remaining to each player separately during play. Naturally, if the game is over for both players, display the general "INSERT COIN" message along with the general countdown display of the credits.

[USA, Europe]

When two players are playing simultaneously, display "INSERT COIN" and credits remaining to each player separately, whether the game has ended for only one player or for both players.

- NEO-GEO (Home-Use)

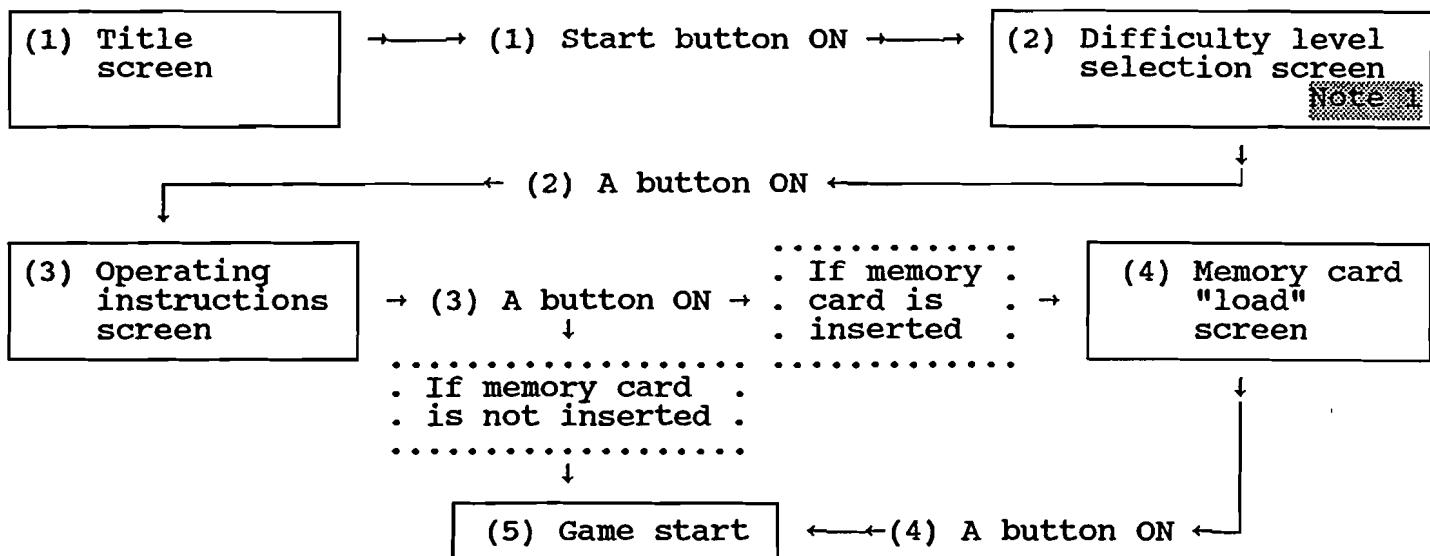
For the home-use system, the sequence of the demonstration displays (1 to 3 in previous page) do not have to be the same as the commercial machines, and no credit display is needed. Thus there are no message display restrictions, other than that minimum messages be displayed in essential locations such as the title display.

Example: "PUSH START" is displayed on the title screen, but not during the story description.

Example : Always display "PUSH START".

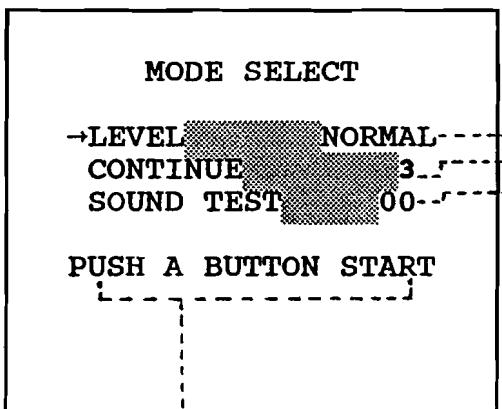
Screen Flow for Starting of a Game for Home-Use Games

- Screen Flow and Applicable Operation at Each Step (buttons to be pressed)



- Be sure to observe the operations (procedures) in moving to the next screen in steps 1 through 4 in the flow diagram above.

(Note) => A note about the unification (note) regarding the operation methods when setting items other than difficulty levels are included.
Example: The difficulty level selection screen for "Last Resort"



The initial setting is always "normal."
The initial setting is always "3."
The item can be called "Sound" or "Music" but be sure not to include sound effects or a closing musical sequence. Also, it is not necessary to display the music title by numbers as shown in the diagram on the left. (The music titles themselves can be displayed.)

Flashing

<Operating Method>

- The initial status of the cursor starts from the first item, as shown above.
- The cursor only moves up or down and is controlled by the up-and-down movement of the control lever. (This does not have to loop.)
- Each mode is set by the left and right movements of the lever. (Do not allow for selection using the buttons.)

Example) For the "level" selection, by pressing the lever to the left from the initial status (normal), it loops in the order of "easy," "MVS," "hard," and "normal." When the lever is pressed to the right, it will loop in the opposite order.

- Press the "A Button" when moving from one screen to the next. (Be sure there is no response when other buttons are pressed.)

Game mode and credits: 2-player game running on MVS (commercial-use)

a The following section applies only to sports games using the 2-player "VS (versus) MODE".

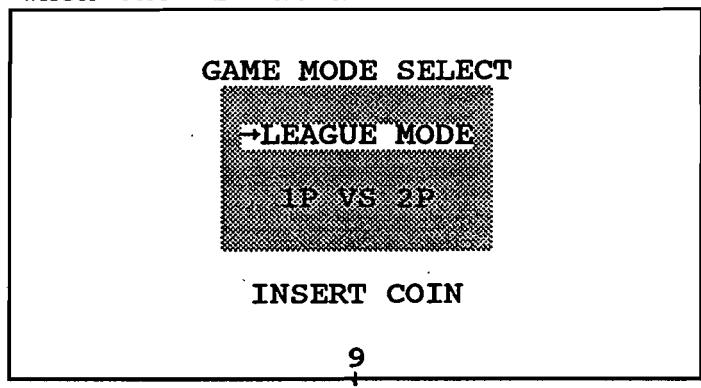
- Please refer to the following example (taken from "SOCCER BRAWL"):

[Domestic and southeast Asia version]

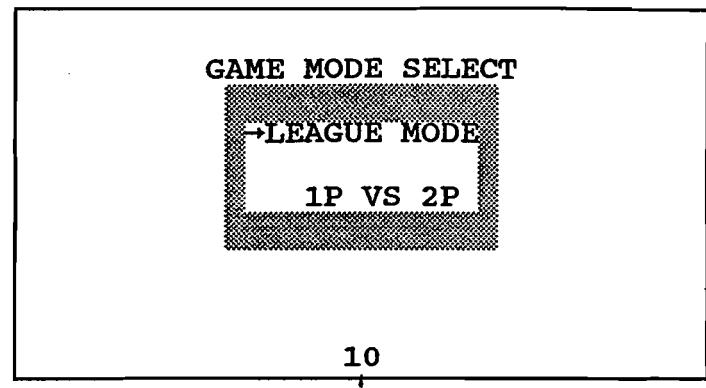
Credits	Start button		Machine's response
	P1	P2	
1	PUSH		<p>Credit decreases by one when player-1 pushes the start button. The display will show the following mode choices: "LEAGUE MODE/VS MODE". The player cannot move the cursor off the "LEAGUE MODE", and use techniques such as dimming the "VS MODE" line (by changing the palette) to indicate to the player that these cannot be selected. If, however, an additional coin is inserted while this screen is displayed, the "VS MODE" will then be displayed in normal brightness, and the player can use the cursor to select the "VS MODE". If the player selects the "VS MODE" and pushes button A, the credit tally will be decremented, and the game will start in the 2-player "VS MODE." (Note) Refer to diagram 1.</p> <p>* Modes can be selected only on the player 1 side. * When an addition coin is inserted when the mode selection screen is displayed, the timer returns to the initial value.</p>
1		PUSH	No response.
2	PUSH		<p>If player-1 pushes the start button, the credit amount will be decreased by one, then the display prompts the player to choose either "LEAGUE MODE" or "VS MODE". The credit will be decreased by one more if the "VS MODE" is selected (total of 2 credits are used).</p> <p>* Only player-1 can select which the mode.</p>
2		PUSH	<p>Credits are decreased by two and the game starts in the "VS MODE" immediately.</p> <p>* Mode select screen is not displayed.</p>

Fig. 1

<Player-1 pushes the start button when one credit is left.>



<When one more credit (coin) is added.>



Timer-----

- Indicate that the player cannot select "VS MODE" by locking the cursor and differentiating the items visually, such as dimming the phrases.
* ~~MUST always be displayed~~
- Make sure to always display "INSERT COIN".

- When one more credit (coin) is added, show the players that the "VS MODE" is now selectable, by enabling the cursor to be moved and showing visually by brightening the line.
- Remove "INSERT COIN" from the display.
- The timer is reset to the initial value.

Credit		Machine's response
1P	2P	
1	0	<p>When player 1 pushes the start button, "LEAGUE MODE/VS MODE" selection is displayed but the cursor is fixed at "LEAGUE MODE" and can not be moved by the player. At this time, "VS MODE" is dimmed.</p> <p>(The display and selection privileges (control priority) when the coins (credits) are added from the above condition)</p>
Ex.1 1~	0	<p>When one or more credits are added by player 1, "VS MODE" display brightens. The player can now select "VS MODE". The credit total decreases when the button A is pushed while "VS MODE" is selected, and the game starts in the "VS MODE". * Only player 1 can select the game mode.</p>
Ex.2 0	1~	<p>When one or more credits are added by player 2, "VS MODE" brightens and the player can then move to and select the "VS MODE". The credit amount decreases when button A is pushed while "VS MODE" is selected, and the game starts in the "VS MODE". * Either player 1 or 2 can select the mode.</p>
Ex.3 1~	1~	<p>When one or more credits are added to both players 1 and 2, the "VS MODE" display brightens and either player can select the "VS MODE". When button A is pushed and the "VS mode" is selected, a credit is deducted from the player 2 side and the game starts in the "VS MODE." * Either player 1 or 2 can select the game mode.</p>
0	1	<p>When player 1 presses the start button, one credit is deducted and the selection screen for the "LEAGUE MODE" or "VS. MODE" is displayed. But the cursor cannot be moved from the "LEAGUE MODE." At this time the "VS. MODE" is dimmed.</p> <p><The display and selection privileges (control priority) when the coins (credits) are added from the above condition> => Follows the above examples 1-3</p>
2	0	<p>When player 1 pushes the start button, the credit amount decreases by one and "LEAGUE MODE/VS MODE" is displayed. If the player selects "VS MODE", one additional credit is deducted. (total 2 credits used) * Only player 1 can select the game mode.</p>
0	2	<p>When player 2 pushes the start button, the credits decrease by one and "LEAGUE MODE/VS MODE" is displayed. If the player selects "VS MODE", one more credit is deducted. (total 2 credits used) * Only player 2 can select the game mode.</p>
1	1	<p>The credit amount of the player who pushed the start button first is decreased by one, and "LEAGUE MODE/VS MODE" selection screen is displayed. If the "VS MODE" is selected, one credit is decreased from the second player's credits. (total 2 credits used)</p> <p>* If player 1 pushes the start button and the "VS MODE" is selected, one of player 2's credits are automatically deducted.</p> <p>* Both players can select the game mode. If "LEAGUE MODE" is selected, the player who pushed the button (and paid the credit) plays.</p>

Note:

When player 1 has two credits left and player 2 has one credit left, if player 1 pushes the start button (one credit deducted from player 1) and selects the "VS MODE", then one credit is taken from player 2. (Player 1 has 1 credit left, and player 2 has none.)

[Other games (Soccer Brawl, Football Frenzy)]

- Even if player 1 and player 2 both have one credit, if player 1 pushes start button and selects "LEAGUE MODE," player 2 cannot participate in the game.
- * In this case, the start button on the player-2 side will not respond, and player 2 has to wait until player 1 finishes the game.
- Player 2 takes the left side of the field and attacks toward the right side, even when player 2 selects "LEAGUE MODE" to play against the computer. Only when player 1 plays against player 2, player 2 would have the left field.
- When the game is in the "VS MODE", the game is over regardless of who wins. Therefore there is no continue play in this mode.

Note: Certain sports games allow the players to continue the game to the end (until the game finishes). These games should treat the (1P and 2P) start buttons, which are used to continue the game, as follows: the domestic version (in which the players share credits) should check both players' start buttons. Game software for use in the USA and Europe (where the players have separate credit amounts) should check only the start button of the player who has credits left.

Regarding Credit Characters

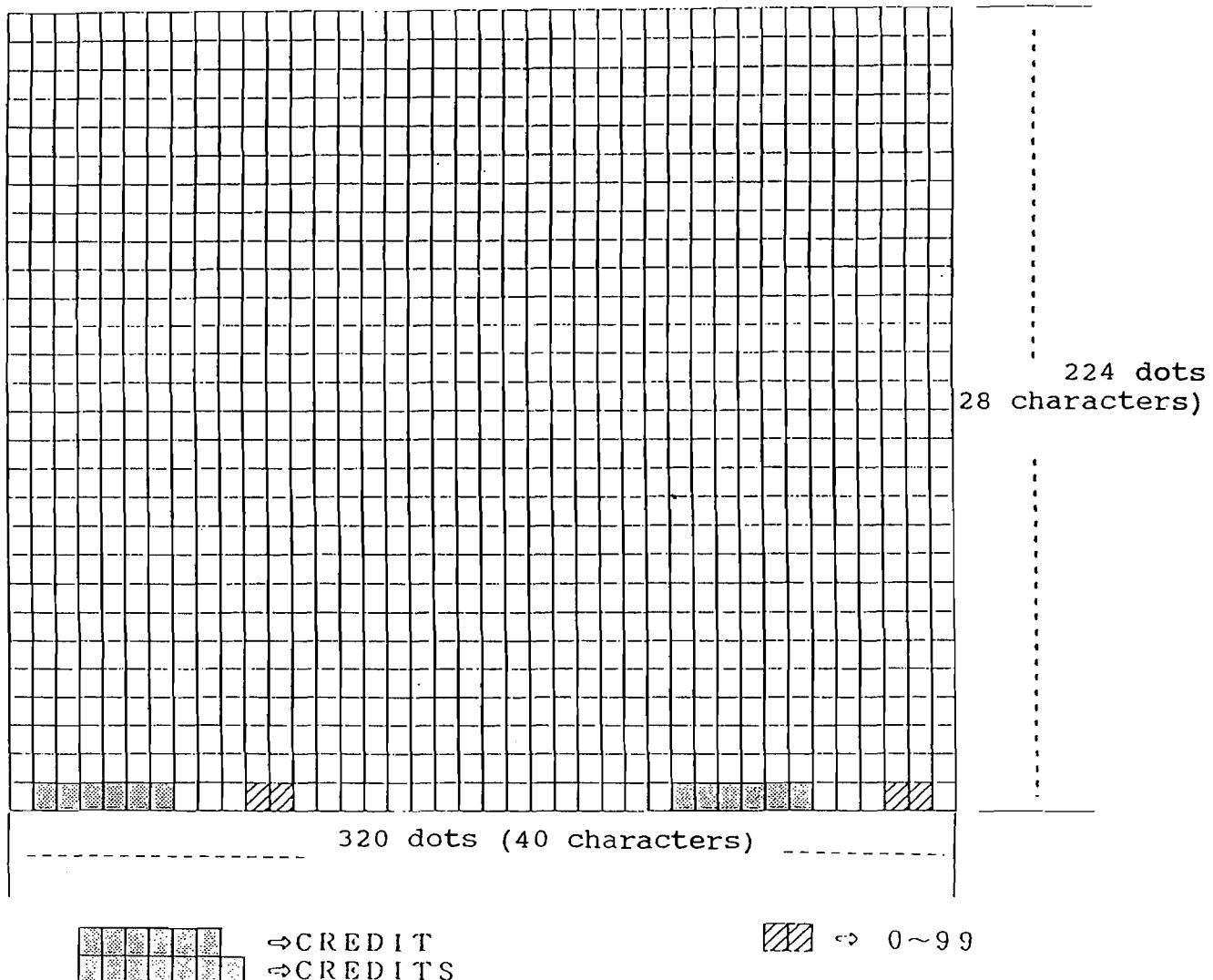
- Please lay out the credit display when using the position indicated below as a reference.

(Match the display line with the specified position.)

There aren't any restrictions for fonts, color usages, etc., if they are not the standard fixed characters. But be sure to use fonts and colors (especially the credit amount display) that are easy for the player to see throughout the game (in the scrolls during the game).

■ Confirmation of the Display Position

□ = 8x8 dot (character size is 8x8 dot)



Regarding the Continue Option

- MVS (Both Domestic and Foreign Countries)

Simply use "Continue" and "Not Continue." (This excludes the number-of-plays display.)

- Home-Use (Domestic and Foreign Countries)

1. Definitions and Number of Repetitions for the Continue Function

- The "continue" function refers to when the player's remaining life, etc., becomes "0" after starting the game, and the player can continue the game although the game is over. The game can be continued up to three times.

Player 1 => Can continue three times

Player 2 => Can continue three times

* The player can continue only three times even if it is a one-player game. Even if the game is replayed by using the memory card, the game can only be continued three times.

