Super Mega Drive Manual

SUPER MEGA DRIVE MANUAL

SEGA ENTERPRISES LTD.

CONFIDENTIAL #161 PROPERTY OF SEGA

Development MEGA DRIVE Specifications

Contents page

1. Power supply

- Input power line AC 85-132V (0.3A fuse)
  A 5V-3A switching regulator power supply with switches is installed.

- Power consumption rate: 5V, 2.3A in operation

2. Memory

· 68000 area

000000-0FFFFF	SRAM(back_up)	Game ROM emulating	Eight 1MbitRAMs
200000-23FFFF	SRAM(back_up)	BACK_UP RAM	Two 1MbitRAMs
800000-87FFFF	EPROM	SHADOW_ROM · Images are generated from 0 to 7FFFF if level 7 is set when the power is turned on.	Two 2MbitROMs
FC0000-FFFFFF	SRAM	Game WORK_RAM · FC0000 to FEFFFF have been added for development. · FF0000 to FFFFFF are used for the GENESIS.	Two 1MbitRAMs
A88000-A88FFF	EEPROM	MODEM development · The lower byte can only be used.	One 16KbitRAM
A87800-A87801	SRAM	HISTORY · A87800 can be used as serial memory (32Kword) if level 7 is set when the power is turned on. · It is used as history with 8192 steps if any level other than 7 is set when the power is turned on.	Two 256KbitRAMs

· Z80 area

0000-1FFF	SRAM	PROGRAM · Switching is performed in the 4-bank (0000-1FFF each) Z80 I/O area. DIP_SW6 is used to turn the switching on/off.	One 256KbitRAM

· VDP area

00000-1FFFF	DRAM	VIDEO_RAM · The 128K mode (bit7=1) is set with VDP reg#2.	Four 256KbitDRAMs

3. Connector
3-a Connector

- 171-5870 board (upper)
  - ◎ = MEGADRIVE standard
  - ☆ = Option

◎Cartrudge pins (Mitsumi 475-32-145)	CN1	Game cartridge 32 pins by two rows
◎FDD pins (60pin,flat)	CN2	FDD connection, substitution of the 60-pin edge pins. 30 pins by two rows
◎JOYPAD pins	CN4,5	
◎MODEM pins	CN6	
◎VIDEO pins	CN9	
◎PHONE pins	CN11	
☆68000 pins (64 pins, flat)	CN7,8	Output from the 68000 pins: the output can be raken by setting ALL_GND on one side. ZAX's 68000 ICE can be connected with a flat cable. 32 pins by two rows....two sets
☆ZAX pins (JST 5 pins =5 pins by one row)	CN16	External break for the 68000 ICE. These are connected to the external break pin of the ZAX's 68000 ICE. The EVI pin of the ICE is set to the low-voltage status when the address checker detects a break signal.
☆RGB pin (Dsub, 15 pin)	CN10	Video signal output of an analog RGB or a personal computer.
☆Palletpins (40 pins, flat)	CN15	These are connected to a color pakket output IC or a color pallet IC.

3-b. Connector

- 171-5871 board (lower)
  - ◎ = MEGA DRIVE standard
  - ☆ = Option

☆External I/O (64 pins, flat)	CN18	External Z80
☆Parallel I/O (34, pins flat)	CN19	8255 (8 bits by 3)
☆Printer (RC, 14 pins)	CN20	
☆Printer output (RC, 36 pins)	CN21	
☆RS232C (Dsub, 25 pins)	CN22	The rate can be in 300 to 19200 baud.
☆MIDI input (DIN, 9 pins)	CN23	
☆MIDI output (DIN, 9 pins)	CN24	

4. Option IC socket

- 171-5870 board (upper)

68-pin PLCC (68 pins with PLCC socket)	IC11	For Altera's EPM5128 ·PAL for ADDRESS_CHECKER
68-pin PLCC (68 pins with PLCC socket)	IC18	For Altera's EPM5128 ·PAL for HARD_BREAK

*Signal lines connected to EPM5128

To 68000	CLK,RESET,DTACK BG,BGACK AS,UDS,LDS,RW FC0-FC2 A1-A23 D0-D15
To Z80	RES,BUSAK
To others	Pin 24 from the board, and pin 18 to the board. Not connected: pins 15, 17, 19, 21, 22 and 23.

5. Timer

- 171-5871 (lower)

RTC62421	IC39	BACK_UP is implemented. Year, month, day, weekly day, hour, minute and second.

6. Switch

- SW1(PUSH_SW) The 68000 is reset.
- SW2(PUSH_SW) The development board is reset.
- SW3(PUSH_SW) Level 7 is set for the 68000.

· SW4(DIP_SW)	Country	Japan=OFF	Overseas=ON
· SW5(DIP_SW)	NTSC/PAL	NTSC=OFF	PAL=ON
· SW6(DIP_SW)	Z80BANK	NO=OFF	BANK=ON
· SW7(DIP_SW)	CPU	ICE=OFF	CPU=ON

7. Designing

a)The unit needs to be compatible with upgarded MEGA DRIVEs.

b)Eliminate errors in RAMs by using the 1-Mbyte SRAM.

c)The ICE is interrupted by a break signal from the address checker.

d)Software can be developed and debugged on the unit alone.
·A monitor break occurs.
·A hard break occurs.
·The address checker sends a break.
·The history function is implemented.
·Prigrams can be uploaded and downloaded.

e)It can be used as a graphics machine for the MEGA DRIVE.
·Video can be simultaneously displayed on a general TV and a monitor TV.
·All the 256 colors can be displayed by using pallet ICs.

f)The main unit alone can be used as an aging machine.
·The debug function is implemented.

g)The unit can be used as a sound source for sound development.
·Communication can be performed by using MIDI.
·MIDI_data can be displayed by using video output.