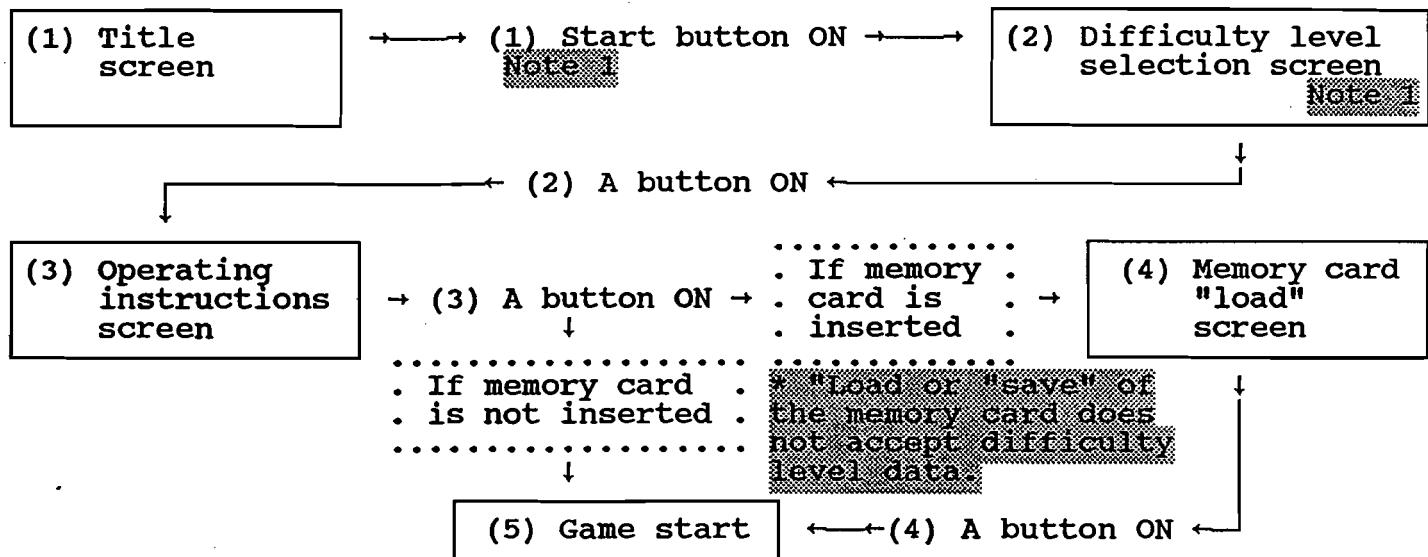
(Note) When the player presses the "start" button at the beginning of the game, the player is not continuing a game, so be sure not to count this as the player's continuation of a game.

- When the player does not choose to "continue" a game during the continue countdown though the player still can continue the game, the game will be over with a one-player game, and the display returns to the demo screen. (The number of times the player can continue returns to the initial value of "3" after the display returns to the demo screen.) Only during a two-player game may a player use the selector to continue after not choosing to continue after the continue countdown. For example, if player one did not choose to continue the game during the continue countdown although it was possible, but as long as player two is still playing the game (as long as the game is not over), player one can continue the game if player one did not continue more than three times.

2. Difficulty Level Setting During the Start of a Game

The difficulty level setting can be changed by the player before starting a game (after pressing the start button), for home-use games. (Refer to the games after the home-use "ASO II.")

► The Flow of the Screen and Operation Methods at Those Times



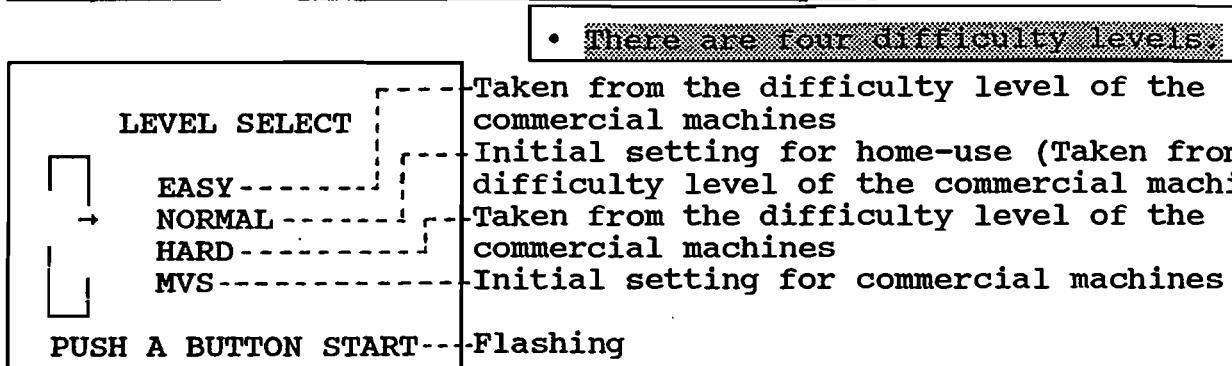
* Be sure to observe the operations (procedures) in moving to the next screen in steps 1 through 4 in the flow diagram above.

(Note) 1 => The "Start" button to enter the difficulty level selection screen, and the priorities for the control:

- Player 1 start button is turned on => Can change the settings for player 1. (Player 2 cannot change the settings.)
- Player 2 start button is turned on => Can change the settings for player 2. (Player 1 cannot change the settings.)
- Both players 1 and 2 press the start button on => Both players 1 and 2 can change the settings together.

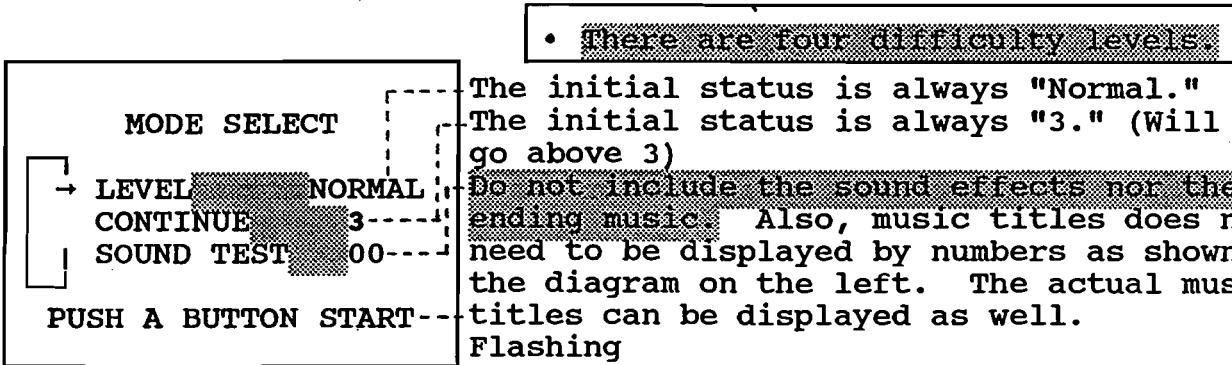
(Note) 2 => Please use the following two examples as reference for the control method for the difficulty level setting screen.

Example 1: ASO II (Only the difficulty level setting)



- * If there are less than four levels of difficulty for the commercial machines, overlap the same levels.
(Example): Have "Normal" and "MVS" at the same difficulty level.
- * If there are more than four levels for the commercial machines, "Normal" should be set at a less difficult level than with the "MVS."
- The lever moves the arrow vertically, and the A button selects the level.
- Position the arrow at "Normal" as the initial level.

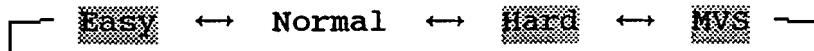
Example 2: Last Resort (When other settings besides the difficulty level are included)



- The initial status of the arrow is set at "Level," as indicated above.
- The vertical movement of the lever moves the arrow. (Only moves vertically. The menu should loop between "Level" and "Sound.")

- The each item should be selected with the horizontal movement of the lever. (Do not let the buttons set the selection.) The selections of each item should loop also.

Example: Level Selections



The selection loops between "Easy" and :MVS" by the left (\leftarrow) and right (\rightarrow) movements of the lever.

- The characters that are displayed in the game should use each game's original ASCII (8x8) characters. The words displayed on the screen should be as shown in the diagrams above. If the arrow character is not available, replace with another character.
- Please decide on the position arrangement of the characters by using the above diagrams as reference. Also, please make design decisions of the menu background for each game, since any type of backgrounds are acceptable.
- * Please decide on the items to display for the "example 2"-type menus for each development. However, ~~be sure to follow the lever and button operations as described in Examples 1 and 2.~~

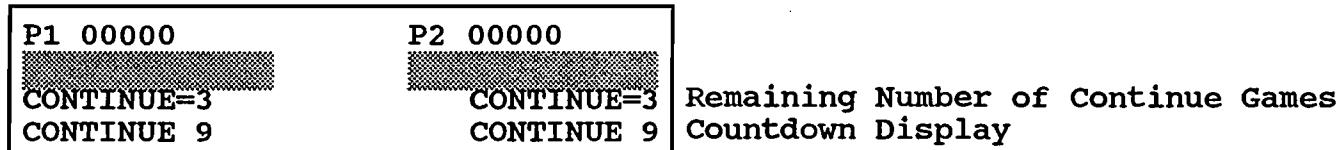
(3) Display of the Remaining Number of Continuing Plays

Display the remaining number of continue plays during the continue game screen in a way that can easily be understood by the player.

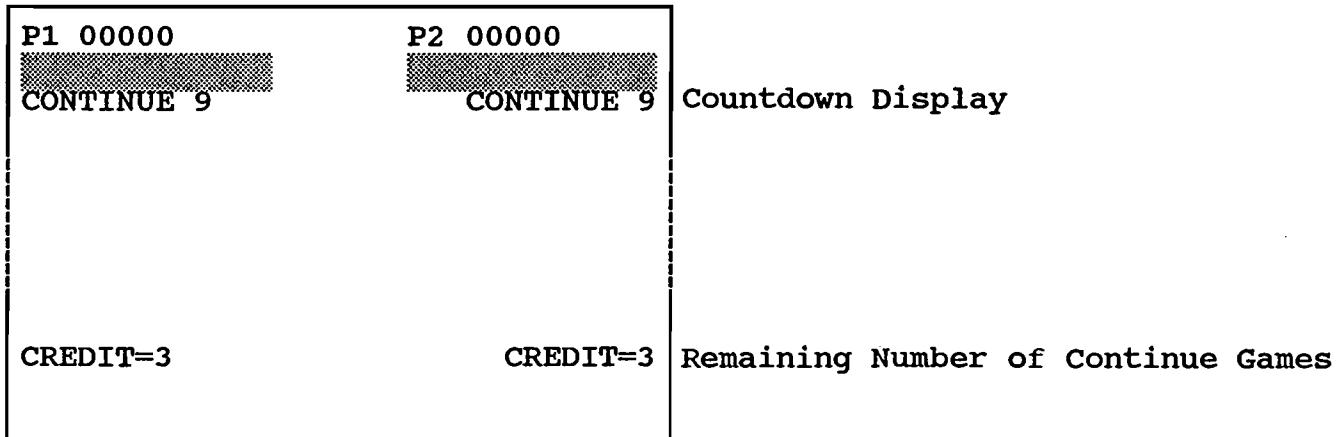
- Display the following during the countdown in the continue game screen:
CONTINUE=3 or CREDIT=3
- Please display these using each game's original ASCII (8x8) characters.
- The number is set at "3" initially, and the number decreases every time the game is continued.
- For the games where two players can play at the same time, display this continue option independently for players 1 and 2. Position this display where it is easy to see for the player (such as the position where the score or remaining lives is displayed).
- * However, the continue game display varies depending on the games, so please make design decisions for every development regarding detailed display settings.

<Example of the Screen Display Layout>

Example 1:



Example 2:



- * In example 1, the remaining number of credits and the countdown are displayed immediately after the player crashes and the screen switches to the continue screen. Then, the remaining number of credits and countdown are erased when the continue operation is exited (the start button is pressed), or the countdown has ended.
- * In example 2, the credit display for the (foreign) commercial machines is used for this purpose, so there is no need to erase the credit (remaining number of continue games) display. Regarding the display timing, display the remaining number of continue games after the start button is pressed. If only one player starts a game where two players can play at the same time, it is acceptable for the side of the player who is not participating in the game to display "CREDIT 4" as the remaining number of continue games. Also, please display "CREDIT 0" instead of erasing the credit display when the last continue game is played.
- ** Although two examples were shown above, it is recommended to use example 2 to allow the player to understand clearly how many continue games are left. However, as stated before, this is not enforced and the design decision for this aspect is up to the developers. The most appropriate method for each development (game) can be used, but the priority should be placed on the player being able to understand the display clearly.

(4) Canceling the Continue Function (Speeding Up the Function)

- By pressing any of the buttons A-D during the continue countdown, the countdown speed can be increased. (When the button is pressed continuously faster than the normal countdown speed (one second/count), the countdown speed is increased to the speed of the press of the button.)
- If both players 1 and 2 crashes (ends the game), and the game enters a "total continue state," only the player that has remaining credit (number of continue games) can cancel the countdown using his/her buttons (A-D).

This is intended to give priority to the player with remaining credit (number of continue games).

(This is to prevent the player with remaining credit from not being able to continue a game because of the player with no remaining credit (number of continue games=0) pressing the buttons by mistake.)

Credit (Number of Continue Games)		Details (for each button A-D)
1P	2P	
1~	1~	Responds to both players 1 and 2
1~	0	Only responds to player 1
0	1~	Only responds to player 2
0	0	Not necessary to allow the players to continue (including any displays). Just a Game Over display is sufficient.

* Please replicate the operation whereby the continue countdown can be speeded up continuously by pressing a button (A-D) for the commercial machines as well.

* The continue operation can only be started by pressing the start button.

(Note) Please refer to the next page regarding canceling (speeding up) the continue function for the commercial machines.

*** (2) Supplement regarding the difficulty level when starting a game

- Having four difficulty levels is a minimum requirement. However, there are basically no restrictions in including different setting items (sound mode, number of lives for each game, etc.) seen in general home-use games.
- Regarding buttons A~D when canceling the continue countdown for the commercial machines

<Domestic and southeast Asia versions>

* When the coin-shooters are the same for 1P and 2P.

Credit	Buttons A~D		The responsiveness of buttons A~D when both players have the continue option
	1P	2P	
1-	Push		Responds
		Push	Responds
0	Push		Responds
		Push	Responds

<US and European versions>

* When the coin-chutes are independent

- Please observe the buttons (A~D) on the player side where the continue option is being offered, regardless of the number of remaining credits for the US and European versions (the coin-chutes are independent).

Continue		The response of buttons A~D
1P	2P	
<input type="radio"/>		When player 1's game is over and the credit for player 1 is "0," the function can only be sped up by the buttons (A~D) on the player 1 side, even if player 2 has some credit left.
	<input type="radio"/>	When player 2's game is over and the credit for player 2 is "0," the function can only be sped up by the buttons (A~D) on the player 2 side, even if player 1 has some credit left.
<input type="radio"/>	<input type="radio"/>	<ul style="list-style-type: none"> When the game is over for both players 1 and 2, "individual speeding" can be performed using the buttons for each player 1 and 2, regardless of whether there is some credit left for player 1 and 2.

<An Example of the Continue Display>

* The games "Garo Densetsu" and "Last Resort" are used as reference examples below:

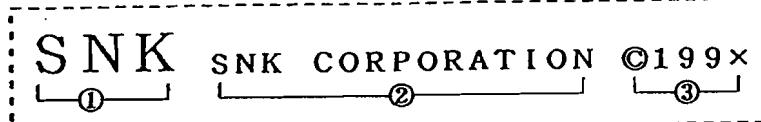
Example: Depending on the game mode selected, the second player cannot join during the middle of a game, but can continue in the middle of a game.

Example: Use "total continue state" when it is a one-player game, and switch to "individual continue state" displayed when a second player participates from the middle of a game. (As a safety measure, by displaying the "individual continue state" on the screen even when the number is being displayed with "total continue state" display, switching would not be necessary when required; just the "total continue state" would be removed from the screen.)

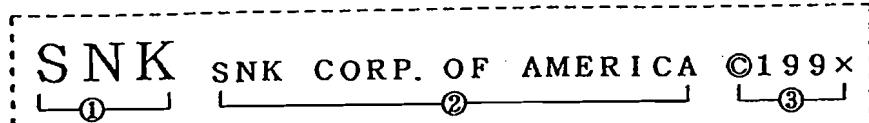
(Note) For games that can be continued until the game is over, such as certain sports games (baseball), observe both 1P and 2P buttons for the domestic and Southeast Asia versions to continue the game (1P and 2P). Observe only the side with some credit left for the US and European versions.

Regarding Company Logos

- Company Logos for the MVS (for business-use)
<Domestic>



<Foreign Countries>



- Company Logos for the Neo-Geo (Home-Use)
<Domestic>



<U.S.A.>



<Other Foreign Versions>



- *
 1. Company Logo (Standard FIX)
 2. Company Name (Original Characters for Each Game)
 3. Copyright and Year

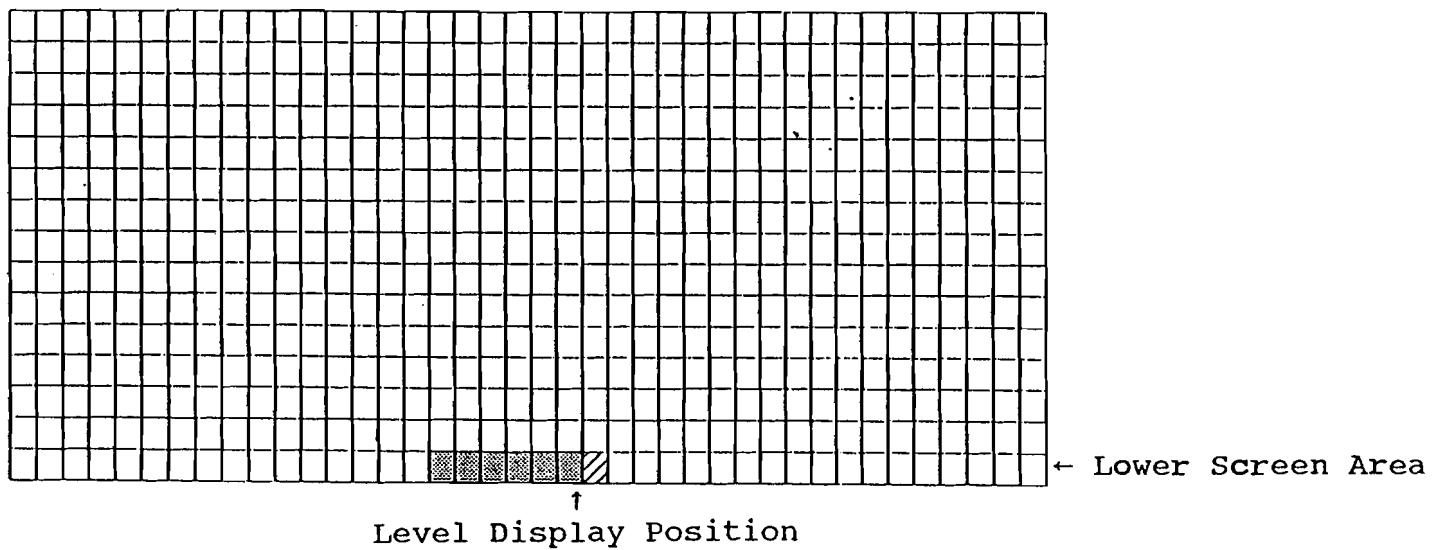
Regarding Debug Dips

- If the debug dips used during the debugging of a game are submitted for "sample location or as "master ROM," be sure none of the debug modes are activated. (If the debug modes are left, when the game is inserted with other games in the MVS unit, the game may halt after one round of the demo.)

Items to be Standardized for Level Displays

- The level (difficulty level) to be displayed on the screen is standardized as shown below.
1. Adapting Range
 - The display is limited to the MVS (domestic and foreign are the same).
 2. Display Position
 - The display position should basically be in the bottom center. However, if this space is necessary for displaying another item, please adapt a new setting, using appropriate judgement for each case.
 3. Character Size and the FIX for Display
 - The character size should be an 8x8 dot matrix. Use the standard FIX, or FIX exclusively used for the game. Please take precautions about visibility and character size when using the FIX exclusively created for the game.
 4. Regarding the Word Usage (Spelling)
 - It is standardized to "LEVEL-□". * □ will be filled with the applicable level.

* Layout Reference Diagram



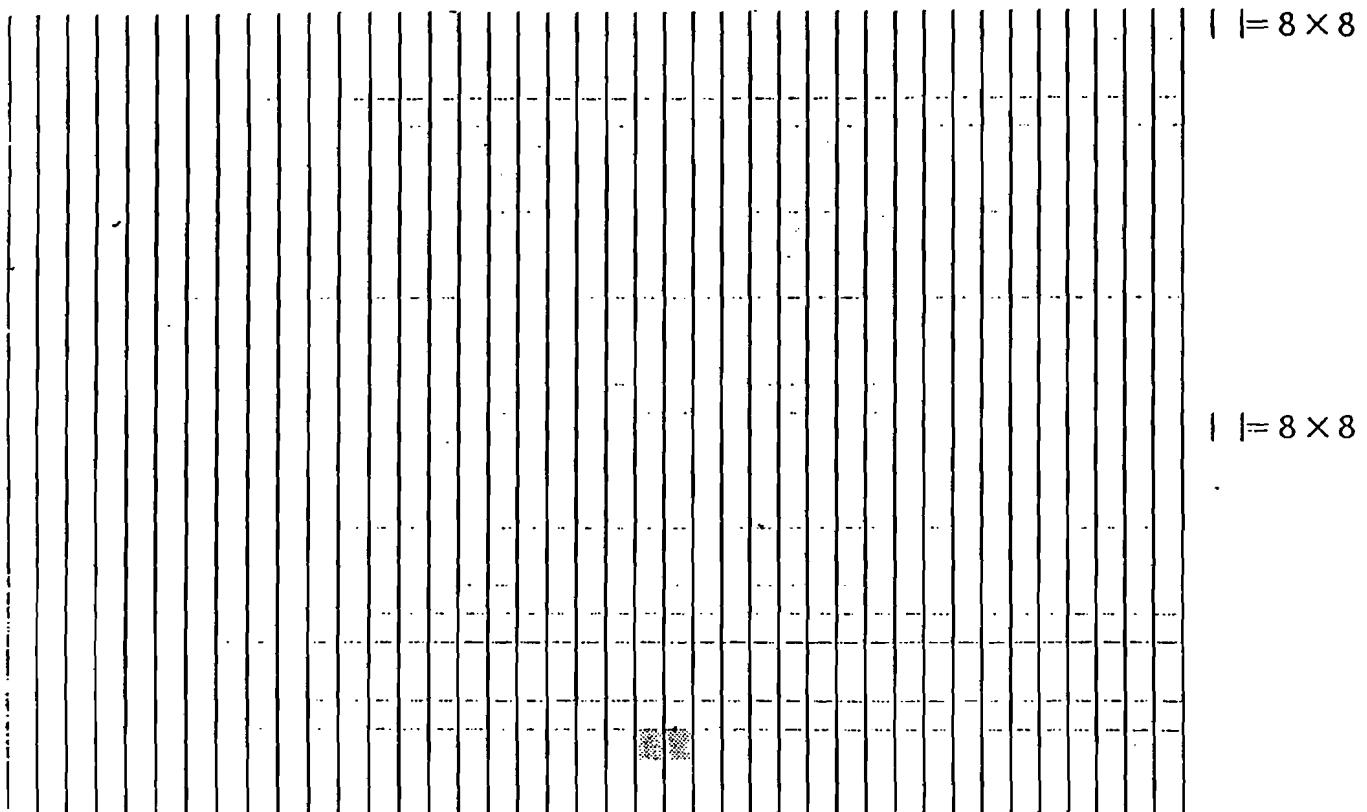
⇒ LEVEL-□

Regarding Other Regulations

- Regulations for the Forced-Start Countdown (only for MVS)

When the MVS unit has "forced start enable," the forced-start countdown starts when a credit becomes available. When the countdown reaches zero, the game starts automatically. (When another credit is added, the countdown is reset, and the countdown starts again.) Please use the FIX characters specified by our company and follow the standard position specified by our company for the display position of the countdown display.

Countdown Display Position



- Display for the Playing Instructions

Mainly for the multiple-slot MVSSs, sufficient playing instructions cannot be described using an instruction card, as usually done for other commercial machines. Therefore, there is a method to explain the operations by displaying the playing instructions on the screen during a game, instead of having an instruction card.

There are no restrictions on characters, etc. for the screen display but limit the display time between 4-10 seconds, so that the play time and demo time will not take too long. Also, have the option available to be able to interrupt this display using the buttons.

If the playing operations are simple and the game can be understood without any instructions, it is not necessary to include the playing instructions. However, we may request that the playing instructions be created after our company's evaluation of the same project.

- Regarding the Starting Method (also refer to "System ROM")

For the Japanese and Southeast Asian MVS units, the 1P button is used to start player 1, and the 2P button is used to start both players 1 and 2. For a 2-player simultaneous play, the game is started using this method, and during the game the 1P button is used to start player 1 and the 2P button is used to start player 2.

For the US and European MVS units, 1P button (left side) is used to start player 1 (left side) and 2P button (right side) is used to start player 2 (right side).

For the Neo-Geo, these design decisions are up to each development.

Regarding Other Regulations

- Naming Regulations

Please follow the following regulations when the naming option is available for high scores and records with the Neo-Geo and MVS games. However, if the game is completely unique, please make design decisions for each development.

- The timing for the naming should be after the game is over (when the remaining lives reaches 0, when time runs out, etc.), and before the continue option is displayed.
- Please allow only three letters for the name entry. This is also to prevent any profane words or any anti-social words to be entered for the machines used overseas.
- The basic operation method when entering the initials is to select the letters using the vertical movement of the lever (or horizontal), and choose using the A button (move to the next letter to be set, or after the third letter, end the naming). In order to return to the previous letter that had been set, include such backspace symbols as "**←**" in the letter set or have the system be able to return to the previously set letter using the B button. Also, include the ending symbol such as "END" in the character set for the player to be able to exit without having to enter all three letters.

- Characters that must always be included

Always include the characters specified by our company in the ROM to display the characters for the system-related displays in the Neo-Geo and MVS.

- Sprites: Neo-Geo eye-catcher logo (use all 58 characters)
- FIX: Standard characters for system control (can use FIX from bank 00 to 02)
- FIX: Characters for the number of credit display (can use all 21 characters)

- Note: Regarding the Manufactured Year Display

The manufactured year display that must be displayed in the title screen, etc. should be the year when the product began being marketed. So please take note for products that were completed over the course of more than one year.

Regarding Other Regulations

• Creating the Soft Dip Chart

We include the soft dip manual for the product for MVS cassette marketing. Please create a soft dip manual for each game when the details of the game soft dip have been determined.

(Creation Example-1) "Quiz Big Search" Made by Our Company
(The actual size is A4.)

Quiz Big Search Soft Dip Details

Quiz Big Search

- ▶ Hero
- Continue
- Demo Sound
- Playing Instructions
- Difficulty Level
- Bonus Rate
- Bonus
- Correct Answer Display
- Credit

1/ Hero	This sets the number of lives the character will have.
Operating Method	Place the arrow to "Hero" using the vertical motion of the lever, and proceed with the setting using the "A" button. The "B" button restores the screen and establishes the setting.
2/ Continue	This sets the number of times the game can be continued.
Operating Method	Place the arrow to "Continue" using the vertical motion of the lever, and proceed with the setting using the "A" button. The "B" button restores the screen and establishes the setting.
3/ Demo Sound	This determines whether there will be sound during the demo game.
Operating Method	Place the arrow to "Demo Sound" using the vertical motion of the lever, then proceed with the setting using the "A" button. The "B" button restores the screen and establishes the setting.

4/ Playing Instructions	This determines whether there will be an operating-instruction screen at the beginning of a game.
Operating Method	Move the arrow to "Playing Instructions" using the vertical motion of the lever, then proceed with the setting using the "A" button. The "B" button restores the screen and establishes the setting.
5/ Difficulty Level	This sets the difficulty level of the game. (There are 15 levels from 1-15, and difficulty increases as the number increases.)
Operating Method	Move the arrow to "Difficulty Level" using the vertical motion of the lever, and proceed with the setting using the "A" button. The "B" button restores the screen and establishes the setting.
6. Bonus Rate	<p>This sets the system where the remaining lives in the game will increase according to the players' scores.</p> <ul style="list-style-type: none"> • Second Bonus The number of lives will increase by one, according to the score, up to two times. • Every Bonus The number of lives will increase by one, according to the scores for an unlimited number of times. • No Bonus No lives will be added.
Operating Method	Place the arrow to "Bonus Rate" using the vertical motion of the lever, then proceed with the setting using the "A" button. The "B" button restores the screen and establishes the setting.
7/ Bonus	This sets the score where the character's life increases by one.
Operating Method	Move the arrow to "Bonus" using the vertical motion of the lever, then proceed with the setting using the "A" button. The "B" button restores the screen and establishes the setting.
8/ Correct Answer Display	This determines whether the correct answer is displayed when the question is answered incorrectly.
Operating Method	Move the arrow to "Correct Answer Display" using the vertical motion of the lever, then proceed with the setting using the "A" button. The "B" button restores the screen and establishes the setting.

9/ Credit	This determines whether there will be a credit display for the MVS.
Operating Method	Move the arrow to "Credit" using the vertical motion of the lever, then proceed with the setting using the "A" button. The "B" button restores the screen and establishes the setting.

Regarding PAL (European) Specifications

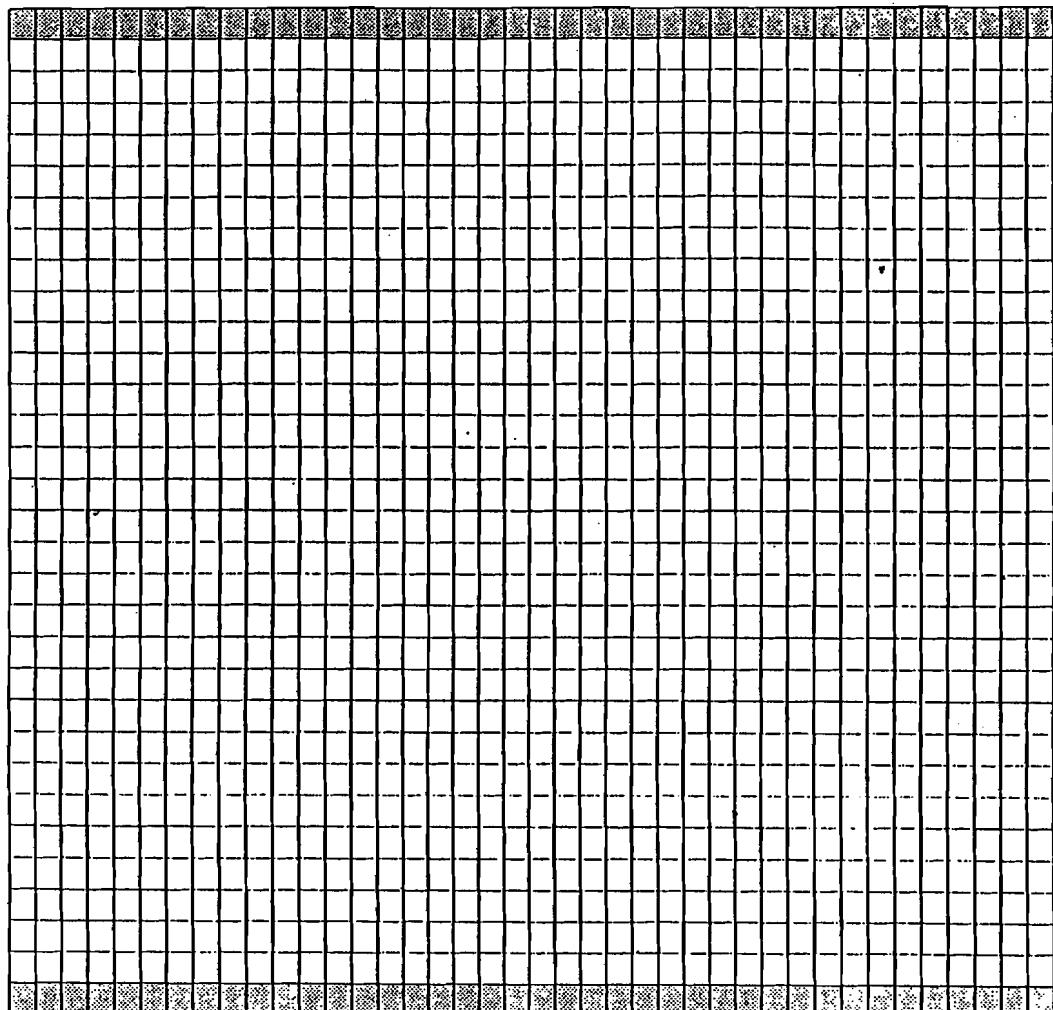
- The following areas are different for PAL (European version) specification from other versions.

1. Only 224 dots are displayed vertically on the screen for the Japanese, US, and Asian versions, but with the European version of PAL, 256 dots are displayed vertically on the screen. Therefore, since the areas that are not displayed for the Japanese, US, and Asian versions are displayed (16 dots each for top and bottom). Items with the undisplayed regions will now be displayed. This may result in an unappealing screen, so please strictly follow these regulations:

- (1) Display the 16 dots at the top and bottom that could not be seen with other versions accurately (able to see) even when changed to the European version. (Create the Japanese, U.S., and Asian versions with awareness of the 256 dot vertical display.)
 - (2) In a game with the 256-dot vertical setting, more than 256 dots will be displayed if the screen is to be swayed vertically to express a shaking motion, etc., so add FIX characters (black coloring) to hide the top and bottom of the screen. However, the size should be just one character (8x8 dot matrix (Note) refer to the diagram below). (Have the vertical length of the shaking within 8 dots.)
2. Only for the PAL specification (European version), the vertical synchronizing is done from 1/60 to 1/50 of a second. (The CPU speed of the program becomes 5/6. The sound does not change.) If the sound and music are supposed to match the screen display (such as the title demo and ranking), there may be some lagging between the two, so adjust these errors with the sound engineers and programmer. (This is not applicable during a game.)

$\square = 8 \times 8$ dot  = Parts to be hidden with FIX characters

- * Japan, U.S., and Asian versions \rightarrow 224 dots (vertical) \times 320 dots (horizontal)
- * PAL version \rightarrow 256 dots (vertical) \times 320 dots (horizontal)



Display area in PAL specifications
256 dots (32 characters) —
..... 224 dots (28 characters)

Regarding Debugging

- What is Debugging?

Debugging is performed when the game is in the condition where it can be played. The existence of bugs must be checked for different conditions including: a bug that appears during game play, bugs that appear between the system program and game program, and sound-related bugs.

When a fatal bug that seriously affects game progress is discovered after the mask ROM has been mass produced or the product has already been placed on the market, the production line may have to be halted or the product may have to be recalled, either of which can lead to a massive loss. Therefore, please be very careful and explicit about debugging.

- Common Debugging Examples

The debugging technique differs depending on the game systems. The following examples are very general, so be more precise when actually debugging, in accordance to judgement of every development.

- MVS: Can the selections be made using the select button, even if multiple software are set in the slot?
- MVS: Has the instruction card corresponding to the game set in the slot been written up?
- MVS: Does the number of credits on the unit display match the number of credits displayed in the game?
- MVS: Does the game reflect every item setting in the Mode Parameter Setting?
- MVS: Are the "unit settings" and "individual software settings" in the Mode Parameter Setting being reflected?
- MVS: Has the "income record" of the Mode Parameter Setting been correctly added/calculated?
- MVS: Does the machine eat up coins?
- Neo-Geo: Does the game correspond to the number of continue limit set?
- Neo-Geo: Does the game reflect the differentiations made in the game contents with the MVS game?
- Do display, coin system, and various settings correspond to various system ROMs?
- Can memory card saving/loading be performed according to the specifications?
- Can the memory card be used between the MVS and Neo-Geo?
- Perform irregular operations, such as changing the memory card with another card after the "Memory Card Save?" prompt is displayed.

- Are there any errors in spelling or display settings?
- Do the lever and button response reflect the specifications?
- Are the 'hits' being correctly registered as they are set for all types of characters?
- Are the shapes and colors of the characters being displayed correctly?
- Does the game "fall" into a stage where the main character can no longer be controlled, or the enemy can no longer move?
- Are the scores being added and calculated correctly?
- Are there any infinite loops?
- Are the priorities for the sprite being displayed as specified?
- When enemies are left on the screen intentionally, are there any bugs from such factors as slower CPU processing speed?
- Similarly, do items disappear that aren't supposed to disappear to proceed with the game, by 96 sprites lining up on the same horizontal line?
- Can the sound (BGM and sound effects) be heard as they were specified?
- Does the continue system function appropriately?

[1] Start-Up Checks

Unit/Classification by Country	Domestic	U.S.A.	Europe	Southeast Asia
Six Slots				
Four Slots				
Two Slots				
One Slot				
Are there any irregularities when inserting a game in any of the slots?				
Does the demo game loop correctly even when multiple software packages are inserted in the slots?				
Can selections be made using the selection buttons with the same settings as above?				
Are instruction cards being written for the slots that are set?				
Are there any irregularities when software from another company is inserted?				

* The checking is done after the backup has been cleared.

* The write-ups with the instruction card are only done with the six-slot units, old SCB units, and American units.