The special software needs to be developed to implement functions d), e), f) and g).

8. Board specifications

The unit can handle the MEGA DRIVE mode and the monitor mode. The monitor mode is set when the power is turned on.

a)In the MEGA DRIVE mode:
1)History is recorded.
2)When a break occurs, operation moves to the monitor mode.

b)In the monitor mode:
1)The MEGA DRIVE functions can be used.
2)History can be read and written.
3)The supplied I/O can be used.
4)Various modes can be set up.

8-a.In the MEGA DRIVE mode:

·Access to the address of the 68000 is recorded in history.
(bus access of the 68000 and bus request by the Z80)

·Operation moves to the monitor mode when the following monitor break functions are used (level 7 occurs; the mode is changed with INT&ACK7).

1)Timeout when DATACK is not returned.
2)Write protect when an attempt is made to write data in the ROM status.
3)Memory out when the quantity of data exceeds the memory size.
4)Break switch when PUSH_SW 'SW3' is pressed.

*1 5) Hard break when the assigned address is accessed.
*2 6) Address checker when the prohibited operations of the MEGA DRIVE are executed.

*1··· To execute this function, the address checker (EP5128) is required for IC11.
*2··· To execute this function, the address checker (EP5128) is required for IC12.

The PALs in *1 and *2 can be modified for or added to the other functins.

8-b. Functions available in the monitor mode

- ·The monitor ROM is selected when the power is turned on (or level 7 occurs).
- ·Level 5 is generated from communication through RS232C or MIDI.
- ·Level 3 is generated from the external I/O (external Z80).
- ·Level 1 is generated from a timer interrupt (RTC62421).
- ·Modification of mapping (mapping address of SRAM is changed).
- ·Modification of the memory size (the memory size of SRAM is in units of 2, 4, 6 or 8 Mbytes).
- ·Write protect (writing in SRAM is inhibited).
- ·History can be read and written (history is suspended).
- ·The MEGA DRIVE functions can be used.
- ·The functions of the supplied I/O can be used.
- ·Operation moves to the monitor mode.

9.Mapping in the monitor mode.

```
*******************
* $000000-$FFFFFF *
*******************
All area
```

$00000	
$10000	rom
$20000	
$20XXX	back_up
$40000	
$43fff	fdd_rom
$60000	
$63fff	fdd_ram
$80000	
$87fff	mon.ROM
$a0000	
$a13ff	i/o
$a8000	
$aafff	mon.I/O
$c0000	
$c0001	vdp
$fc000	
$fdfff	mon.RAM
$ff000	
	ram
$fffff	

```
*******************
* $A00000-$AFFFFF *
*******************
MON.I/O
```

$a0000	i/o
$a2000	
$a8000	EXP.I/O
$aa000	EXT.I/O
$ac000	
$b0000	

```
******************
* $A80000-A88FFF *
******************
EXP.I/O
```

$a80000	PalletIC
$a81000	Monitor
$a82000	mapctl
$a83000	adrcker
$a83800	breaker
$a84000	uPD7201
	clk.div
$a85000	RTC
	RTC_int
$a86000	8255_0
$a86800	8255_1
$a87000	memwp
$a87800	history
$a88000	EEPROM
$a88800	

10. Extended I/O mapping

10-a) Board 1 (171-5870), extended I/O

```
extcol          equ     $a80001         * External color pallet (CN15)
monclr          equ     $a81001         * Monitor mode clear (write)
brkmode         equ     $a81001         * Break mode data (read)
memctl          equ     $a82001         * memory map control
adrcker         equ     $a83000         *(WORD) ADDRESS checker (IC12)
breaker         equ     $a83800         *(WORD) HARD_WARE break (IC11)
```

10-b) Board 2 (171-5871), extended I/O

```
232c_dat        equ     $a84001         * μPD7201  232C_data
232c_cmd        equ     $a84003         * μPD7201  232C_command
midi_dat        equ     $a84005         * μPD7201  midi_data
midi_cmd        equ     $a84007         * μPD7201  midi_command

clkdiv          equ     $a84801         * 232C clock divider      (IC51)

rtc             equ     $a85001         * Timer IC     (RTC62421:IC39)

rtcint          equ     $a85801         * LEVEL1 enable(Timer IC occurs)

prtin           equ     $a86001         * 82C255_0 sentro in     (CN21)
prtout          equ     $a86003         * 82C255_0 sentro out    (CN20)
prtcnt          equ     $a86005         * 82C255_0 sentro cnt    (CN20,21)
prtmod          equ     $a86007         * 82C255_0 8255_0 mode

expA            equ     $a86801         * 82C255_1 parallel_A    (CN19)
expB            equ     $a86803         * 82C255_1 parallel_B (CN19)
expC            equ     $a86805         * 82C255_1 parallel_C  (CN19)
expmod          equ     $a86807         * 82C255_1 8255_1 mode

memwp           equ     $a87000         * memory write protection

hist            equ     $a87800         * cpu access history

EEPROM          equ     $a88000         * 2Kbytes EEPROM (IC15)

extio           equ     $aa0000         * 128Kbytes reserved(CN18)
```

11.Extended memory mapping (in the monitor mode)

11-a)Board 1 memory (171-5870)

```
extram          equ     $fc0000         * extend work_ram
                *       $feffff              (IC32,35)
```

11-b)Board 2 memory (171-5871)

```
monrom          equ     $800000         * Monitor ROM    (upperIC16)
                *       $87ffff                          (lowerIC17)
backram         equ     $880000         * BACKUP_RAM     (upperIC1)
                *       $8bffff                          (lowerIC6)
mainram         equ     $900000         * Cartridge emulating RAM
                *       $9fffff          (upIC15,4,3,2 lwIC10,9,8,7)
```

12.Extended I/O function (in the monitor mode)

a-1) Memory map control (memctl)

·Mapping can be changed (map address of SRAM can be changed).