[2] Soft Dip Inspection (Unit Settings)

Item	Contents	Demo	Game	Remarks
Coin 1	1 Coin= Credit			
	2 Coins= Credit			
	Coin= Credit			
	Coin= Credit			
Coin 2	Coin= Credit			
	Coin= Credit			
	Coin= Credit			
	Coin= Credit			
Game Select	Only when there is credit			
	Free			
Forced Start	1-60 seconds			
	None			
Demo Sound	Set by each game			
	None			

Item	Contents	Demo	Game	Remarks
Continue Limit	Limit	****		
	None	****		
Demo Sound	Sound			
	None			
Playing Instructions	Instructions	****		
	None	****		
Credit Display	Display			
	None			
Difficulty Level	1	****		
	2	****		
	3	****		
	4	****		
	5	****		
	6	****		
	7	****		
	8	****		
Bonus	None	****		
	Second Bonus	****		
	Every Bonus	****		

Item	Contents	Check	Remarks
Unit	Coin 1		
	Coin 2		
	Service		
Software	Number of plays		
	Number of times to continue		
	Average play time		

[5] Inspection in the Coin-Insertion Regions

Item	Coin1	Coin2	Counter1	Counter2
After the power is turned on, continuously insert coins before the demo starts and confirm the effects.				
Also, confirm the service switches in the situation above.				
Continuously insert coins when the demo starts.				
Continuously insert coins when the demo screen is changing.				
Continuously insert coins when the demo screen is complete.				
Continuously insert coins at the start of a game.				
Continuously insert coins when the game screen is changing.				
Continuously insert coins when one pattern has been completed.				
Continuously insert coins when the game is over.				
Continuously insert coins when entering the name.				
Continuously insert coins when the lever and buttons are pressed.				
Insert coins at the same time the start switch is turned on.				
Press the one-player button and two-player button at the same time when the credit display shows "1," and confirm that the credit display will not show "99."				
Start player-one or player-two when the credit display shows "10," and confirm that the credit display changes to "9."				
Confirm that the counter is linked with the number of coins inserted, and not the credit.				
Confirm that the credit display on the unit matches the credit display on the screen.				
Insert more than 99 credits.				

[6] Memory Card Inspection

Item	Check
Can the memory card load and save according to the setting?	
Can the memory card be used between the MVS and Neo-Geo?	
Insert a memory card that is full.	
Insert an unformatted memory card.	
Insert a defective memory card.	

[7] Game Demonstration Inspection

Item	Check
Confirm the company logo.	
Are there any spelling errors or display errors?	
Are there any irregularities in the characters or background screens?	
Are there any irregularities in the shapes or forms of characters' items or energy gauges?	
Will the credit increase without having any coins inserted?	
Are there any irregularities when the start switch, lever, and buttons are pressed without the coin being inserted?	
Are there any irregularities when the start switch is not pressed after inserting a several coins?	
Are there any irregularities when coins are inserted to start a game during screen changes?	
Are there any irregularities when the start button is pressed during a forced start?	
Confirm the demo after the game is over during standard play.	
Confirm the same situation as above during continued play.	
Confirm the demo after entering the name for the high-score record.	
Confirm what happens after the game is over if the demo sound is turned off.	
Confirm what happens after the game is over if the demo sound is on.	

[8] Game Operation Inspection (* Perform according to each software development.)

Item	Check
Are there any mistakes in the display settings for any of the displays?	
Are the lever and buttons as specified in the setting method?	
Are the colors and shapes of the characters displayed properly?	
Is the score being calculated correctly?	
Does the sound (BGM and sound effects) correspond to the setting?	
Is the continue-service function working properly?	
Does the random pattern occur?	

Continue pressing the lever and buttons for player 2 when player 1 is playing.	
Keep on pressing the lever and buttons for player 1 when player 2 is playing.	
Let players 1 and 2 start, and do not touch anything.	
Confirm any malfunctions when different items are used continuously when player 1 is playing.	
Also confirm the same as above item for player 2.	
Apply attacks on every enemy character using every item for player 1 and player 2.	
Try entering from different angles in the areas on the screen where entry is not allowed for players 1 and 2.	
Escape as much as possible without regarding the enemies for players 1 and 2.	
Minimize the attacks on the enemies and go forward as fast as possible for players 1 and 2.	
Kill the enemies as fast as possible or in a way that the most points will be obtained for players 1 and 2.	
Alternate players 1 and 2 and proceed during continued play.	
When player 1 is playing, have player 2 participate from the middle of the game, matching the timing with when player 1 is making a move or when the enemy is attacking, etc.	
When player 2 is playing, have player 1 join in the same manner as described above.	
With player 1 only, have the player obtain a high score at the final screen of each area. Then get killed there, and proceed to the next scene after entering the name for a high score.	
Confirm the same setting as above for player 2.	
During a two-player game, keep on pressing the levers and buttons after completing each area, before the next area begins.	
During a two-player game, proceed by attacking each other or being intertwined together.	
During a two-player game, proceed by making the same motions simultaneously for both players.	
During a two-player game, proceed by making the same motions as the enemies, simultaneously.	
Proceed through the final screen with just player 1.	
Clear the final screen with just player 2.	
Clear up to the final screen by showing all possible characters, filling up the screen.	

Note: Be sure to perform debugging with the MVS with the software debug dip removed. (When inserted with certain software, it may be reset during the title loop.)

Be sure the debugging mode does not start up in any case, for both the samples and master ROM.

Inspection of System Development Simulation

- * Bug-checking using the simulation of Neo-Geo System Development ROM (Ver. 6.1).

- (1) The simulation setting screen (1) is called up when the Select + C button is pressed on the player-2 side.

(1)

MODE SET
MAIN SOFT DIP
GAME SOFT DIP
RESTART
DEFAULT START
GAME START TEST
CARD FULL
CHECK SUM
BOOKKEEPING (CARD)
GAME DEBUG DIP

MODE SET	Sets the version setting
MAIN SOFT DIP	MVS unit setting
GAME SOFT DIP	Soft dip for each game
RESTART	Reset
DEFAULT START	Clear the backup area
GAME DEBUG DIP	Soft dip for the setting during debugging of a game

- (2) Simulation of each specification is performed according to MODE SET.

(2)

TYPE	NORMAL	NEO-GEO
COUNTRY	JAPAN
.....
.	.	.
.....

TYPE	Neo-Geo/MVS
COUNTRY	Japan/USA/Europe-Asia

- * The Europe-Asia specification for the MVS is only for Southeast Asia (Korea).
If checking the MVS Europe, select USA.
 - * When checking the Neo-Geo, select Normal Neo-Geo.
- Start the game by setting the MODE, returning to screen 1, and pressing DEFAULT START (clear the backup area).

(3) Main Soft Dip (MVS unit setting)

(3)

MAIN SOFT DIP	
COIN 1	1 COIN = 1 CREDIT
COIN 2	1 COIN = 2 CREDIT
GAME SELECT	ONLY WHEN THERE IS CREDIT
GAME START	COMPULSION 30S
DEMO SOUND	SET FOR EACH GAME

- Start the game by making the selections with the A button and returning to screen (1) by the C button.
- * When inserting coins for the game, press Select + A for coin 1 and Select + B for coin 2.
- Match the setting of the initial specification of the version set in (2). (The MVS system matches the setting with the initial specification automatically.) * Refer to the MVS Regional Specification.
- Confirm that everything is created as specified for every version.
- * The meanings of coin 1, coin 2, start button, coin counter, and credit display differ depending on the version.
- * When a bug is detected during the coin check, check again with the MVS, and when the bug is not detected with the MVS, it is considered normal.
- * The demo sound will be off if it is in the MVS setting.
- Change the settings and confirm whether the system will operate properly.

(4) Game Soft Dip [Soft dip for each game (4)]

(4)

GHOST PILOTS	
HERO	3
CONTINUE	NO LIMIT
DIFFICULTY	3
BONUS	SECOND BONUS
BONUS LATE	10000/30000
HOW TO PLAY	WITH
DEMO SOUND	WITH
CREDIT DIPS	WITH

- Start the game by making the selections with the A button, returning to screen 1 using the C button and pressing RESTART.
- Confirm that the game operates as specified.
- Change the settings and confirm that the game operates properly.
- Confirm that it corresponds correctly with the MAIN SOFT DIP.

- (5) Debug Dip (Soft dips used during debugging by the software developers, changing the settings as necessary.)

(5)

DEBUG DIP	
	12345678
DIP 1	00000000
DIP 2	00000000

- Start the game by making the selections with the A button, returning to screen 1 using the C button, and pressing RESTART.
- In order to correspond with the bugs that appear with certain conditions, confirmations are made by reproducing the conditions using the debug dip.
- Make different settings by hypothesizing different conditions, and confirm that it operates correctly.
- ◎ For the simulation bug-checking, examine thoroughly such main areas as whether the game has been created correctly according to the specification, or whether there are any display errors or characters errors.
- * Refer to the MVS bug-checking list when checking.

<Measures to Take When a Bug Has Been Detected>

1. Version MVS/Neo-Geo, region, number of players
2. Date/Time Write down the date and time when the program ROM has been completed and when the bug has been detected.
3. Effects Write clearly the conditions of the bug. If taping the game on video, be sure to have a counter.
4. Cause Write clearly what scene, what operations were being done, and what condition the game was in when the bug occurred.
5. Frequency Reproduce the same conditions several times to confirm their frequency.

(* A= The bug always shows when the conditions are met.
B= The bug shows frequently.
C= The bug sometimes shows.
D= The bug rarely shows.)

- Note and comprehend the items 1-5, report to the planner, and submit this to the programmer.
- Always confirm the new program ROM when the bug has been eliminated.
- Confirm that no other parts of the software was effected by fixing this bug.
- Report to the programmer.

Location Testing

- What is location testing?

After the bugs have been removed and the game has been completed, location testing is done by actually setting up this game at the arcade to observe the reactions of the general users and income numbers to make final adjustments.

The location testing sometimes cannot be continued if a bug has been detected during location testing, and accurate data may not be obtained; so please make sure to perform the testing after all the bugs have been removed. The most problematic types of bugs during debugging are listed below:

- Bugs that stop the proceeding of a game such as the game being reset during play, the screen freezing, and the main character or enemies "falling" to a stage where they cannot move.
- Income record is not accurately factored.

Besides the data collection of income numbers, game play time, etc. the developers must also go to the arcade to check the popularity of the game, reactions of the players, difficulty levels, etc.

(General Examples)

- * How is the income?

The most important factor in location testing is the income. If the game does not have a good income during location testing, adjustments and improvements must be made to the game. It is also a problem if the initial income is high, but the income drops significantly after a week. It is best if the game maintains a stable high income for a long period of time.

- * What about the play time?

If the average play time is long, this may result in decrease of income because each player is playing for too long. Adjustments and improvements may also be necessary if the play time is too long during location testing. (The ideal play time that our company determined for an MVS game is 2 minutes 30 seconds to 3 minutes 30 seconds)

- * What about the attraction?

Check to see if it attracts people's attention during the location testing. The attraction will not be good if the game is not appealing as a new product.

* Is the game being understood?

Check to see that the player understands the operation methods, game objectives, and character intentions. If there are problems for the players in grasping the game, make appropriate adjustments, correct the display method, etc.

* How are the difficulty levels?

The difficulty level cannot be evaluated by just the average play time. Developers actually need to observe the players to evaluate this. If the game is too difficult (or easy), adjustments must be made. (Please be careful to differentiate between general players and expert players, to avoid making erroneous assumptions.)

* What about the number of continue plays?

The number of times the game is continues increases for more appealing games. It is best if the game makes the players want to continue the game.

* Are there any bugs?

Remove the bug immediately if a bug is found during the location testing. (If the bug causes problems in proceeding with the location testing, the location testing obviously cannot be continued. So please perform the debugging completely and carefully before the location testing.)

- If another development company uses our company's location testing sites, please have an observer (one who observe the location testing on a regular basis) provided by the development company. Also, make sure to notify our company of where we can reach the person in charge at the development company during the location testing. (Please do so for weekends and holidays also.)
- Reference: "Regarding Our Company's Location Testing" (February 1991 to present)

Corrections and improvements (remove the items that are not necessary) are made after a minimum of one week's worth of location testing data with our policy. Then, after the completion of appropriate adjustments the final version goes through another location testing for three days.

Do Not Copy



NEO-GEO BASE UNIT MANUAL FOR DEVELOPMENT

SNK

SNK
0062

This NEO-GEO base unit has been modified on the basis that it is to be used for development. The unit has NEO-MVS SYSTEM ROM which allows game debugging. Also it simulates arcade software with the home entertainment system.

USE OF NEO-MVS SYSTEM ROM

[Function]

Press 2P-A-BUTTON while pressing 2P-SELECT ⓧ enters COIN1

Press 2P-B-BUTTON while pressing 2P-SELECT ⓧ enters COIN2

(It is only valid in MVS mode. When in USA mode, COIN2 is PLAYER2-COIN)

2P-C-BUTTON while pressing 2P-SELECT ⓧ system menu

(When SYSTEM_MODE bit7 is 0 this function is disabled)

[System Menu]

- MODE SET
- MAIN SOFT DIP
- GAME SOFT DIP
- RESTART
- DEFAULT START
- GAME START TEST
- CARD FULL
- CHECK SUM
- BOOK KEEPING (CARD)
- GAME DEBUG DIP

(What follows is done through the 2P controller)

lever up, down	select command (the menu loops)
A button	command execute
C button	return to game
D button	to see game screen. pressing the D button again will return you to the system menu screen. this can be used as a pause.

* HOW TO USE THE COMMANDS

[MODE SET]

Switching for arcade, home system, Japan, America, Europe (also South East Asia). Also, it will display part of SYSTEM WORK.

TYPE COUNTRY	DEVEL. JAPAN	HOME USE	(1) (2)
xx WORK LIST xx			
USER REQUEST	00		(3)
USER MODE	00		(3)
PLAYER MODE	00	00 00 00	(3)
CREDIT	00	00	(4)
GAME CODE	0044		(5)

(1) TYPE

A, B button allows switching to DEV.HOME USE (development for home system), DEVEL.MVS (development for arcade), and NORMAL HOME USE (home system).

* DEVELOPMENT MODE AND NORMAL MODE

TYPE	MODE	SET UP	10FE80H(.W)	DIFFERENCE
Home system	Development	DEV.HOME USE	Other than 0	BACK UP AREA saved when RESET
	Normal	SYSTEM ROM for NORMAL HOME USE	0	Initializes when RESET
Arcade system	Development	DEV.MVS	Other than 0	Define CREDIT to 10FE00H
	Normal	Normal SYSTEM ROM	0	Define CREDIT to D00034H

(2) COUNTRY

With the A and B button, change the COUNTRY_CODE to, JAPAN (=0), USA (=1), EUROPE or EUROPE-ASIA (=2).

☞ You can change (1) and (2) with the up and down direction of the controller. Also you can return to System Menu with the C button.

(3) USER REQUEST, USER MODE, PLAYER MODE

Each SYSTEM_WORK details are displayed.

(4) CREDIT

CREDIT number is displayed. The addresses are 10FE00H and 10FE01H (these are valid only from 2P side or USA).

(5) GAME CODE

Displays GAME_CODE located at the address 108H.

[MAIN SOFT DIP]**[GAME SOFT DIP]**

Soft dip, used in the arcade system, allows changing the display with this selection. Please be careful when this function is chosen. Some of the ones which use "kanji" characters might not be properly displayed. For the USA use, COIN2 rate will be same as the coin rate at the time of continue. Moving the controller up and down will allow for the menu selection, A and B button to change the content, and C button to return to the System Menu.

[RESTART]

This choice will reset (initialize) the software. The parameters chosen at the MODE SELECTION will remain the same. Even with the hardware reset, MODE and SOFT DIP values will not be cleared. Power on reset of USER_REQUEST=0 will only be entered once at the time of start up even if the TYPE has been set for the home system. The only exception of this is when the TYPE has been chosen as the NORMAL HOME USE.

[DEFAULT START]

BACK UP AREA and SOFT DIP values are reinitialized after RESET.

[GAME START TEST]

With the MVS forced start MODE, USER_REQUEST=3 and USER is entered, and after 10 interrupts PLAYER_START is requested. If start is not accepted at this time, ERROR is displayed.

[CARD FULL]

Available memory of the MEMORY CARD (used for the games) are filled with dummy data. Please use this to check the memory card program.

[CHECK SUM]

The CHECK SUM of the 8 Mbit program area (8 bit, 16 bit) is displayed. You can not return to the System Menu. If the game ID is not found at the time of reset, the system will automatically go into this MODE. This will allow you to examine the CHECK SUM of the character ROM's.

[BOOK KEEPING (CARD)]

This displays the net income of the MVS system. (The operation method is the same as the MVS. You may not return to the System Menu.)

[GAME DEBUG DIP]

This will change the front 2 byte of the game BACK UP AREA to bit format. Please use this for development (ex: no death mode). C button will allow you to return to the System Menu

Notes On: NEO-GEO Base Unit for Development

- Development using NEO-GEO ☐ connect Jumper 2 (J2) with a solder.
- Debugging using NEO-GEO ☐ disconnect Jumper 2 (J2).

DO NOT COPY

**NEO GEO SOUND PROGRAM
(SOUND2.REL)
USER'S MANUAL**

SNK

YM2610 CHIP OUTLINE

MVS, NEO-GEO circuit boards uses the YM2610 chips made by YAMAHA. This chip has the capacity to output the following at the same time: 7 voices ADPCM (voice synthesis), FM source 4 voices, and SSG 3 voices (PSG compatible). Explanation of all three follows.

ADPCM

This is voice synthesis (PCM). There are two types, ADPCMA and ADPCMB, and the differences are listed in a table below.

	Output	Sampling Rate	Data ROM	The Seconds on 4M ROM
ADPCMA	6 voices	18.5 kHz fixed	4M mask ROM	approximately 1 second
ADPCMB	1 voice	2.0-55.5 kHz	4M mask ROM	lowering the quality, 2 sec.

The sampling rate listed above maybe analogous to the tape recorder speed. The larger the value, the faster the rotating speed. The higher the sampling rate, the better the quality, requiring more memory. In contrast, the lower the value, the lower the quality; but the memory that is required is less. Also, changing this value at the time of reproduction will change the interval. Even though ADPCMA is fixed, ADPCMB is not and recording a musical instrument will allow the use of the sounds as BGM.

FM Source

FM allows 4 voices output at one time. Compared to the previously used FM source of the YM3812 chip, the output is more realistic.

SSG Source

This is compatible to the PSG, and allows 3 voice output with noise mixing of rectangular wave form.

(1) This source chip is in stereo. There is a choice from Lch (Left channel) output, Rch (Right channel) output, or L+Rch (both channel) output. If a specific sound is required, there might be the need to use two voices and manipulation of each level settings.

(2) Dividing the sources: These sound sources are divided into Sound Effect and BGM and are shown in the table below.

BGM	ADPCMA 3 voices, FM 4 voices	ADPCMB 1 voice
Sound Effect	ADPCMA 3 voices, SSG 1 voice	ADPCMB 1 voice

ADPCMB is compatible to both BGM and Sound Effect. It is used for one or the other depending on the game.

Files needed for NEO-GEO sound development are included in the NEO-GEO Sound Program floppy disks.