〈〉 memory map control

```
$a82001          (write_only)
        7 6 5 4 3 2 1 0                     bit0=1 ctrg-)(fdd),fdd-)(ctrg)
        | | | | | | | +---------map_change bit0=0 ctrg-)ctrg ,fdd-) fdd
        | | | | | | |                [ctrg] and [fdd] is swapped (effective when bit 3= 0).
        | | | | | | |
        | | | | | +-+-----------memory map control
        | | | | |                    Mapping can be changed.
        | | | | |
        | | | | +---------------CARTRIDGE_pin disable (0)/enable(1)
        | | | |                      [ctrg] and [fdd] is swapped by means of cartridge pins
        | | | |                      (bit 0 is not valid).
        | | | |
        | | | +-----------------emulator mapping bit
        | | |                        Mapping for the the emulate mode is temporarilly
        | | |                        executed in the monitor mode.
        | | |
        | | +-------------------auto_vector mode(0),vector(1)
        | |                          The vector mode is changed to the auto_vector mode or
        | |                          vice versa.
        | |
        +-+---------------------vector page(vecter bank change)
                                     The vector_address is changed in the vector mode.
```

```
          BOOT
$800000 +-------+
        | rom   |
        |       |
$880000 +-------+
        | bram  |(back_up)
$8c0000 +-------+
        |       |
$900000 +-------+
        | ram   |(emu_ram)
        |       |
        |       |
        |       |
        |       |
$a00000 +-------+
```

cart	:	cartridge
fdd	:	floppy disk unit
BOOT	:	boot_rom,emulation_ram & back_up_ram
ram	:	emulation_ram(8Mbits)
bram	:	back_up_ram(2Mbits)
ex cart	:	cartridge slot [cart] line
work	:	work_ram (extnded)
rom	:	boot_rom

12.Extended I/O function (in the monitor mode)

a-2 memory map control (memctl)

<< memory map control >>

```
+-------------++---------------------------+---------------------------+
|  hard mode  ||          emulate          |          monitor          |
|             ||(if bit4=1 in monitor_mode)|                           |
+-------------++------+------+------+------+------+------+------+------+
|   bit 2,1   || 0  0 | 0  1 | 1  0 | 1  1 | 0  0 | 0  1 | 1  0 | 1  1 |
|  ($a82001)  ||      |      |      |      |      |      |      |      |
      $000000 +------+------+------+------+------+------+------+------+
              | boot | ram  |      |      | boot | boot | boot | ram  |
              |      +------+      |      |      |      |      +------+
              |      |      | cart | cart |      |      |      |      |
      $200000 +------+      |(fdd) |(fdd) +------+------+------+      |
              |      |      |      |      |      |      |      |      |
              |      |      |      |      |      |      |      |      |
              |      |      |      |      |      |      |      |      |
      $400000 +------+------+------+------+------+------+------+------+
              |      |      | ram  |      |      |      |      |      |
              |      | fdd  +------+ fdd  |      |      |      |      |
              |      |(cart)|      |(cart)|      |      |      |      |
              |      |      |      |      |      |      |      |      |
              |      |      |      |      |      |      |      |      |
      $800000 +------+------+------+------+------+------+------+------+
              | boot | boot | boot | ram  | boot | boot | boot | ram  |
              |      |      |      +------+      |      |      +------+
              |      |      |      |      |      |      |      |      |
      $a00000 +------+------+------+------+------+------+------+------+
              |      |      |      |      |      |      |      |      |
              |      |      |      |      |      |      |      |      |
              |      |      |      |      |      |      |      |      |
      $c00000 +------+------+------+------+------+------+------+------+
              |      |      |      |      |      |      |      |      |
              |      |      |      |      |      |      |      |      |
              |      |      |      |      |      |      |      |      |
      $e00000 |      |      |      |      |      |      |      |      |
              |      |      |      |      |      |      |      |      |
              |      |      |      |      +------+------+------+------+
      $fc0000 |      |      |      |      |      |      |      |      |
      $ff0000 |------+------+------+------+ work | work | work | work |
              +------+------+------+------+------+------+------+------+
```

12.Extended I/O function (in the monitor mode)

a-3) memory map control (memctl)

〈 map change (bit0) & crat_pin(cartridge.B_side pin32) 〉

bit 3	bit 0	cart_pin	$000000	$400000
0	0	*	cart	fdd
0	1	*	fdd	cart
1	*	0	cart	fdd
1	*	1	fdd	cart

〈 interrupt_mode & interrupt_address 〉

bit5=1

bit7	bit6	int1	int2	int3	int4	int5	int6	int7
0	0	$064	$068	$06c	$070	$074	$078	$07c
0	1	$0e4	$0e8	$0ec	$0f0	$0f4	$0f8	$0fc
1	0	$164	$168	$16c	$170	$174	$178	$17c
1	1	$1e4	$1e8	$1ec	$1f0	$1f4	$1f8	$1fc

12.Extended I/O function (in the monitor mode)

b-1) Write Protect (memwp)

·The memory size is changed (the size of the SRAM memory is in units of 2, 4, 6 or 8 Mbytes).

() memory write protect

$a87001 (write_only)

```
7 6 5 4 3 2 1 0
| | | | | | +-+----------0H=2M,1H=4M,2H=6M,3H=8M_bye
| | | | | |             (rom_emulate size)
| | | | | |
| | | | +-+-------------0H=0k,4H=64k,8H=256k,0cH=1024k
| | | |                 ( back_up memory size)
| | | |
| | | +-----------------write_enable rom_emulate   (for_break)
| | |                   Write_enable when bit 4 = 1 is set for the RAM during
| | |                   emulation.
| | |
| | +-------------------write_enable ram in BOOT
| |                     Write_enable when bit 5 = 1 is set for the image RAM in
| |                     the boot area.
| |
| +---------------------lower.write_enable back_up (for_break)
|                       Lower write_enable when bit 6 = 1 set for the back_up
|                       RAM during emulation.
|
+-----------------------upper.write_enable back_up (for_break)
                        Upper write_enable when bit 7 = 1 set for the back_up
                        RAM during emulation.
```

```
+--------------++-------------------------+-----------------------+
|  hard mode   ||         emulate         |        monitor        |
+--------------++-------------------------+-----------------------+
|   bit 1,0    || 1 1 | 1 0 | 0 1 | 0 0 | 1 1 | 1 0 | 0 1 | 0 0 |
|  ($a87001)   ||(8M) |(6M) |(4M) |(2M) |(8M) |(6M) |(4M) |(2M) |
         $000000+-----------------------+-----------------------+
                |  m  |  m  |  m  |  m  |  M  |  M  |  M  |  M  |
         $040000|-----|-----|-----|=====|-----|-----|-----|=====|
                |  m  |  m  |  m  |     |  M  |  M  |  M  |     |
         $080000|-----|-----|=====|     |-----|-----|=====|     |
                |  m  |  m  |     |     |  M  |  M  |     |     |
         $0c0000|-----|=====|     |     |-----|=====|     |     |
                |  m  |     |     |     |  M  |     |     |     |
         $100000|=====|     |     |     |=====|     |     |     |
                |  B  |  B  |  B  |  B  |  X  |  X  |  X  |  X  |
                |     |     |     |     |     |     |     |     |
                |     |     |     |     |     |     |     |     |
                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

B_area = access break (change_mode to monitor)
m_area = if bit4=0,then write break (change_mode to monitor)
M_area = write_enable on monitor·mode
X_area = don't care

12.Extended I/O function (in the monitor mode)

b-2) Write Protect (memwp)

· Write protect (write to the SRAM is inhibited).

```
+--------------++-----------------------+-----------------------+
|  hard mode   ||        emulate        |        monitor        |
+--------------++-----------------------+-----------------------+
|   bit 3,2    || 1 1 | 1 0 | 0 1 | 0 0 | 1 1 | 1 0 | 0 1 | 0 0 |
|  ($a87000)   ||(2M) |(512)|(128)|( 0K)|(2M) |(512)|(128)|( 0K)|
+--------------++-----------------------+-----------------------+
        $200000 +-----------------------+-----------------------+
                | h l | h l | h l |     | H L | H L | H L |     |
        $204000 |-----|-----|=====|     |-----|-----|=====|     |
                | h l | h l |     |     | H L | H L |     |     |
                |-----|-----|     |     |-----|-----|     |     |
                | h l | h l |     |     | H L | H L |     |     |
                |-----|-----|     |     |-----|-----|     |     |
                | h l | h l |     |     | H L | H L |     |     |
        $210000 |-----|=====|     |     |-----|=====|     |     |
                | h l |  B  |  B  |  B  | H L |  X  |  X  |  X  |
                |-----|     |     |     |-----|     |     |     |
                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                |-----|  B  |  B  |  B  |-----|  X  |  X  |  X  |
                | h l |     |     |     | H L |     |     |     |
        $240000 +=====|     |     |     |=====|     |     |     |
                |     |     |     |     |     |     |     |     |
                |     |     |     |     |     |     |     |     |
                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
B_area = access break                          (change_mode to monitor)
X_area = don't care
h_area = if bit7=0,then upper_write break      (change_mode to monitor)
l_area = if bit6=0,then lower_write break      (change_mode to monitor)
H_area = if bit5=1,then write_enable
L_area = if bit5=1,then write_enable
```

12.Extended I/O function (in the monitor mode)

c) Change to emulate (monitor)

·Operation moves to the emulator mode.

〈〉 monitor

```
$a81001        (write)
        * * * * * * * *---------change to EMULATE

$a81001 (read)
        7 6 5 4 3 2 1 0
        | | | | | | | +---------in bgack
        | | | | | | +-----------in monitor
        | | | | | +-------------by write protect
        | | | | +---------------by 232c_hunt
        | | | +-----------------by dtack time_out
        | | +-------------------by break_chip
        | +---------------------by address_checker
        +-----------------------by break_switch

                        * all_bit active low
```

12.Extended I/O function (in the monitor mode)

d) RS-232C & MIDI

· Level 5 is generated from communication with RS232C or MIDI.

〈〉 uPD7201A

		write	read
$a84001	ch-A data	232C receive_data	232C transmit_data
$a84003	ch-A control		
$a84005	ch-B data	MIDI_out data	MIDI_in data
$a84007	ch-B control		

〈〉 rs-232c baud rate clock generator

```
$a84801        (read/write)
        * * * * 3 2 1 0
                | +-+-+---------baud rate
                +---------------uPD7201A interrupt mode
                                0:!int5
                                1:!int7(nmi)
```

bit 2	bit 1	bit 0	7201 Tx/Rx_clock
0	0	0	4800
0	0	1	9600
0	1	0	19200
0	1	1	38400
1	0	0	76800
1	0	1	153600
1	1	0	307200
1	1	1	614400

*A 300 baud rate is set when $A84801 = 0 and the uPD7201A internally divides the frequency by 16.

12.Extended I/O function (in the monitor mode)

e-1) Address Checker

(emap) · When an attempt is made to access the address that the MEGA DRIVE cannot access, a break occurs.

(ereg) · When an attempt is made to set VDP data that the MEGA DRIVE cannot set, a break occurs.

(ebus) · When a bus request is issued from the Z80 immediately after I/O is accessed, a break occurs.

(timi) · If the work_ram was not accessed 2μsec before the last destination data is set by setting the DMA, a break occurs.