FILE NAME	CONTENTS
SOUND2.REL	sound driver
BANK.REL	copy program
WINDOW.COM	copy program
SDATA5.DAT	table including data
SDATA6.DAT	table including data
VO_FM.SRC	music or sound effect data source file
VO_PCMA.SRC	music or sound effect data source file
VOPCMB.SRC	music or sound effect data source file
TBL_PCMB.SRC	music or sound effect data source file
VO_HOST.SRC	music or sound effect data source file
TBL_MU.SRC	music or sound effect data source file
OAF0.SRC	OA??.SRC initial source file
OBF0.SRC	OB??.SRC initial source file
CONFIG20.SRC	initial music source file
CONFIG_S.SRC	initial SSG source file
YM5F.SRC	eye catch music data file
SM7F.SRC	coin sound data file
EQU4.ASM	file which enters EQU specification
SD.BAT	batch file used during assembling and linking
SDB.BAT	batch file used during assembling and linking
SB.BAT	batch file used during assembling and linking
INIT.MCR	macro file used during PROICE operation
ANDO.COM	program which transfers data from the host computer to the ROM writer
ANDO.EXE	program which transfers data from the host computer to the ROM writer

CREATING NEO-GEO YM2610 PCM

Please follow the process listed below if Macintosh is to be used to create the NEO-GEO PCM.

1. Using an AKAI sampler, sample the needed sound effects and musical instruments. Please record the bass drums, bass and other low frequency instruments at 44.1 kHz; and record high frequency instruments, such as the cymbals, at half the sample rate mentioned earlier. (If the instrument does not seem to be in either class of frequency, please try with both the sampling rate. Depending on which rate chosen will determine the sound value that will be reproduced.)
2. Save as a file in Macintosh using Alchemy.
3. Convert the Macintosh files using a file converter or Ether Net (Tops) into PC98 DOS format.
4. Convert the PC98 DOS format file to YM2610 compatible file using ADPCM compressor (provided by our company) and burn in the data using the ROM writer. Please refer to the ADPCM compressor user manual provided separately.
 - * The pitch table specifying ADPCMB scale is sampled at 12kHz. Please set the ADPCMB instrument sound at 12 kHz (or 24 kHz).
 - * The address value of ADPCM data created are input in VO_PCMB.SRC and VO_PCMA.SRC. The method of address value calculation is shown below.
 - (1) Find out the ADPCM data file size using the MS-DOS command "DIR."
 - (2) $\boxed{\text{File size}} \div 8 = \boxed{A}$
 - (3) Change \boxed{A} into HEX form $\boxed{A'}$
 - (4) $\boxed{A'} \div 10H = \boxed{A''}$ This value is the file size entered into in VO_PCMB.SRC and VO_PCMA.SRC.
 - * Please transfer the ADPCM data, grouped in 1 Mbits (or 4 of these grouped in 4 Mbits), with the use of the MS_DOS command "COPY" and the switch "/B" to the ROM writer.

ADPCM Data Entry & Edit

File Name: VO_PCMA.SRC

Entry & Edit into this file the sound data PCMA created by the YM2610 ACQUISITION UNIT & OPNB SYNTHESIZE UNIT (abbreviated as PCMA from now on).

Macro Definition of "TBL_KEY_PCMA," "TBL_KEY_PCMA2," and "TBL_KEY_PCMA3."

TBL_KEY_PCMA	;MAIN PCMA VOICE TABLE \$00~\$FF ;(DON'T USE VOICE00A_1~VOICE1F_1 FOR 2 BITE CODE)	
;E1=18,VO_No		
;D1=14,VO_No		
① ② ③ ④ ⑤ ⑥ ⑦ ⑧	
;VOICE00A	VOICES	01,0000,0001,00,0000,0001,PC,1F ;(COMMENT)
;VOICE01A	VOICES	01,0000,0001,00,0000,0001,PC,1F ;(COMMENT)
;VOICE02A	VOICES	01,0000,0001,00,0000,0001,PC,1F ;(COMMENT)

Every row has "VOICE??A," "VOICES" and eight number values. These eight values are the parameters for the "VOICE??A." Each parameter will be explained below.

- 1- Will enter the "VOICE??A" (sound from PCMA) in priority order. The value is from 0 to FF. The lower the numerical value, higher the priority.
- 2- Will enter the numerical value of the lower two figures cut off of the start address entered during PCMA data creation. The maximum numerical value is FFFF.
- 3- Will enter the numerical value of the lower two figures cut off of the end address entered during PCMA data creation. The maximum numerical value is FFFF.
- 4- Number of loops (repeat) of the "VOICE??A" (sound from PCMA) is specified. The value is from 0 to FF. The value FF corresponds to infinite loop.
- 5- Will enter the start address (cutting off the lower two figures) when looping.
- 6- Will enter the end address (cutting off the lower two figures) when looping.
- 7- Will define the output destination of the "VOICE??A" (sound from PCMA).
R channel output ↗ enter PR
L channel output ↗ enter PL
L and R channel output ↗ enter PC

- 8- Sets the volume level of the “VOICE??A” (sound from PCMA). The value is from 00 to 1F. 1F is the maximum.

* Macro Definitions “TBL_KEY_PCMA,” “TBL_KEY_PCMA2,” and “TBL_KEY_PCMA3” have “VOICE??A” values of 0 to FEh. With 255 for each, a maximum of 765 PCMA sound may be entered.

Macro Definition of “PCMA_EI_TBL,” “PCMA_EI_TBL2,” and “PCMA_EI_TBL3.”

PCMA_EI_TBL	①	②
;	01234567 89ABCDEF	
DB	00000000B,00000000B	;00
DB	00000000B,00000000B	;10
DB	00000000B,00000000B	;20
DB	00100000B,00000000B	;30
DB	00000000B,00000000B	;40
DB	00000000B,00000000B	;50
DB	00000000B,00000000B	;60
DB	00000000B,00000000B	;70
DB	00000000B,00000000B	;80
DB	00000000B,00000000B	;90
DB	00000000B,00000000B	;A0
DB	00000000B,00000000B	;B0
DB	00000000B,00000000B	;C0
DB	00000000B,00000000B	;D0
DB	00000000B,00000000B	;E0
DB	00000000B,00000000B	;F0
;	01234567 89ABCDEF	

There are three tables in “VO_PCMA.SRC”, similar to the table above. The table “PCMA_EI_TBL” is the DI, EI table (switch determining output of “VOICE??A,” output or not) corresponding to “TBL_KEY_PCMA;” “PCMA_EI_TBL2” to “TBL_KEY_PCMA2;” and “PCMA_EI_TBL3” to “TBL_KEY_PCMA3.” If PCMA sound is used as a sound effect, this DI, EI table is used. (If PCMA sounds are used as instrumental sounds, there is no need for the use of this table.)

① determines the first significant place “?” in “VOICE??A,” ② determines the second significant place. (For example, the table above gives “VOICE32A.”) The current table is “PCMA_EI_TBL,” but this look up method is the same for other tables. The value, in the above case “1,” determines if there is an output. When the numerical value “0” signifies “DI” (no sound output), and value of “1” signifies “EI” (sound output).

To output “VOICE??A” sound during a game, “1” must be entered in the corresponding “??” table cell, and the following codes must be sent from the main program:

If numerical value “1” of “VOICE??A” is entered in “TBL_KEY_PCMA” “18”

If numerical value “1” of “VOICE??A” is entered in “TBL_KEY_PCMA2” “1A”

If numerical value “1” of “VOICE??A” is entered in “TBL_KEY_PCMA3” “1C”

To interrupt the sound output of “TBL_KEY_PCMA,” please send “14;” and for “TBL_KEY_PCMA2,” please send “15.”

Ex:

To output sound "VOICE32A" of "TBL_KEY_PCMA2" from the main program, send "1C, 32."

(Precautions)

- 1 Please do not enter sound effects in "VOICE00A" through "VOICE1FA" of "TBL_KEY_PCMA." (This will conflict with the Sound Program System Codes.) Also, please do not enter "1" in 00 through 1Fh of "PCMA_EI_TBL." (These areas maybe used for instrumental sounds.)
- 2 Only sound effects are allowed to be entered in "TBL_KEY_PCMA2" and "TBL_KEY_PCMA3."
- 3 Please do not enter sound effects in "VOICE00A" through "VOICE1FA" of "TBL_KEY_PCMA2" and "TBL_KEY_PCMA3." (This will conflict with the Sound Program System Codes.)

File Name: VO_PCMB.SRC

Entry & Edit into this file the PCMB sound data created by the YM2610 ACQUISITION UNIT & OPNB SYNTHESIZE UNIT (abbreviated as PCMB from now on).

Macro Definition of "TBL_KEY_PCMB."

TBL_KEY_PCMB		;MAIN PCMB VOICE TABLE \$00~\$FF																							
	 ① ② ③ ④ ⑤ ⑥ ⑦ ⑧																							
VO1CE00		VOICES	00, 00, 00, 00, 00, 00, 00, FF	VOICESLF0	00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00	⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱																			
VO1CE01		VOICES	00, 00, 00, 00, 00, 00, 00, FF	VOICESLF0	00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00																				
VO1CE02		VOICES	00, 00, 00, 00, 00, 00, 00, FF	VOICESLF0	00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00																				

The numerical values listed in the rows "VOICES," 1 through 8, and "VOICESLF0," 9 through 18, define the parameter of "VOICE??" The 16 numerical values will be explained.

- 1- Will enter the "VOICE??B" (sound from PCMB) in priority order. The value is from 0 to FF. The lower the numerical value, higher the priority.
- 2- Will enter the numerical value of the lower two figures cut off of the start address entered during PCMB data creation. The maximum numerical value is FFFF.
- 3- Will enter the numerical value of the lower two figures cut off of the end address

entered during PCMB data creation minus "1." The maximum numerical value is FFFF.

- 4- Number of loops (repeat) of the "VOICE??B" (sound from PCMB) is specified. The value is from 0 to FF. The value FF corresponds to infinite loop.
- 5- Will enter the start address (cutting off the lower two figures) when looping.
- 6- Will enter the end address (cutting off the lower two figures) when looping.
- 7- Will enter the frequency of the ADPCM data created during sampling. The calculated sampling rate must be entered at the top.

(Sampling Rate Calculation Method)

$$PB = \frac{65536 \times FB}{55.5}$$

PB = pitch FB = frequency
(sampling rate) (kHz)

Ex: Calculation of sampling rate with sampling frequency of 8 kHz.

$$PB = \frac{65536 \times 8}{55.5} = 9446.630\ldots$$

rounding off to the nearest whole number $\lceil 9447 \rceil$

Enter this numerical value at the top of "VO_PCMB.SRC" as shown below.

K8 EQU $\lceil 9447 \rceil$
SPRCE

If the sound created by YM2610 ACQUISITION UNIT & OPNB SYNTHESIZE UNIT is divided into 16 kHz, 8 kHz, and 18 kHz sampling frequency; calculate and enter the sampling rate values for 16 kHz, 8 kHz, and 18 kHz sampling frequency and enter it as shown above.

The "K8" in front defines the sampling frequency as "8 kHz."

- 8- Defines the volume of "VOICE??" (PCMB sound). The values ranges from 0 to FF. Larger the value, larger the volume.
- 9- Using LFO on "VOICE??" (PCMB sound), the value here specifies SYNC DELAY TIME (time delay before LFO is used). The values ranges from 0 to FF. LFO is used as the output begins if the value is 0. Larger the value, longer the delay time.
- 10- Defines the LFO envelope type. There are ten envelope types to choose from and are shown below.

(Shown on next page)

The numerical value listed in the table determines the first significant place value, while the second significant (tenth) place determines the "VOICE??" (PCMB sound) output destination.

R channel output $\lceil 1 \rceil$

L channel output $\lceil 2 \rceil$

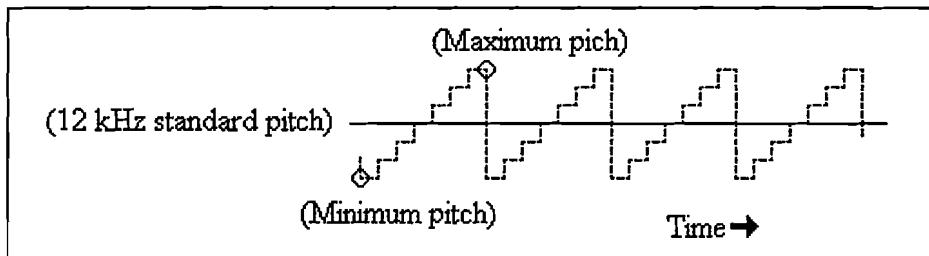
L and R channel output $\lceil 3 \rceil$

No.	Envelope Shapes
1-3	↑ _____
4-7	↑ _____
8	~~~~~
9	↑ _____
A	~~~~~
B	N _____
C	~~~~~
D	/ _____
E	~~~~~
F	↑ _____

11- Minimum.

12- Maximum pitch.

LFO depth is defined here. Please see the drawing below for visual aid.
Ex: PCMB sound with sampling frequency of 12kHz.



Using the calculation used in 7 with 12 kHz, the sampling rate should be 14170. This value is the 12 kHz standard pitch shown in solid line from in the drawing above. Addition of a value ?? to 14170 results in the Maximum pitch and negation of the depth results in the Minimum pitch. The Minimum and Maximum pitch define the depth of the LFO.

(Input method)

$$\begin{array}{ll} \text{Maximum pitch} & \text{?} 14170 + \boxed{\quad ?? \quad} = (1) \\ \text{Minimum pitch} & \text{?} 14170 - \boxed{\quad ?? \quad} = (2) \end{array}$$

?? value is the LFO depth. Change value (1) to base 16 and input into 11, (2) to base 16 and input into 12. The numerical value ranging from 0 to FFFF.

13- Defines the LFO speed. The value ranging from 0 to FFFF. The larger the value the faster the speed. When the value is 0, LFO will not be used.

14~18 Defines the volume portmanteau (Please think of this as the attack portion of the volume envelope.) of "VOICE???" (PCMB sound).

14- Defines the type of volume portmanteau.

There is no volume portmanteau when the first significant number is "0."

The volume is lowered when the first significant number is “1.”
The volume is raised when the first significant number is “2.” (one place)

- 15- Defines the time (speed) required from the start volume to the end volume. The numerical value ranges from 0 to FF. Smaller the value the faster the speed. When the value is set to 0, as the output of sound begins, the end volume level is reached with the effect being nullified.
- 16- The initial volume level that is to be modified is defined. (This parameter is useless when the numerical value “1” [where the volume is lowered] is chosen at 14.) The value ranging from 0 to FF. Larger the value, larger the level.
- 17- The final volume level of the sound that is to be modified is defined. The value ranging from 0 to FF. Larger the value, larger the level.
- 18- Defines the speed (chosen in 15) increase. (In other words, to define the speed of volume portmanteau, both 15 and 18 must be define as positive values.) The value ranging from 0 to FF. Larger the value, larger the increase of speed. When the value is 0, there is no modification.

Macro Definition “PCMB_EI_TBL,” and “PCMB_EI_TBL2.”

The contents are similar to “PCMA_EI_TBL,” “PCMA_EI_TBL2,” and “PCMA_EI_TBL3.” Please input accordingly. However, there are few differences which are listed below.

- * The voice EI code entered in “PCMB_EI_TBL” is “19.”
- * The voice EI code entered in “PCMB_EI_TBL2” is “1B.”

(Sending in the following order will result in sound output: EI code, voice number.)

(Sample) ADPCM hardware address, OPNB SYNTHESIZE UNIT address, and address relationship inside the files (such as relationship between “VO_PCMA.SRC” and “VO_PCMB.SRC”).

1- Word count of OPNB SYNTHESIZE UNIT from 0~4000H data is 1 Mbit.

2- 400H of address input area inside of “VO_PCMA.SRC” & “VO_PCMB.SRC” is 1 Mbit.

<Sample>	(Start Address)	(End Address)
1 M	0	~ 400 (H)
2 M	400 (H)	~ 800 (H)
3 M	800 (H)	~ C00 (H)
4 M	C00 (H)	~ 1000 (H)
5 M	1000 (H)	~ 1400 (H)
6 M	1400 (H)	~ 1800 (H)
7 M	1800 (H)	~ 1C00 (H)

8 M	1C00 (H)	~	2000 (H)
9 M	2000 (H)	~	2400 (H)
:	:		:
:	:		:

- 3- There is no difference on the NEO-GEO system between ADPCMA data ROM and ADPCMB data ROM. Access point address values of “VO_PCMA.SRC” and “VO_PCMB.SRC” are the same. Please make sure not to overlap “VO_PCMA.SRC” and “VO_PCMB.SRC” address values. (Please be aware, that there is no error message display when there are address value overlaps or if the same values are entered.)
- To avoid confusion, please organize ADPCMA data (ROM) and ADPCMB data (ROM). (Each data and ROM should be placed separately as units consisting of one data per ROM.)
- 4- During development, the ADPCMA and ADPCMB data ROM hardware position will probably change. At this time the address input value must be changed in “VO_PCMA.SRC” & “VO_PCMB.SRC” files. The example below illustrates the method of shifting the address values entered in files “VO_PCMA.SRC” & “VO_PCMB.SRC” by the same amount.

Ex:

In this case, there is a need to move every address values by 1 M (i.e. each address value requires an addition of 400H). If the Macro Definition “VOICER MACRO” of “VO_PCMB.SRC” is rewritten as follows (Figure 1), there is no need to rewrite each address values.

Figure 1

VOICER	MACRO	&PI,&ST,&EN,&CT,&FD,&ED,&PT,&LV
	DB	\$&PI
	DW	\$&ST/2+200H
	DW	\$&EN/2-1+200H
	DB	\$&CT
	DW	\$&FD/2+200H
	DW	\$&ED/2-1+200H
	DW	&PT
	DB	\$&LV
	ENDM	

Because of Macro Definition purposes, 200H is used to calculate 1 M. For example, increasing the value 200H to 400H will equal 2 M, and increasing to 600H results in 3 M of address shift for a file in “TBL_KEY_PCMA?”

* This can be done for either “VO_PCMA.SRC” or “VO_PCMB.SRC.”

* “ADPCM” ROM should be installed starting from “V1” (from address 0000H), “VO_PCMB.SRC” address input starting from 0000H as well, and depending on the number of ADPCM ROM’s, the above operation should be done in “VO_PCMB.SRC.”

File Name: “TBL_PCMB.SRC”

This allows the instrumental sounds entered in “VO_PCMB.SRC” to be used in MUSIC.SRC.