```
〈〉   address checker

$a83000          write
bit      f e d c b a 9 8 7 6 5 4 3 2 1 0
         * * * * * * * * * * * * * | | +-mapping check on(1)/off(0)
                                   | |   Check related to mapping becomes effective
                                   | |
                                   | +---vdp_cmd1 check on(1)/off(0)
                                   |     The first word that is set in VDP is checked.
                                   |
                                   +-----vdp_cmd2 check on(1)/off(0)
                                         The second word that is set in VDP is checked.

power_on        default
         * * * * * * * * * * * * * 1 1 1
                                           The address_checker remains valid
                                           when the power is turned on.
```

12.Extended I/O function (in the monitor mode)

e-2) Address Checker

```
                read
$a83000
bit      f e d c b a 9 8 7 6 5 4 3 2 1 0          error_status & error address (H)
         | | | | | | | | | | | | | | | +- a16
         | | | | | | | | | | | | | | +--- a17
         | | | | | | | | | | | | | +----- a18
         | | | | | | | | | | | | +------- a19
         | | | | | | | | | | | +--------- a20
         | | | | | | | | | | +----------- a21
         | | | | | | | | | +------------- a22
         | | | | | | | | +--------------- a23
         | | | | | | | |                    STATUS of MEGA DRIVE
         | | | | | | | +-------- rwmd       on write (1) /read (0)
         | | | | | | +---------- size       byte access(1) /no (0)
         | | | | | +------------ bga        in back (1) /no (0)
         | | | | +-------------- cmd2       next_vdp cmd is 2nd
         | | | |                    ERROR STATUS
         | | | +---------------- timi       error no_work access before DMA
         | | +------------------ ebus       error Z80 access after I/O ($a100xx)
         | +-------------------- ereg       error VDP_reg access
         +---------------------- emap       error mapping

$a83002
         a a a a a a a a a a a a a a a u
         1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 d          error address(L)
         5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 s

$a83004
         d d d d d d d d d d d d d d d d
         1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0          error data
         5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0

$a83006
                       z a a a a a a a a
                       r 2 2 2 2 1 1 1 1          last_program address(H)
                       e 3 2 1 0 9 8 7 6          Reset of zres = Z80
                       s
$a83008
         x x x x x x x x x x x x x x x x

$a8300a
         a a a a a a a a a a a a a a a u
         1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 d          last_program address(L)
         5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 s

$a8300c
         d d d d d d d d d d d d d d d d
         1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0          last_vdp data
         5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

12.Extended I/O function (in the monitor mode)

f) HARD_WARE Break

```
·If the address is set in the emulate_mode, then:
(wre=1,rde=1)   a break occurs, when this address is accessed;
(wre=1,rde=0)   a break occurs, when data is written in this address;
(wre=0,rde=1)   a break occurs, when data is read from this address;
                Three points ch.A,ch.B,ch.C can be set.
```

() hard ware breaker

```
power_on        default        Each channel is suspended when the power is turned on.

$a83800 * * * * * * 0 0 0 0 0 0 0 0 0 0         * break_CH.A disable
$a83802 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$a83804 * * * * * * 0 0 0 0 0 0 0 0 0 0         * break_CH.B disable
$a83806 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$a83808 * * * * * * 0 0 0 0 0 0 0 0 0 0         * break_CH.C disable
$a8380a 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

         write

bit     1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
        5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0

$a83800             r w a a a a a a a a
                    d r 2 2 2 2 1 1 1 1         break address(H)ch.A
                    e e 3 2 1 0 9 8 7 6

$a83802 a a a a a a a a a a a a a a a u
        1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 d         break address(L)ch.A
        5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 s

$a83804             r w a a a a a a a a
                    d r 2 2 2 2 1 1 1 1         break address(H)ch.B
                    e e 3 2 1 0 9 8 7 6

$a83806 a a a a a a a a a a a a a a a u
        1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 d         break address(L)ch.B
        5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 s

$a83808             r w a a a a a a a a
                    d r 2 2 2 2 1 1 1 1         break address(H)ch.C
                    e e 3 2 1 0 9 8 7 6

$a8380a a a a a a a a a a a a a a a a u
        1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 d         break address(L)ch.C
        5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 s
```

12.Extended I/O function (in the monitor mode)

g) REAL_TIME Clock

·Level 1 is generated by means of a timer interrupt from RTC62421.

〈〉 real time clock RTC-62421

```
$a85001 S1            * * * * 3 2 1 0---------sec ( *1 )
$a85003 S10           * * * * 3 2 1 0---------sec ( *10 )
$a85005 MI1           * * * * 3 2 1 0---------min ( *1 )
$a85007 MI10          * * * * 3 2 1 0---------min ( *10 )
$a85009 H1            * * * * 3 2 1 0---------hour ( *1 )
$a8500b H10           * * * * 3 2 1 0---------hour ( *10 )
$a8500d D1            * * * * 3 2 1 0---------day ( *1 )
$a8500f D10           * * * * 3 2 1 0---------day ( *10 )
$a85011 MO1           * * * * 3 2 1 0---------month ( *1 )
$a85013 MO10          * * * * 3 2 1 0---------month ( *10 )
$a85015 Y1            * * * * 3 2 1 0---------year ( *1 )
$a85017 Y10           * * * * 3 2 1 0---------year ( *10 )
$a85019 Week          * * * * 3 2 1 0---------week ( *1 )
$a8501b CD            * * * * 3 2 1 0---------control register D
$a8501d CE            * * * * 3 2 1 0---------control register E
$a8501f CF            * * * * 3 2 1 0---------control register F
```

〈〉 rtc interrupt mask

```
$a85801               on write
               * * * * * * * 0---------1:disenable
                                       0:enable (!int1 by timer)
                      on read
               * * * * 0 0 X X---------enable/disenable
                             +-----------timer int
```

12.Extended I/O function (in the monitor mode)

h) 82C255 (equivalent to two 8255s)

- The loader can be used even when the port of the MEGA DRIVE is not used.

```
<>  8255 no.1 printer in/out

$a86801 port-A
                  7 6 5 4 3 2 1 0---------data in (36pin conector)
$a86803 port-B
                  7 6 5 4 3 2 1 0---------data out(14pin conector)
$a86805 port-C
                  7 6 5 4 3 2 1 0
                  | | | | | | | +--------- !ack   (36pin_no.10)
                  | | | | | | +----------- !stb   (14pin_no. 1)
                  | | | | | +-------------  nc
                  | | | | +---------------  nc
                  | | | +----------------- !stb   (36pin_no. 1)
                  | | +-------------------  busy  (36pin_no.11)
                  | +--------------------- !init  (36pin_no.31)
                  +-----------------------  busy  (14pin_no.11)

$a86807         control


<>  8255 no.2                      34pin conector
                           gnd ... 19,21,23,25,27,29,31,33 (pin)
                           nc  ... 17,34(pin)

                  bit              7 6 5 4 3 2 1 0

$a86001           port-A           2 4 6 8 1 1 1 1 (pin_no.)
                                           0 2 4 6

$a86003           port-B           1 3 5 7 9 1 1 1 (pin_no.)
                                             1 3 5

$a86005           port-C           1 2 2 2 2 2 3 3 (pin_no.)
                                   8 0 2 4 6 8 0 2
$a86007           control
```

12.Extended I/O function (in the monitor mode)

i) EEPROM

Note:Wait 10msec or longer after one byte of data is written.

<> EEPROM 16Kbits (2Kbytes)

```
$a88001          lower_byte only
   |
   |
$a807ff
```

12.Extended I/O function (in the monitor mode)

j)-1 History

· Hisotry can be read and written (history is suspended).

1) 1+4N word
   $FFFF is set when history data is written.

2) 2+4N word
   a23-a15 (access address), fc2-0, r/w, break, monitor

3) 3+4N word
   a14-a1 (access address), uds, lds

4) 4+4N word
   Data value that is read from or written in the access address.

Notes:To access history, always access the address for 32768-times history (8192 steps by 4). (32768 times or its multiple including read and write). If the number of items are not correct, operation stops.

〈〉 cpu access history

```
$a87800              1word_address only

          1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
          5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0         (bit)          n=0,1fff

$a87800

          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 (1st,5th,9th,····,4n+1 word)

          m b b f f f r a a a a a a a a a
          o r g c c c / 2 2 2 2 1 1 1 1 1 (2nd,6th,10th,···,4n+2 word)
          n k a 2 1 0 w 3 2 1 0 9 8 7 6 5

          a a a a a a a a a a a a a a u l
          1 1 1 1 1 0 0 0 0 0 0 0 0 0 d d (3rd,7th,11th,···,4n+3 word)
          4 3 2 1 0 9 8 7 6 5 4 3 2 1 s s

          d d d d d d d d d d d d d d d d
          1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 (4th,8th,12th,···,4n+4 word)
          5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

mon = 0 ; monitor   Memory access while monitoring.
mon = 1 ; emulate   Memory access while emulating.

brk = 0 ; break   Break conditions occur.
brk = 1 ; no_break No break

12.Extended I/O function (in the monitor mode)

j)-2 History

```
                      ( mode_chage timing )

               start:            [power_on] or [break]
x  n  m         |
x  n  m         |
x  n  m         |
x  n  m         |
x  n  m   801058  MR 4E71 SP nop
x  n  m   XXXXXX  ME XXXX SD xxxx
x  n  m   80105A  MR 4E71 SP nop
x  n  m   XXXXXX  ME XXXX SD xxxx
x  n  m   80105C  MR 33C0 SP move d0,$xxxxxx                 cmd.[ to EMULATE ]
x  n  m   80105E  MR 00A8 SP -------,$a8xxxx
x  n  m   801060  MR 1000 SP -----------1000
x  n  m   801062  MR 4E73 SP rte
x  n  m   A81000  MW FFFF SD **  (write data to $a81000)     wrt.[ to EMULATE ]
x  n  m   801064  MR XXXX SP    (prog. pre fetch)
x  n  m   FFFFF6  MR 2000 SD                     read sr
x  n  m   FFFFF8  MR 00FF SD                           read pc. high
x  n  m   FFFFFA  MR 8000 SD                           read pc. low
x  H  e   FF8000  MR 341A SP (flip_mode to EMULATE)        exe.[ to EMULATE ]
x  H  e   FF8002  MR 51C9 SP
x  H  e     |
x  H  e     |
B  H  e     |
B  H  e     |
B  H  e   XXXXXX  MW XXXX SD                           write pc.low
B  H  e   fffffe  MR FFFE IA                           intrrupt_acknowldge
x  n  m   XXXXXX  MW XXXX SD  (flip_mode to MONITOR)   write sr
x  n  m   XXXXXX  MW XXXX SD                           write pc.high
x  n  m   XXXXXX  MR XXXX SP                           new   pc.address
|  |  |
|  |  +---------- m=monitor_mode       ;Monitor status
|  |              e=emulate_mode       ;Emulating status
|  |
|  +------------- H=history_on         ;History in operation
|                 n=no_history         ;History suspended
|
+---------------- B=break_occur        ;break conditions occur
                  x=no_break
```

13.Operation with ICE

a-1) Connection

Carefully insert the ICE probe facing the 68000 socket (IC16). Connect the ZAX's pins (CN16 near SW_1) to the external ICE break pins. (The leftmost pin of the ZAX's pins is connected to ground; the rightmost pin of the external ICE break pins is to ground.)

Connect the printer cable to the main unit and CN21.
Connect JOY-PAD to CN4 and CN5.
Connect the video cable to the TV and CN9.
Connect the power line cord.
Connect to the front power supply of the cartridge when the catridge is use.

a-2) Switch

SW4(dip_sw1)	Set it to the required position.
SW5(dip_sw2)	Set it to the required position.
SW6(dip_sw3)	Turn it off.
SW7(dip_sw4)	Turn it on.

b-1) ICE macros