Macro Definition “TBL_OKEY_PCMB”

Ex:

TBL_OKEY_PCMB										~	+F
	① +0	+1	+2	+3	+4	+5	+6	+7	+8	~	+F
DB	\$18,\$00,\$00,\$00,\$00,\$00,\$00,\$00									~	\$00 ; +\$40
DB	\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00									~	\$00 ; +\$50
DB	\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00									~	\$00 ; +\$60
DB	\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00									~	\$00 ; +\$70

The VOICE NUMBER (the ?? of “VOICE??”) of the PCMB instrumental sounds created in “VO_PCMB.SRC” should be entered in the dotted line box. The first entry in the above example is “18.” This is the VOICE NUMBER of “VO_PCMB.SRC.” And the VOICE NUMBER that is corresponding to “18” in “TBL_OKEY_PCMB” is “40.” (① determines the first significant value, +0, and ② determines the second significant value, +40 ≠ 40.) Entering this value into the prescribed position of MUSIC.SRC, and entering the musical score will produce musical sounds with VOICE NUMBER “18.” (It will be discussed further in “Creating MUSIC.SRC.”)

Macro Definition “TBL_OCTAVE_PITCH”

This is a pitch table that adds scale to each sound that was entered into Macro Definition “TBL_OCTAVE_PITCH.” The scale has been adjusted with a sampling rate of 12 kHz. (Because of this, it is possible to use the instrumental sounds with sampling rate of 24 kHz.) If there is a need to use different sampling rates, this is where the changes should be entered. But it is our recommendation to use this sampling rate of 12 kHz.

Macro Definition “TBL_MKEY_PCMB”

Normally, sampling one note that the musical instrument produces, and playing it with a wide range of frequencies, the lowest and highest frequency output would be interpreted by the human ears as coming from different instruments. In order to compensate for this failure to emulate a real instrument, the following steps should be taken. First, sample the

desired instrument sound. Second, sample the sound one octave (two octave in some cases) at a time two to three times higher and lower than what was sampled in the first step. (Number of times higher and lower can be changed according to how wide a range the musical score is.) Finally, the sounds that were sampled should be entered separately into “VO_PCMB.SRC,” and entering also in “TBL_MKEY_PCMB” results in a score sounding as if it was coming from a real instrument rather than being simulated. (Of course this will require more memory.)

Ex:

TBL_MKEY_PCMB		; MAIN \$80~\$BF :type multi								
;		octave	0	1	2	3	4	5	6	7
PCMB_KEY	① 80	DB	\$00,\$00,\$00,\$21,\$22,\$23,\$00,\$00	②	③	④				
PCMB_KEY81		DB	\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00							
PCMB_KEY82		DB	\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00							
.	
.	

Each row has “PCMB_KEY??,” “DB,” and 8 parameters with octave numbers from 0 to 7. Each parameter represents an octave. “octave 0” represents the lowest octave and “octave 7” represents the highest octave. Enter the VOICE NUMBER of PCMB in one of the octave locations. In the above example, ② “\$21,” ③ “\$22,” and ④ “\$23” are the VOICE NUMBER created in “VO_PCM.SRC.” (Assuming each sample range is 1 octave, this example allows instrumental output over three octave range.) “80” represents these parameters. Entering this numerical value in the prescribed area of MUSIC.SRC, the creation of the musical score is initiated.

The above description is what “TBL_MKEY.PCMB” is normally used for, but it also maybe used as a “split” (i.e.: It can be used to separate a musical interval into high and low frequencies).

Macro Definition “TBL_MULTI.PITCH”

This is a pitch table that adds scale to each sound that was entered into Macro Definition “TBL_MULTI.PITCH.” The sampling rate of this scale is 12 kHz as well. Please use this table as is.

File Name: "VO_HOST.SRC"

MUSIC, EFFECT that are to be outputted from the game, must receive the code from the main CPU. The file “VO_HOST.SRC” is a table which stores the code of MUSIC, EFFECT.

Macro Definition “TBL_EFFECT_PCMB”

Enter the VOICE NUMBER of PCMB sounds of EFFECT, entered (or edited) in file "VO_PCMB.SRC," to correlate with the codes.

Ex:

TBL_EFFECT_PCMB		;HOST REQUEST \$80~\$BF								;type HOST request				
		①	0	1	2	3	4	5	6	7	8	~	F	
DB		\$0A,\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00										~	\$00	② ;8
DB		\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00										~	\$00	;9
DB		\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00										~	\$00	;A
DB		\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00										~	\$00	;B

Enter (inside the dotted line box) the VOICE NUMBER of PCMB sounds of EFFECT entered and edited in “VO_PCMB.SRC.” The only non-zero entry inside the dotted line box is “\$0A;” the corresponding code for this value is “80.” ① determines the first significant value, 0, and ② determines the second significant value, 8 & 80.)

Macro Definition “TBL EFFECT PCMA”

It is similar to “TBL EEFECT PCMB.” Please refer to the section above.

64 sounds with code 80~BF in “TBL_EFFECT_PCMB” may be entered and 64 sounds with code C0~FF in “TBL_EFFECT_PCMA.” (If there are more 64 sounds, please use the method discussed earlier in “PCMA EI TBL.”)

Macro Definition “TBL_MCODE”

This is to open each code of the MUSIC, EFFECT. Please think of this as the switch to output the sound of MUSIC, EFFECT.

Ex.

```

DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; 50
DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; 60
DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; 70
DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; 80
DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; 90
DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; A0
DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; B0
DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; C0
DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; D0
DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; E0
DB      2,2,2,2,2,2,2,0,0,0,0,0,0,0,2 ; F0
;*
;*
          0 1 2 3 4 5 6 7 8 9 A B C D E F

; NO ENTRY      = 0
; SYSTEM CODE   = 1
; MUSIC ENTRY   = 2
; EFFECT PCMA   = 3
; EFFECT PCMB   = 4
; EFFECT SSG    = 5

```

① determines the first significant value, 0, and ② determines the second significant value, 20 ~ 20, which is the code corresponding to the flag (switch, “0”).

A code with a value from 00~1F is a SYSTEM CODE (related to sound program) and is not normally modified here.

A code with value from 20~5F is a MUSIC.SRC flag and requires a value of “2” in the flag section to be active. (MUSIC.SRC related codes will be explained later.)

A code with value from 60~7F is a S. S. G. SRC flag and requires a value of “5” in the flag section to be active. (S. S. G. SRC related codes will be explained later.)

A code with value from 80~BF is a PCMB EFFECT sound flag and requires a value of “4” in the flag section to be active.

A code with value from C0~FF is a PCMA EFFECT sound flag and requires a value of “3” in the flag section to be active.

Please specify all unused codes to be “0” (if it is not set to “0,” it will affect the sound program).

◆ Steps Taken from YM2610 ACQUISITION UNIT & OPNB SYNTHESIZE
UNIT Instrumental and EFFECT PCMA, B Sound; To Assembling In The Sound
Program: And Finally To Sound Output.

(1) Find the instrumental and or EFFECT sound; sample the sounds using YM2610
ACQUISITION UNIT & OPNB SYNTHESIZE UNIT and store this data into the
ROMs.

(2) Store the address of the sampled sound in “VO_PCMA.SRC” or
“VO_PCMB.STC.”

(EFFECT SOUND)

Inside “VO_HOST.SRC,” specify the code
for PCMA sound Macro Definition “TBL_
EFFECT_PCMA” and for PCMB sound
Macro Definition “TBL_EFFECT_PCMB”
and open flags for Macro Definition
“TBL_MCODE.”

(INSTRUMENTAL SOUND PCMB)

Enter the VOICE NUMBER for PCMB
instrumental sounds in
“TBL_PCMB.SRC” and convert it to
numerical values for MUSIC.SRC.

◆ Enter into MUSIC.SRC

◆ Send code for sound output

The method shown above is the basic flow
for PCM data entry.

(Caution) For EFFECT sound output method discussed in Macro Definition
“PCMA_EI_TBL,” “PCMA_EI_TBL2,” and “PCMB_EI_TBL3” also exist.
Thus there are two methods including the method shown above. (Both
methods are valid.)

* MUSIC, SRC S. S. G. SRC Entry

MUSIC.SRC must be entered before MUSIC DATA, S. S. G. DATA. (Similarly for S. S. G. SRC) Please follow the steps taken below.

First, create an initial MUSIC, SRC file. Copy file "CONFIG20.SRC" in MS DOS. ("CONFIG20.SRC is the template for MUSIC, SRC.)

Ex:

① ②
>COPY CONFIG20.SRC HM21.SRC

① --- Enter the two alphabet character of choice.

② --- Enter a numerical value from 20~5F. (This number will correspond with CODE.)

◎ Copy "CONFIG_S.SRC" for S. S. G. SRC, and use alphabet characters other than those used in the MUSIC.SRC. The numerical values should chosen from 60~7F.

Enter the MUSIC.SRC file HM21.SRC into the following three files:

"TBL_MU.SRC"
"SDATA5.SRC"
"VO_HOST.SRC"

◎ Enter S. S. G. SRC into the three files as well.

File: "TBL_MU.SRC"

Macro Definition "TBL_MUSIC"

Table which corresponds with MUSIC.SRC.

Ex:

②	①
DEFW	0 ; MUSIC20
	④ ③
DEFW	_____ MUSIC21
DEFW	0 ; MUSIC22
DEFW	0 ; MUSIC23

The value of ① ranges from 20~5F (maximum of 63 songs) and there are "0 ;" (②) corresponding to this value. For example, entering "HM21.SRC" into this table, enter the value 21 into ③ and erase the value "0 ;" as in ④. (Please enter S. S. G. SRC into Macro Definition "TBL_SSG_MUSIC" using a similar method as was shown.)

File: "SDATA5.DAT"

File which collects all files associated with sound. MUSIC. SRC and S. S. G. SRC are files and must be taken account of in "SDATA5.DAT."

Ex:

```
=====
Play MML data
=====
① INCLUDE    TBL-MU.SRC
② INCLUDE    HM21.SRC
.
.
```

Please scroll down to "Play MML data." Please enter after ①(INCLUDE TBL_MU.SRC) as in ②. With each increase of MUSIC.SRC, please enter "INCLUDE__??" in numerical order. Please follow the same steps for S. S. G. SRC.

File: "VO_HOST.SRC"

The details are as they were discussed earlier. Please enter the CODE number for MUSIC.SRC, "2," or S. S. G. SRC "5," into Macro Definition "TBL_MCODE." (Ex: MUSIC.SRC for the above example would have the value "2" at location "21")

✿ Music Score Data (MUSIC.SRC) Creation

Please follow the steps listed below before entering the music score data.

“HM21.SRC” will be used as an example. The “HM21.SRC” file will initially be as shown below, a copy of “CONFIG20.SRC,” but the file name and data inside will not be matching. Please change the values in to “21” and to “MUSIC.” (This must be done every time a new MUSIC.SRC is created.

***** SOUND CONDUCTER CONFIG.SYS FILE *****

1
CONFIG[20]

EQU \$

***** PLAYING MODE *****

DB \$00 ; SELECT CODE

***** DATA ENTRY POINT ADDRESS TABLE *****

DW SCORE[20]_1 ; PART 1

DW SCORE[20]_2 ; PART 2

DW SCORE[20]_3 ; PART 3

DW SCORE[20]_4 ; PART 4

DW SCORE[20]_5 ; PART 5

DW SCORE[20]_6 ; PART 6

DW SCORE[20]_7 ; PART 7

DW SCORE[20]_8 ; PART 8

DW SCORE[20]_9 ; PART 9

DW SCORE[20]_10 ; PART 10

DW SCORE[20]_11 ; PART 11

***** TEMPO(MUSIC) OR PRIORITY(EFFECT) PARAMETER **

Music score creation will be discussed below. The file shown is the initial state of MUSIC.SRC.

```
;***** SOUND CONDUCTER CONFIG.SYS FILE *****  
;  
CONFIG20      EQU      $  
;  
;***** PLAYING MODE *****
```

```

;
;           DB      $00          ; SELECT CODE
;

***** DATA ENTRY POINT ADDRESS TABLE *****

;
DW      SCORE20_1      ; PART 1
DW      SCORE20_2      ; PART 2
DW      SCORE20_3      ; PART 3
DW      SCORE20_4      ; PART 4
DW      SCORE20_5      ; PART 5
DW      SCORE20_6      ; PART 6
DW      SCORE20_7      ; PART 7
DW      SCORE20_8      ; PART 8
DW      SCORE20_9      ; PART 9
DW      SCORE20_10     ; PART 10
DW      SCORE20_11     ; PART 11

;
***** TEMPO(MUSIC) OR PRIORITY(EFFECT) PARAMETER *****

;
TEMPO20    EQU      $  

;           DB      $00  ⌈ ①
;

***** TOTAL LEVEL *****

;
LEVEL20    EQU      $  

DB      $0C  

DB      $3F  ⌈ ②      ; initial PCMA total level  

DB      $50      ;
;

***** OPEN / CLOSE FLAG TABLE *****

;
;           OPEN (01,00)  CLOSE(00,00)
;

DB      $00,00      ; PART 1
DB      $00,00      ; PART 2
DB      $00,00      ; PART 3
DB      $00,00      ; PART 4
DB      $00,00      ; PART 5
DB      $00,00  ⌈ ③      ; PART 6
DB      $00,00      ; PART 7
DB      $00,00      ; PART 8
DB      $00,00      ; PART 9
DB      $00,00      ; PART 10
DB      $00,00      ; PART 11

```

; ***** FM Config.sys *****

SCORE20_1	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40
SCORE20_2	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40
SCORE20_3	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40
SCORE20_4	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40
SCORE20_5	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40
SCORE20_6	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40
SCORE20_7	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40
SCORE20_8	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40
SCORE20_9	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40
SCORE20_10	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40
SCORE20_11	EQU	\$
	DB	\$01,\$40,\$00
	DB	\$40

④

⑤

⑥

- ① --- Defines tempo, values ranging from 00~FF. The larger the value the faster the tempo.
- ② --- Defines the total level of volume of FM sound, PCMA sound, and PCMB sound. Top value ranging from 0~127 and is for the total level of FM sound. Smaller the value larger the volume.
 Second value ranges from 0~3F and is for the total level of PCMA sound.
 Because EFFECT sound created in PCMA sound will also be read in, please leave the level at 3F.
 Bottom value ranges from 0~FF and is for the total level of PCMB sound. The larger the value, larger the volume.
- ③ --- This is the switch to OPEN and CLOSE (to output or not) of each SCORE for ④, ⑤, and ⑥.

“\$00,00” ↗ OFF “\$01,00” ↗ ON

- ④ --- Input method for FM sound notes

The area boxed is the score data input area and is for 4 scores. Each score has capacity to output one sound (at one time).

(Part 1) Inputting note data

One note data has the following: note length, volume & tie specification, and octave & key. Each numerical values are hexadecimal.

Ex:

DB	\$18	\$88	\$4C
(note length)	(volume & tie specification)	(octave & key)	

(note length)

The numerical values range from 00~30, and correspond to notes as follow.

1/2 note ↗ 30	1/4 note ↗ 18
1/8 note ↗ 0C	1/16 note ↗ 06
1/32 note ↗ 03	dotted 1/4 note ↗ 24
dotted 1/8 note ↗ 12	dotted 1/16 note ↗ 09
3 notes per 2 beat ↗ 03 (quarter note triplet)	3 notes per beat ↗ 24 (triplet)
six notes per beat ↗ 04 (six group note)	

(volume & tie or a rest)

volume ranges from 80~BF, smaller the value, larger the volume. If the note is a tie, the range is from 00~3F (negate 80 from volume range). Also, for a rest, use the value 40.

Tie example: \$18, \$9A, \$4C (notes before tie)	\$18, \$1A, \$4C (notes after tie)
Changing the value (octave & key) of the note after the tie note will make a (musical) slur.	

(octave & key)

An octave is defined by the top 4 bits (tenth place). The range is 7 octaves, with numerical value of 0~E (even values).

A key is defined by the bottom 4 bits (one place). The numerical value is chosen from 0~F with the following relationship.

value	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
note	G	C [#]	D [#]		F [#]	G [#]	A [#]				A	B	C	D	E	F

(Part 2) Inputting tone, total level of each score, and total octave of each score.

Ex:

DB	\$8A	\$84	\$84
(tone)	(total level)	(total level)	(total octave)

These three consists a set that should be inputted before note data.

(tone)

The values here correspond with the Voice Number of the FM tone created in “VO_FM.SRC.”

(total level)

The total level of each score is defined here with range from 80~9F. Smaller the value, the larger the volume.

(total octave)

The total octave of each score is defined here with ranges 80~8F. 80 is the base value with even number values (2, 4, 6, 8, A, C, E) corresponding to a raise of an octave per interval. To lower an octave, odd values (3, 5, 7, 9, B, D, F) should be used; each interval corresponds to one octave being lowered.

The three numerical values should be inputted to FM Score in the beginning, but if there is a need to have a different value, it is possible to change.

⑤--- Inputting PCMA Scores

The 6 Scores boxed in is the PCMA sound score input section. Normally, Score??5~Score??7 is used. (PCMA sound may be outputted with 6 sound

simultaneously, but 3 sounds are normally used for EFFECT. If EFFECT is not being used, Score??5~Score??10 may be used.)

(Part 1) Inputting note data

Ex:

DB	\$18	\$88	\$18
(note length) (volume & tie specification) (Voice No. of PCMA)			

The three hexadecimal values is a set.

Input method for (note length) and (volume & tie specification) are the same for FM sound note input method. After these two values, please enter the Voice Number of the instrumental sound that is to be used from “VO_PCMA.SRC.” Also, the range of (volume & tie specification) is \$80~\$9F with \$9F being the maximum value.

(Part 2) Inputting total level of each Score (PCMA sound)

Ex:

DB	\$80	\$95
(volume specification code) (PCMA total level of each Score)		

The two numerical value makes a set for input.

(volume specification code) Please enter “\$80” in front of (PCMA total level of each Score).

(PCMA total level of each Score) The range is 80~9F, larger the value larger the volume.

These values must be entered, but may be modified as needed.

⑥ --- Input method for PCMB sound note

1 Score that is inside the box is the note data for PCMB sound.

(Part 1) Inputting PCMB sound note data

The method is the same as the input method for FM sound note data entry.

(Part 2) Inputting method for Score total level, total octave, and key.

Ex:

	①	②	③
DB	\$80,	\$FF,	\$80
	④	⑤	
DB	\$46,	\$40	

① --- Volume specification code. Please enter the value before ②, because the set consists of ① and ②.

② --- The total level of PCMB part is specified. The range is \$80~\$FF, larger

the value, larger the volume.