```
macro ram                            *The SRAM mode is set.
MA 0A82000,0A87FFF=US
E/N 0A87000=0f
MA 0A82000,0A87FFF=NO

macro rom                            *The cartridge mode is set.
MA 0A82000,0A82FFF=US
E/N/b 0A82001=16
MA 0A82000,0A82FFF=NO

macro rw                             *The SRAM can be read and written.
MA 0A87000,0A87FFF=US
E/N 0A87000=1f
MA 0A87000,0A87FFF=NO

macro ro                             *The SRAM can only be read.
MA 0A87000,0A87FFF=US
E/N 0A87000=0f
MA 0A87000,0A87FFF=NO
```

b-2) How to set external breaks

```
ev adchk EXT=LO                      *The required event is set up.

b adchk                              *The required break point is set up.
```

Sega Ozisoft

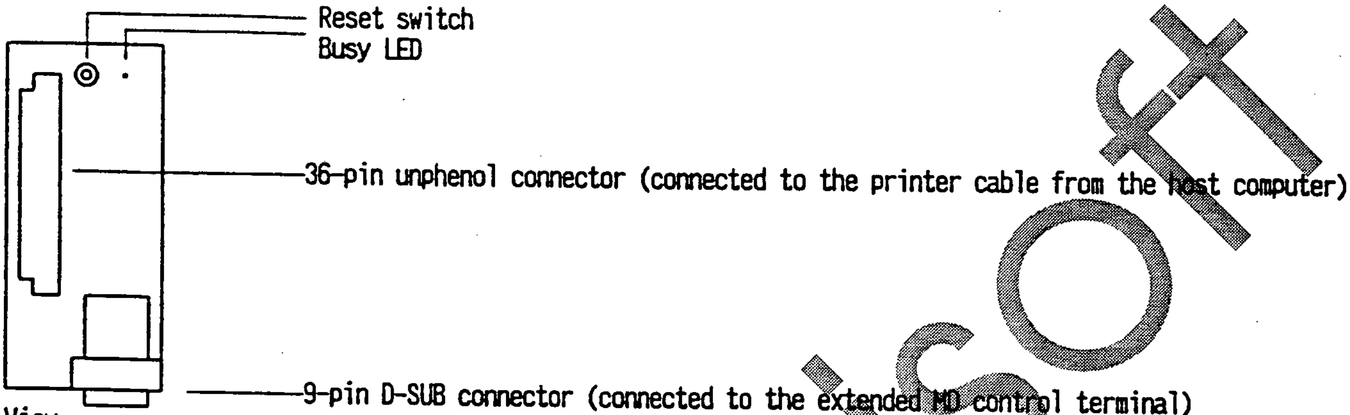
MEGA DRIVE LOADER
HANDLING MANUAL

SEGA ENTERPRISES, LTD.

Contents

§1 Outline

The loader serves to download program files to the MEGA DRIVE from the printer board (complying with Centronics) of the host computer. Since the loader is completely programmable, any computer having the printer board can be used regardless of the computer model.


Reset switch
Busy LED
36-pin unphenol connector (connected to the printer cable from the host computer)
9-pin D-SUB connector (connected to the extended MD control terminal)

Toy View

To simplify the description below, this manual assumes that MS-DOS (PC-9801 series and PC-286 series) is used.

§2 How to use the loader

① Start up MS-DOS and copy the files (ICD_BLK.COM and LD.S28)[1] onto the system disk.
② Set up the ICE, loader, cables, etc., and turn on the power.
③ Start up the ICE, and download LD.S28 by using the download function, etc. implemented in the ICE.
④ Exit the ICE while LD.S28 is running.
⑤ Output the files, which must be downloaded, to the printer port using ICD_LDR.COM.
⑥ Start up the ICE, and stop LD.S28.

The downloading of the program files is completed at this point.

[1]... Refer to 'Required files' in §3.

§3 Required files

Two files are required : ICD_LDR.COM and LD.S28
ICD_LDR.COM is a program for outputting program files to the printer port, and can send data much faster than COPY does.
To execute this function, type:
A)ICD_LDR Filename.Ext ↵
Operation returns to MS-DOS upon completion of outputting.[2]

LD.S28 is the receiver program whose start address is $FF0000. This program file is made by linking the program shown in appendix 2 to the assembler (based on the Motorola S format).

[2]... Refer to 'Trouble with downloading' in §4.

§4.Trouble with downloading

When trouble occurs in downloading, check the followings and perform the suggestions described below.

① Data is not properly loaded.
- The printer cable is not properly connected.
  →Connect the printer cable properly.
- The destination memory for downloading has been set to 'read only' in the board side.
  →Set it to 'read/write'.

② The host computer hangs up.*3
- Break points have been set up in the destination memory for downloading.
  →Remove the break points.