- ③ --- Total octave is defined with \$80 as the base (lowest) value. With the addition of even values (of \$02, \$04, \$06,\$08, \$0A, \$0C, \$0E) the octaves will go up by one octave at a time. The maximum value is \$8F.
- ④ --- Code which defines the tone. Please enter the value before ⑤, because the set consists of ④ and ⑤.
- ⑤ --- Please enter “TBL_OKEY_PCMB” or “TBL_PCMB.SRC” or code from “TBL_MKEY_PCMB.”
(For example, looking at the file example “Macro Definition “TBL_OKEY_PCMB” from section “File Name: “TBL_PCMB.SRC,” inputting “\$40” will designate the instrumental sound “VOICE 18” in file “VO_PCMB.SRC.” Inputting “\$80” with “TBL_MKEY_PCMB,” the series of instrumental sounds in line “PCMB KEY80” will be outputted.)

Please enter the values listed above as a set before PCMB part notes.

Other commands that are necessary for creating scores will be discussed below.

1. \$40 (Music stop code)

Please enter after each score. (Before entering this code, please put in a rest “\$01, \$40, \$00.”)

2. \$42 (Return first part code)

Please enter this code to loop a MUSIC.SRC piece after each score.

3. \$44 (LR switch code)

Defines the direction (L channel, R channel, and LR channel) of the output for each score. Please enter the following codes for each directional choice (These may be defined anywhere inside the Score.)

L channel output
DB \$44 \$02
R channel output
DB \$44 \$01
LR channel output
DB \$44 \$03

4. \$47 (Next music code)

Please enter this code to play one MUSIC.SRC after another. (Ex: after the introduction MUSIC.SRC.)

Ex:

Playing HM21.SRC and HM22.SRC.

After each score of HM21.SCR please enter the following.

\$47, (22 of HM22.SRC)

5. \$3F (Fade out code)

Defines fade out. Please enter the following inside 1 Score of choice: \$3F, \$?? (fade out speed). The range of fade out speed is 0~FF, and larger the value faster it fades out.

6. \$33 (Tempo change code)

Using this code allows change of tempo.

\$33 \$?? (??

* Special Commands for Musical Score Creation

1. “\$31” or “\$32” (repeat code)
“\$34” or “\$38” (repeat end code)

Allows repeat of groups of notes in each Score.

Ex:

```
①   ③  
$31, $10  
$??, $??, $??, $??, $??, $??, ~  
----- note data ----- note data  
  
②  
$34
```

- ① --- This is the repeat code. Please input this code before the group of note data you want repeated.
- ② --- This is the repeat end code. Please enter this code after the group of note data you want repeated.
- ③ --- Please enter the number of times you would want this loop to be repeated. The maximum value is 255 times. Please enter this number in hexadecimal. (i.e. 0~FF).

In the above example, the group of note data are repeated 16 times.

Ex 2:

```
$32, $02  
$??, $??, $??, $??, $??, $??, ~  
----- note data ----- note data  
  
$31, $10  
$??, $??, $??, $??, $??, $??, ~  
----- note data ----- note data  
  
$34  
$38
```

Using \$32 (repeat code) and \$38 (repeat end code) allows double repeat.

2. Chorus command (\$B0~\$BF)

Please choose 2 of the 4 Scores of FM sound to be outputted in unison. Entering the chorus command in the beginning of one of the 2 Scores will create the chorus effect. The range is B0~BF. “\$B0” designates no chorus, and larger the value the larger the effect.

3. B_JUMP command

If 1 pattern of phrase is to be used in numerous places, B_JUMP command should be used. This will save input time and storage space.

Ex:

① DB B_??_?? \$??, \$??, \$??, \$??, \$??, \$??, ~
note data note data

② DB B_RET

Before the phrase desired, please enter ①; and ② after. Please enter the chosen letters from the alphabet of numerical values in “?” of ①.

Please enter the following command where ever the desired phrase is to be outputted.

③ DB B_JMP
DW B_??_??

The characters in “??” must correspond with the characters in ①. The phrase may be entered as many times as needed inside MUSIC.SRC. Also, by changing the characters “??,” numerous B_JMP commands may be used.

FILE: "VO_FM.SRC"

FM tones to be used in MUSIC.SRC should be created by entering numerical values in this file. There are numerous publications dealing with FM sound equations, please refer to these books.

Ex:

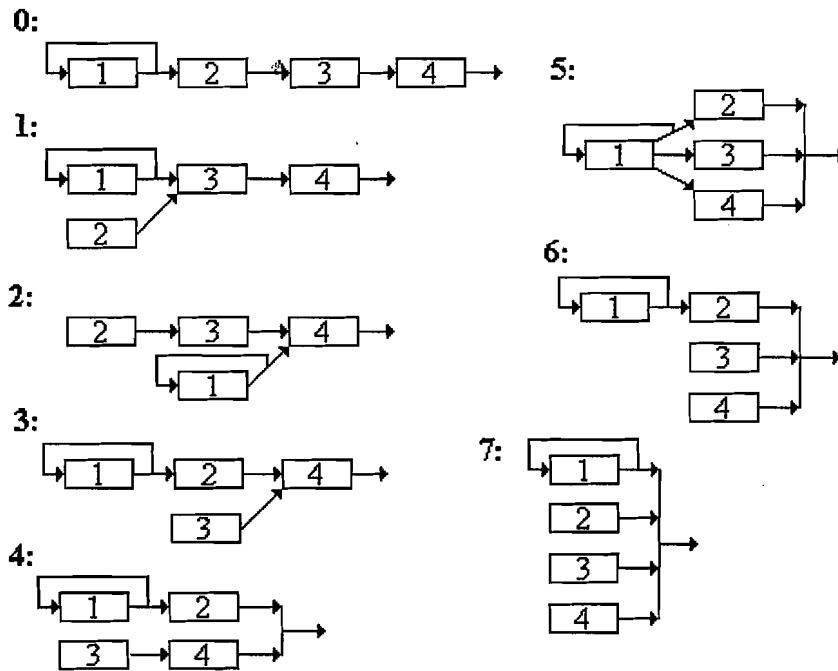
VOICE_No8A①

DEFB	7				→ ②
DEFB	5				→ ③
DEFB	3	-3	-3	3	→ ④ ⑤ ⑥ ⑦
DEFB	0	1	1	1	→ ⑧ ⑨ ⑩ ⑪
DEFB	27	0	0	0	→ ⑫ ⑬ ⑭ ⑮
DEFB	3	2	3	3	→ ⑯ ⑰ ⑱ ⑲
DEFB	16	16	16	16	→ ⑳ ㉑ ㉒ ㉓
DEFB	3	6	7	7	→ ㉔ ㉕ ㉖ ㉗
DEFB	0	0	0	0	→ ㉘ ㉙ ㉚ ㉛
DEFB	14	15	15	15	→ ㉜ ㉝ ㉞ ㉟
DEFB	5	5	7	7	→ ㉞ ㉟ ㉟ ㉟
DEFB	0	0	0	0	→ ㉛ ㉛ ㉛ ㉛
DEFB	0				→ ㉔
DEFB	0				→ ㉕
DEFB	0				→ ㉖
DEFB	00				→ ㉗
DEFB	0				→ ㉘
DEFB	0				→ ㉙
DEFB	0				→ ㉚
DEFB	00000000B				→ ㉛
DEFB	00000000B				→ ㉛

① --- This is the FM Voice Number for the set of numerical values. Please enter this number to call the sound for output in MUSIC.SRC. The values start from 80 and ends with FF, i.e. 127 tone entry capacity.

② --- Defines Feed back. Value ranges from 0~7, larger the value, stronger the effect.

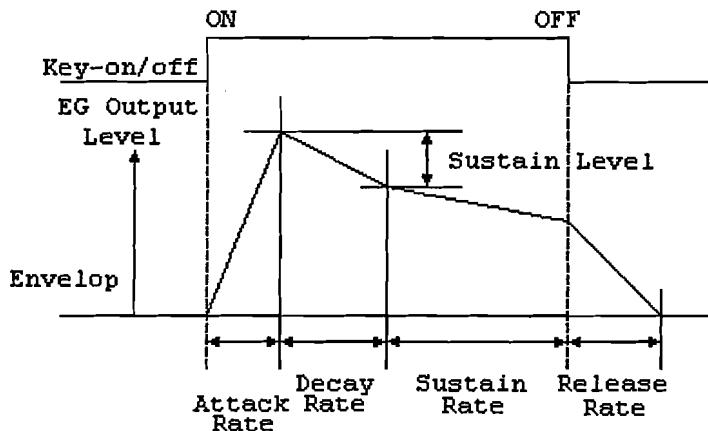
③ --- Defines the algorithm. Please choose from below; with the numbers shown below.



The following 4 values correspond with operators 1, 2, 3, and 4 respectively.

- ④⑤⑥⑦**--- Defines the de-tune value of each operator. The range is from -4~+3.
- ⑧⑨⑩⑪**--- Defines the multiple (frequency rate) of each operator. The range is 0~15, larger the value, the stronger the effect.
- ⑫⑬⑭⑮**--- Defines the output level of each operator. The range is 0~127, 0 having the highest amplitude value.
- ⑯⑰⑱⑲**--- Defines the key scale rate of each operator. The range is 0~3, larger the value, the larger the correction.
- ㉐㉑㉒㉓**--- Defines the attack rate of each operator. The range is 0~31, larger the value, faster the rate.
- ㉔㉕㉖㉗**--- Defines the decay rate of each operator. The range is 0~31, larger the value, faster the rate.
- ㉘㉙㉚㉛**--- Defines the sustain rate of each operator. The range is 0~31, larger the value, faster the rate.
- ㉒㉓㉔㉕**--- Defines the sustain level of each operator. The range is 0~31, larger the value, lower the level.
- ㉖㉗㉘㉙**--- Defines the release rate. The range is 0~15, larger the value shorter the release time.

(Reference Envelope Generator)



④⑩⑪⑫⑬--- Defines SSG type Envelop (assigns envelop created in SSG sound source to each operator). The shape and value relationship is as shown below. (Entering a value 0~7 designates OFF.)

No.	Envelope Shapes
8	
9	
10	
11	
12	
13	
14	
15	

SSG Type Envelop

⑭--- Defines the wave form of LFO (wave shape of low frequency output created by LFO). The value and shape relationship are as follows.

0 : saw tooth wave

1 : square wave

2 : triangle wave

3 : sample & hold (random wave)

⑮--- Defines Sync Delay (timing of LFO after key-on). The range is 0~FF, with 0 designating Sync Delay OFF.

- ⑥---** Defines the LFO Speed. The range is 0~7FFF.
- ⑦---** Defines PMD (Pulse Modulation effect defined). The range is 0~FF, larger the value, more intense the effect.
- ⑧---** Defines PMS (Pulse Modulation general [over all] effect defined). The range is 0~FF, larger the value, more intense the effect.
- ⑨---** Defines Int Count (LFO general [roughly] timing defined. The range is 0~FF, when LFO is used, please set the value to 1 (normally).
- ⑩---** Is not in use. (Please set to 0.)
- ⑪---** PMS-AMS1  Parameter which defines AMS.PMS of LFO from the hardware. (Above LFO is from the software.) Lower 4 bits is the parameter for AMS with range 0~3. Upper 4 bits is the parameter for PMS with range 0~7. (* Please change to binary value when inputting.)
- ⑫---** H-LFO-Switch F LFO switch from hardware. 0th bit is for AM of operator 1. 1st bit for AM of operator 2. 2nd bit for AM of operator 3. 3rd bit for AM of operator 4. When each of the bit value 1 equals ON, and value 0 equals OFF.

* Creating File “S·S·G.SRC”

Please do the following before entering data into the file “S·S·G.SRC.”

The file “CONFIG_S.SRC” (the initial state of “S·S·G.SRC” file) is shown below. Please copy file “CONFIG_S.SRC” with another name. For example, if it is copied as “SM61.SRC,” change all the area boxed to value with “61.”

```
; ***** BEGIN TO FILE HEADER *****
;
; SMUSIC[60] EQU      $
;
; ***** SSG priority *****
;
;     DB      [01]  ↵ ①
;
; ***** DATA ENTRY POINT ADDRESS TABLE *****
;
; DW      SCORE[60]S_1      ; PART 1
; DW      SCORE[60]S_2      ; PART 2
; DW      SCORE[60]S_3      ; PART 3
; DW      SCORE[60]S_4      ; PART 4 ; noise channel.
;
; ***** OPEN / CLOSE FLAG TABLE *****
;
;          OPEN (01)    CLOSE (00)
;
;     DB      $00           ; Part 1
;     DB      $00           ; Part 2
;     DB      $00           ; Part 3
;     DB      $00           ; Part 4 Noise mixed
;
; ***** Noise mixed sw *****
;
;     DB      [00000000B]  ↵ ③
;
; SCORE[60]S_1
;     DB      $55
; SCORE[60]S_2
;     DB      $55
; SCORE[60]S_3
;     DB      $55
; SCORE[60]S_N
;     DB      $55
;
;             ↵ ④
;
;             ↵ ⑤
```

① --- The priority order of that S'SG.SRC is defined. The range is 0~FF, smaller the value, higher the priority.

② --- Switch (to output or not) to OPEN or CLOSE each SCORE of ④ and ⑤.

③ --- A switch to define the SCORE of output destination of the NOISE part. Please enter the data for the NOISE part in SCORE??_N. This SCORE can not be outputted on its own. This SCORE must be mixed with one of the following SCORE??S_1~SCORE??S_3.

Ex:

DB	00000000B	
1	2	3
0	1	2

If noise to be mixed with SCORE??S_1, change the value of “0” at ① to “1.”

If noise to be mixed with SCORE??S_2, change the value of “0” at ② to “1.”

If noise to be mixed with SCORE??S_3, change the value of “0” at ③ to “1.”

(NOISE part not used)

Please do the following:

(1) Please enter a rest in SCORE??S_N with the score length of one of the SCORE??S_1~SCORE??S_3.

(2) Please turn “ON” (“\$01”) the appropriate SCORE in area ②.

(3) Please turn “OFF” all of the switches which define the SCORE of output destination of the NOISE part (values of “0”).

④ --- This is the music data for S'SG sounds. Input method is the same as mentioned for FM sound of MUSIC.SRC (please refer to earlier sections). The set is shown below.

DB \$(length of score), \$(volume & tie), \$(octave & key)

⑤ --- Please enter the NOISE part into this SCORE. Input method for length of score and volume & tie are the same as above (please refer to earlier sections). Please enter the following value for noise frequency, 0-1F. Smaller the value, lower the frequency and larger the value, higher the frequency. The set is shown below.

DB \$(length of score), \$(volume & tie), \$(noise frequency)

(Other Commands Used For SSG.SRC)

1. Envelop change

The use of 10 pre-set envelop may be used on SSG sound of SCORE??S_1~3.

Ex:

DW \$53, \$(Envelop Shape), \$(Envelop Speed)

(These three values consist a set, and should be entered in front of the note data of SCORE??S_1~3.)

(Envelop Shapes)

Please enter the value shown in the table below for the corresponding envelop shape.

No.	Envelope Shapes
1-3	
4-7	
8	
9	
A	
B	
C	
D	
E	
F	

(Envelop Speed)

Defines the envelop speed. The range is from 0~FFFF, smaller the value, the faster the speed.

2. \$55 (SSG Stop Code)

Please enter at the end of each SCORE.

3. Repeat Command

The command that was explained in the MUSIC.SRC section may be used as well in SSG.SRC. Please refer to the earlier sections for instructions.

File Name: "OA??.SRC" and "OB??.SRC"

If two or more sound effect is ADPCM sounds, using OA??.SRC; or if they are ADPCMB sounds, using OB??.SRC will allow output of one following another with one code. (Ex: If there are two sound effects entered, such as "click" and "boom," in "TBL_KEY_PCMA" of "VO_PCMA.SRC;" using "OA??.SRC allows use of one code to output the sound set of "click boom.")

File: "OAF0.SRC"

```
;;
;           ***** BEGIN TO FILE HEADER *****
;
;           AMUSIC[F0]    (A)      EQU      $
;
;           ***** DATA ENTRY PRIORITY *****
;
;           DB      [\$01]    (①)
;
;           ***** DATA ENTRY POINT ADDRESS TABLE *****
;
;           DW      SCOREA[F0]    (A)
;
;           ***** OPEN / CLOSE FLAG TABLE *****
;
;           OPEN (01)      CLOSE (00)
;
;           DB      [\$01]    (②)
;
;           ***** PART 1 DATA AREA *****
;
;           SCOREA[F0]    (A)
;
;           DB      \$80,\$9F      (③)
;
;           DB      \$30,\$88,\$23
;           DB      \$18,\$88,\$1A      (④)
;
;           DB      \$5A,\$00      (⑤)
```

- ① --- Defines the priority of the sound effects entered (defined below). Smaller the value, higher the priority.
- ② --- Flag table for sound effects entered. "\$01" designates output, and "\$00" designates no output.

- ③ --- Defines the volume level of the outputted sound effects. “\$80” is the volume level designation, and the raising the value, “9F” above, defines the raise of volume level. The range is \$80~\$9F.
- ④ --- The three bytes defines a set. The first byte defines the length of output. The output length is the same as the each note length of tempo “\$B8” of “MUSIC.SRC.” (Ex: “\$30,\$??,\$??” with tempo “\$B8” is the same as a half note.) The range is \$00~\$30. If the need is to have a longer length, please use the second byte as a tie as it was done in “MUSIC.SRC.”

The second byte designates the velocity and tie. Definition and inputting method is the same as PCMA part in “MUSIC.SRC.” Please refer to earlier sections.

The third byte defines the VOICE NUMBER (numerical value entered in “TBL_KEY_PCM?” of “VOPCMA.SRC”) of the sound effect.

- ⑤ --- This is the STOP CODE for “OA?.SRC.” Please enter this code at the end of these files.

Input Method for “OA?.SRC”

1. Copy the file and re-input the numerical value in (A), shown in the previous page, to correspond with the file name. (If the file name is “OAF0.SRC,” change the value in (A) to “F0.”)
2. Please remove “;0” in front of “AMUSICF0” of “TBL_PCMA_EFFECT2” in “VO_PCMA.SRC” as is shown below:

TBL_PCMA_EFFECT2	;OA?.SRC TABLE \$F0~FF
DEFW	AMUSICF0
DEFW	0; AMUSICF1
DEFW	0; AMUSICF2
DEFW	0; AMUSICF3

3. Including “SDATA5.DAT.”

=====	Play MML data	=====	
=====	INCLUDE	TBL_MU.SRC ;MUSIC TBL	=====
-----<	PCMA OA FILES >	-----	
-----<	INCLUDE	OAFO.SRC	-----

4. Please enter the file name numerical value in a location in “TBL_EFFECT_PCMA” of “VO_HOST.SRC.” The value in the location will be the code. (Ex: If the file is “OAF0.SRC,” input “F0” into the dotted box. Then the code for “OAF0.SRC” would be “E0.”) Also, please enter “3” in the code area for “TBL_MCODE.” (Please enter “3” in the boxed area of TBL_MCODE for “E0.”)

***** REQUEST CODE FLAG TABLE *****

; 0 : NO ENTRY
 ; 1 : ENTRY CODE

0 1 2 3 4 5 6 7 8 9 ABCDEF

TBL_MCODE	EQU	\$	
DB		0,1,1,1,1,1,1,1,1,1,0,0,0,1,1	; 00
DB		1,1,0,0,1,1,0,0,1,1,1,1,0,0,0	; 10
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; 20
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; 30
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; 40
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; 50
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; 60
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; 70
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; 80
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; 90
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; A0
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; B0
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; C0
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; D0
DB		3,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; E0
DB		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	; F0

0 1 2 3 4 5 6 7 8 9 ABCDEF

;	NO ENTRY	= 0
;	SYSTEM CODE	= 1
;	MUSIC ENTRY	= 2
;	EFFECT PCMA ENTRY	= 3
;	EFFECT PCMB ENTRY	= 4
;	EFFECT SSG ENTRY	= 5

TBL_EFFECT_PCMB	;	HOST REQUEST \$B0~\$BF	;	type HOST request
;	0 1 2 3 4 5 6 7 8 9 A B C D E F			
DB	\$00,\$00,\$00,\$00, \$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00,\$00, \$00			;

```

DB    $00,$00,$00,$00, $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00 ; 9
DB    $00,$00,$00,$00, $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00 ; A
DB    $00,$00,$00,$00, $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00 ; B
;
TBL_EFFECT_PCMA      ;HOST REQUEST $C0~$FF      ; type HOST request
;      0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
DB    $00,$00,$00,$00, $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00 ; C
DB    $00,$00,$00,$00, $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00 ; D
DB    $00,$00,$00,$00, $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00 ; E
DB    $00,$00,$00,$00, $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00 ; F
;
COIN_CODE
        DB          $7F

```

After doing the set up shown in 1~4, sending “E0” code will output the sound entered in “OAF0.SRC.”

Important Information on “OA???.SRC” and “OB???.SRC”

- * Input into “OB???.SRC” is similar to “OA???.SRC.” Please refer to information on “OA???.SRC.” There is however, one difference. In ③ mentioned in the previous page, the values are different with “\$80,\$FF,\$80,\$46,\$00.” “\$FF” is the volume with range of \$00~\$FF. Note that \$FF is the maximum volume.
- * ADPCM sound and ADPCMB sound may not be combined.
- * Maximum of 15, “OAF0.SRC”~“OAFE.SRC” may be created for “OA???.SRC.” Maximum of 15, “OBF0.SRC”~“OBFE.SRC” may be created for “OB???.SRC.”
- * Please begin “OA???.SRC” with “OAF0.SRC.” Once “OAF0.SRC” has been entered, “VOICE0A~VOICEFA” of “TBL_KEY_PCMA” in “VO_PCMA.SRC” can not be used.

CREATING M1 ROM

- (1) Please use the assembler on “SDATA5.DAT” to compile.
- (2) Please link the above output machine language with “SOUND2.COM” (this is the sound driver file). (Data address starts at 2E00H.)
- (3) The “SOUND2.COM” created by the above operation should be transferred to address 0 and “WINDOW.COM” (to prevent copying) to address 18000H with the ROM writer. If the “SOUND2.COM” exceeds address F7FFH, please create “BANK.COM” and transfer to address 10000H. (Creating “BANK.COM” will be covered later.)
- (4) After transfer, please input the lower two digits of check sum in address 14; and the numerical value 100H minus lower two digits of check sum (i.e. 100H - value in address 14) in address 15. These values should be inputted to the RAM of ROM writer. (These values are necessary for start up ROM and RAM check of NEO-GEO and MVS systems.)

After the operation in (1)~(4), please write into 1M ROM.

- * If the assembler is from “IWASAIKI,” operations (1)~(3) may be performed by the use of batch file “SB.BAT” to compile and to link. (For “BANK.COM,” please use “SB.BAT.”)
- * If the ROM writer being used is from “ANDO,” all transfers of COM file may be done with “ANDO.EXE.”

(Use)

Please type “ANDO (drive name) (file name).” After this input, instructions will appear on the monitor. Please follow its instructions and set up the ROM writer.

CREATING BANK.COM

(Explanation of BANK switch over)

There are “BANK0” and “BANK1” inside M1 ROM. The address of M1 ROM is “00000H~1FFFFH” with “BANK0” at “00000H~0FFFFH” and “BANK1” at “10000H~1FFFFH.”

(Area 0F800H~0FFFFH of “BANK0” is used as a work area)

Data inputted into "BANK1" are "MUSIC.SRC" related. The switch over from "BANK0" to "BANK1" is accomplished through the different MUSIC code sent and is done automatically in the sound program.

SOUND2.COM (data included in SOUND2.REL+SDATA5.DAT) is inputted from 00000H in “BANK0.” If SOUND2.COM exceeds 0F7FFH, “BANK1” is used. In this case, “BANK.COM” (MUSIC data included in SDATA6.DAT) will be inputted from 10000H. Method of creating BANK.COM is shown below.

(Creating BANK.COM)

1. For "TBL SBANK No" in "SDATA5.DAT"

Please enter “0” in the area that corresponds to the MUSIC code for the MUSIC data that is to be entered in “BANK0.” Please enter “1” in the area that corresponds to the MUSIC code for the MUSIC data that is to be entered in “BANK1.” If “BANK1” is not to be used, please enter “0.” Sound program refers to this table to switch from one BANK to the other.

Ex:

MUSIC data with MUSIC code “32” is to be outputted from “BANK1”

2. Please include the MUSIC data file name, that is to be entered in "BANK1," in "SDATA6.DAT." Next, define ORG to be at 8000H and please assemble (using /S) the segment using ASEG (absolute). (ORG and ASEG have been defined in "SDATA6.DAT." Also if the assembler is made by "IWASAKI" corporation, "SDB.BAT" will assemble the segment.)

Ex:

In the example below, MUSIC data "HM21.SRC" and "HM25.SRC" are included.

```
;-----  
;(DO NOT TOUCH HERE)  
ASEG  
ORG      8000H  
INCLUDE      EQU4.ASM  
;  
  
INCLUDE      HM21.SRC  
INCLUDE      HM25.SRC
```

3. After assembling "SDATA6.DAT," please check the top address of MUSIC data (to be inputted in BANK1) included in "SDATA6.DAT." Please input these addresses in "TBL_MU.SRC" directly. (If the assembler is made by "IWASAKI" corporation, checking "SDATA6.\$SY" after assembling will be sufficient. For example, if the MUSIC file is HM21.SRC, look for the letters "HM21.SRC" in "SDATA6.\$SY," the head address is the 4 digit numerical value left of "SDATA6.\$SY.")

Ex:

The top address of MUSIC data "HM21.SRC" and "HM25.SRC" included in "SDATA6.DAT" is directly inputted into "TBL_MU.SRC"

```
 ;*  
 ;*  
 ;***** MUSIC SCORE DATA TOP ADDRESS *****  
 ;*  
 ;*  
 TBL_MUSIC    EQU      $  
               DEFW     0;MUSIC20  
               DEFW     08000H  
               DEFW     0;MUSIC22  
               DEFW     0;MUSIC23  
               DEFW     0;MUSIC24  
               DEFW     0815FH  
               DEFW     0;MUSIC26
```

4. Please enter the value “2” in the code area in “TBL_MCODE” of “VO_HOST.SRC” which corresponds to the music number of the music source included in “SDATA6.SRC.”
5. Please link the following files, “BANK.REL” (to prevent copying) and “SDATA6.REL” created so far with the procedures mentioned earlier. “BANK.COM” should be created by this procedure. When linking, please define the data area starting from 8000h. (If the assembler is made by “IWASAKI” corporation, “SB.BAT” will do the linking operation.) Also, create “SOUND2.COM” (Following procedures 1 and 3, assemble, and link the files) and transfer “SOUND2.COM” starting from 0000H, “BANK.COM” starting from 10000H, and “WINDOW.COM” starting from 18000H to the 1M ROM.

Explanation of System Codes

System codes are codes necessary from the main CPU (68000) to control the sound program. For example, to stop all request of BGM and effect noise, or to fade the music out; the main CPU sends these codes. The code values range from 00 ~ 1F, and each of their functions are listed below.

SOUND PROGRAM SYSTEM CODE TABLE

code	FUNCTION	code	FUNCTION
00		10	ROM & RAM check
01	bank change	11	fade out stop
02	request of "eye-catch" BGM	12	
03	reset of sound program	13	
04	ALL DI	14	ADPCM(TBL_KEY_PCMA) sound effect stop
05	MUSIC DI	15	ADPCM(TBL_KEY_PCMA2) sound effect stop
06	EFFECT DI	16	
07	ALL EI	17	
08	MUSIC EI	18	ADPCM(TBL_KEY_PCMA) sound effect stop
09	EFFECT EI	19	ADPCM(TBL_KEY_PCMB) sound effect stop
0A	fade out	1A	ADPCM(TBL_KEY_PCMA2) sound effect stop
0B		1B	ADPCM(TBL_KEY_PCMB2) sound effect stop
0C		1C	ADPCM(TBL_KEY_PCMA3) sound effect stop
0D		1D	
0E	tempo change	1E	
0F	SSG STOP	1F	

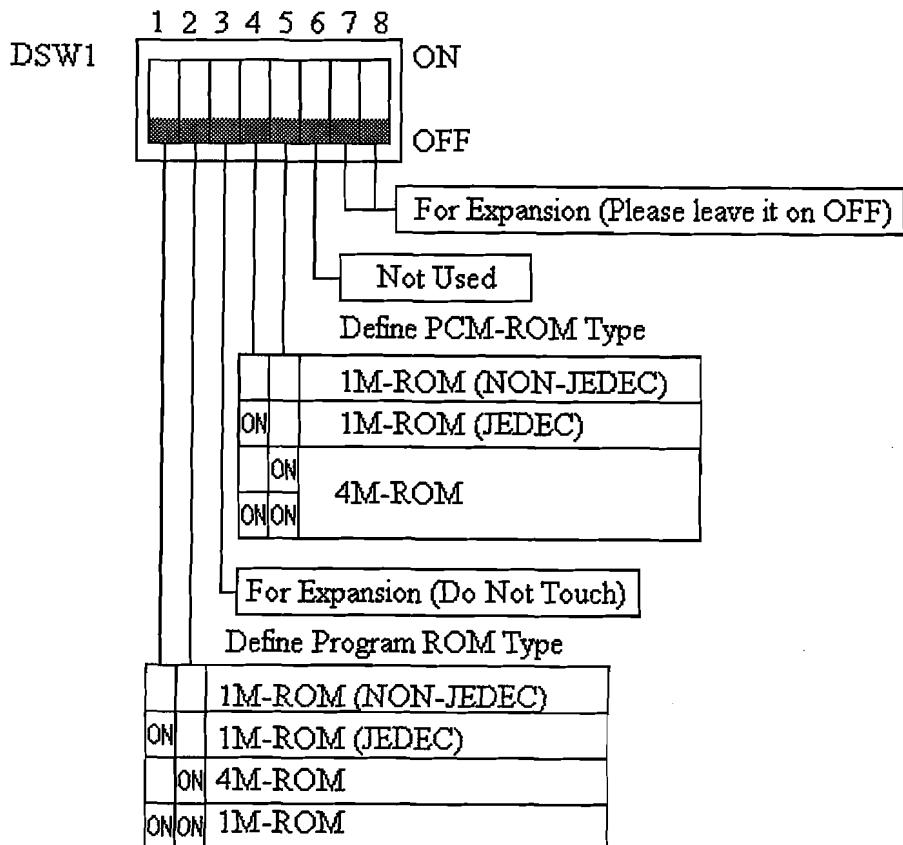
Explanation of Codes

code

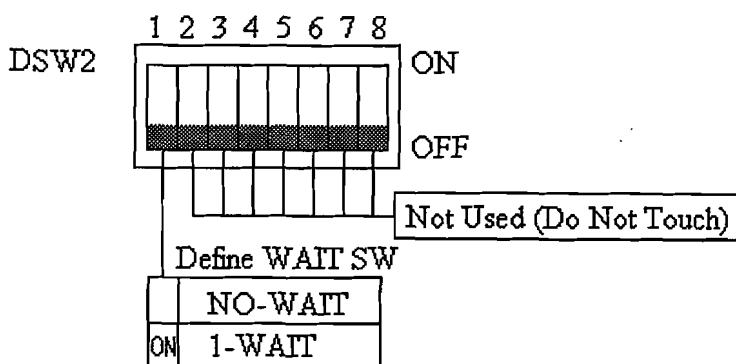
- 01 Code for the MVS. Please do not send this code.
- 02 This code will play the "eye-catch" BGM (5F).
- 03 Sound program reset. Please send this code in the beginning.
- 04 Will not accept any sound code (even if reset code is sent, this status will remain).
The coin sound (7F) is the only exception.
- 05 Will not accept any MUSIC codes (20 ~ 5F).
- 06 Will not accept any EFFECT codes (60 ~ FF).
- 07 Will accept all sound codes. This code must be sent after reset or ALL DI for any sound to be outputted.
- 08 Will accept all MUSIC codes (20 ~ 5F).

DEVELOPMENT BOARD DIP SWITCH QUICK REFERENCE TABLE

X0007-EPIC0 (Prog-Side) Dip-SW Quick Reference Table



007-EPIC (Prog-Side) Dip-SW Quick Reference Table



(NOTE)

1. A blank space inside the small boxes designate Dip-SW OFF.
2. Please do not touch the areas which are marked as "For Expansion." There is no guarantee that the system will run properly when they are turned ON.

- 09 Will accept all EFFECT codes (60 ~ FF).
- 0A Will fade out the BGM currently being played. Send the speed value, 00 ~ FF, after this code. Larger the value the faster the fade out speed.
- 0E Will change the tempo of the BGM currently being played. Please send the tempo data after this code (Tempo data is the same as the MUSIC file data).
- 0F Will stop the SSG sound effect (It will only stop the output).
- 10 Please do not send this code.
- 11 Sending this code during fade out will stop the fade out.
- 14 Sending voice number entered in “TBL_KEY_PCMA” after this code will stop that sound effect.
- 15 Sending voice number entered in “TBL_KEY_PCMA2” after this code will stop that sound effect.
- 18 Sending voice number entered in “TBL_KEY_PCMA” after this code will start output of that sound effect.
- 19 Sending voice number entered in “TBL_KEY_PCMB” after this code will start output of that sound effect.
- 1A Sending voice number entered in “TBL_KEY_PCMA2” after this code will start output of that sound effect.
- 1B Sending voice number entered in “TBL_KEY_PCMB2” after this code will start output of that sound effect.
- 1C Sending voice number entered in “TBL_KEY_PCMA3” after this code will start output of that sound effect.

NOTE:

00H ~ 1FH table of “TBL_MCODE” in “VO_HOST.SRC” are part of the sound program system code entry. Please enter the value “1” inside the corresponding area of “TBL_MCODE” with respect with the sound program system code entry.

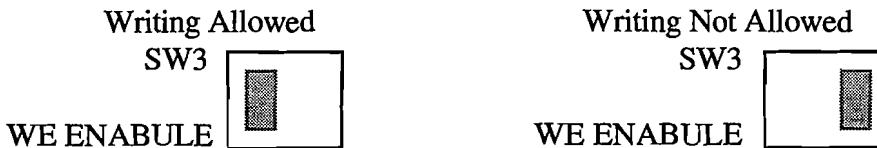
If sending two or more codes consecutively, please wait 64 ms or more for reset code, or 32 ms or more for other codes (including codes other than system codes).

About the PCM-ROM

For the PCM-ROM, you may use three types of ROM with different capacity and pin configuration. The ROMs that can be used in this board is: 1M JEDEC type ROM, 1M NON-JEDEC type ROM, 4M-ROM. There are three choices, but using a different type ROM among the JEDEC types result in the program not working properly and may result in loss of the very expensive ROMs.

About the Program ROM

1. Just like the PCM-ROM, three ROM types mentioned above may be used. In addition 1M-RAM is also available. When the program area is used as a RAM, you can not write directly into this area. When loading the program into RAM, please start from address 200000H. RAM address 0H has a correspondence with address 200000H, except you can not write into this area.
2. After changing the program area to RAM, there is a danger of running the program and destroying the area. To prevent this kind of accident, there is a memory write protect SW on the board. There should be a switch with the marking "SW3." Pushing the switch toward the right side will stop any writing requests from the main board. (This switch is valid for memory cards as well.)



3. When using a 4M-ROM for the program ROM, please insert ROM into socket P1 and P2. Inserting into other sockets will not work.

About the Memory Card

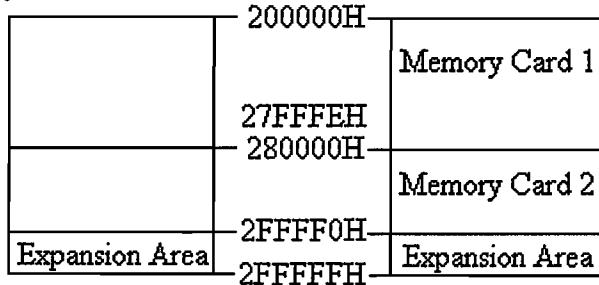
1. On this board, the memory card may be used as a program RAM. The memory card that may be used are JEIDA format. Also you may insert two 4Mbit memory card.
2. This board does not support back up, but by using the memory card, it offers the same function. When using the memory card, please do not insert anything into program sockets (P1 ~ P8).
3. Inserting the memory card will light the CD1EN or CD2EN LED. If the LED is not lit, please make sure the memory card has been inserted correctly. Also, if the write protect SW of the memory card is ON, CD1WP or CD2WP LED will go off.

NOTE:

1. Although in RAM mode, addresses 20000H and later addresses are used for program load, please do not use addresses 2FFFF0H ~ 2FFFFFFH since they are reserved for expansion.
2. Leaving the memory card inserted after the power has been turned off results in faster consumption of the back up battery of the memory card. And for protection of the memory card, please take the card out and into a protective casing when turning the power off.

X0007-EPIC0 Memory Map

1. When Using Memory Cards



- * The above memory map corresponds to when memory is being written in.
During read, start address may be set at 0H.

2. When Using RAM

D15	D8 D7	D0	
P2	P1		200000H
P4	P3		240000H
P6	P5		280000H
P8	P7		2C0000H
EVEN	ODD		2FFFFFFH

- * The above memory map corresponds to when memory is being written in.
During read, start address may be set at 0H.