- The destination memory for downloading has been set to 'read only' by the mapping command.
  →Set it to 'read/write'.

*3...When the host computer hangs up, hold down the reset switch on the loader and press the stop key on the host computer.

§5. Description of the registers of the loader

◎ $A10007 : Extended control pin data register

For read

×	×	×	STB	D3	D2	D1	D0

For write

0	SEL	G	0	0	0	0	0

◎ $A1000D : Extended control pin command register

Write only

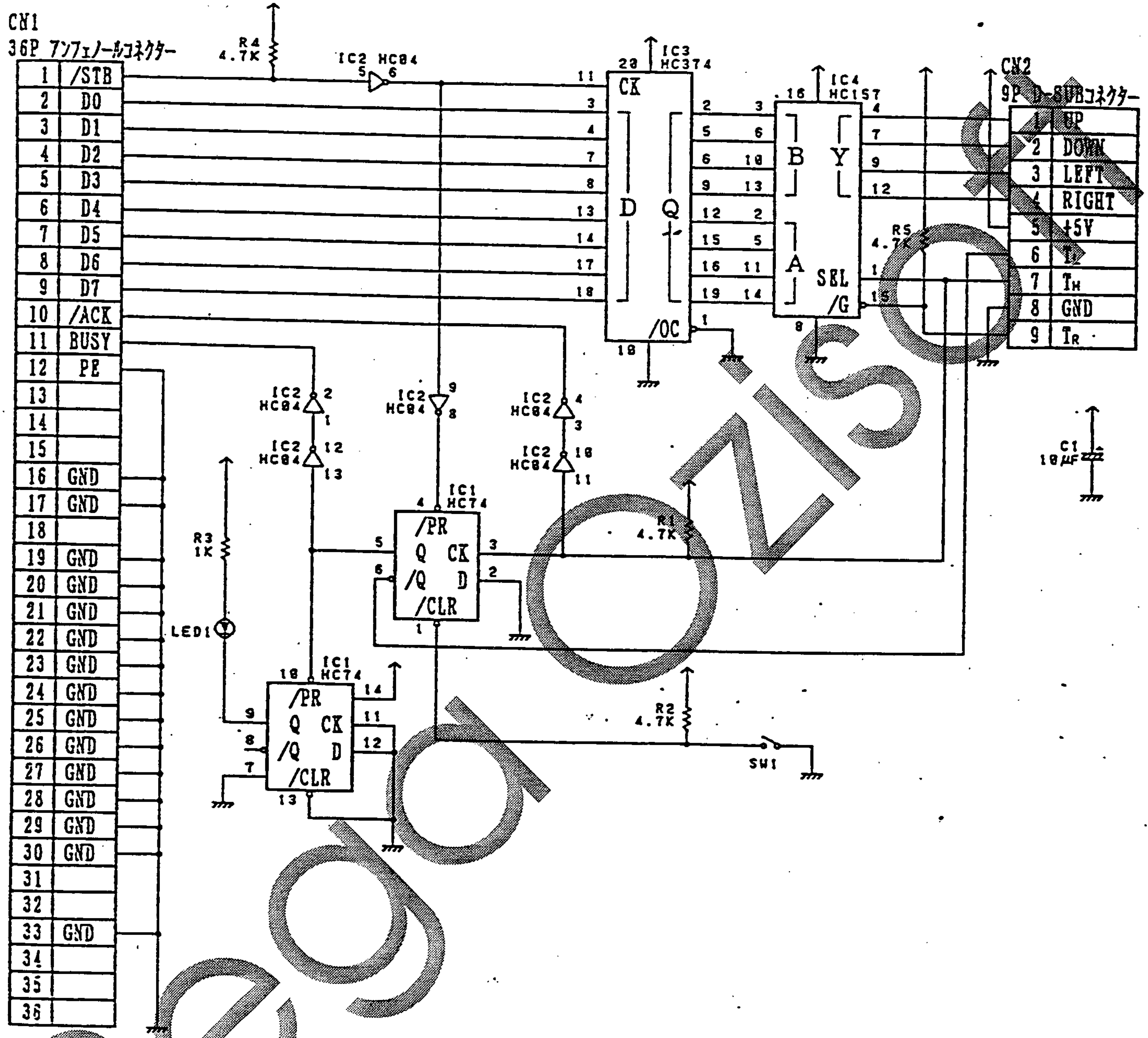
0	SEL	G	0	0	0	0	0

Refer to the MD software manual for the specifications on the above registers. The operational procedures are only described in this document.

① Write $60 in address $A1000D first, and set T_R and T_H to 'output'.
② Write $00 in $A10007.
③ Write $40 in $A10007.
④ Check bit 4 in $A10007 repeatedly until it becomes 0.
⑤ Read bits 0 to 3 in $A10007 (lower 4-bits).
⑥ Write $00 in $A10007.
⑦ Read bits 0 to 3 in $A10007 (higher 4-bits)
⑧ Repeat steps ③ to ⑦.

**** Care is required for bit 6 in $A1000D, since this is used for the reset judgement.

§6. Loader circuit drawing



** A 0.22μF axial capacitor is connected to almost every IC.

Appendix 1

MD sample program for sending data (PC-9801 and PC-286 series with MS-DOS)

```
                                             ;------------------------------------------------
                                             ;        ERX-318 file loader Ver. 01.02
                                             ;               1987.04.24 : m.kobayashi
                                             ;               1988.12.09 : m.kobayashi
                                             ;               1989.04.14 : Y.Sake
                                             ;                      (for MEGA DRIVE)
                                             ;------------------------------------------------

=-0001                           true        equ      -1
= 0000                           false       equ      0

=-0001                           debug       =        true
                                 ;debug      =        false

= 000E                           p_stb       =        0eh      ; for VX
                                 ;p_stb      =        04h      ; for XA/XL

0000                             cseg        segment
                                             assume   cs:cseg,ds:cseg,es:cseg

0080                                         org      80h
0080      80 [                   cmdln       db       128 dup (?)
               ??
                   ]

                                 ;
0100                                         org      100h
0100                             entry:
0100  FA                                     cli                              ; di = disable interrupt
0101  E9 03F7 R                              jmp      main
                                 ;
0104  0D 0A 24                   msg_crlf:   db       0dh,0ah,'$'
0107  82 CC 83 8D 81 5B          msg_comp:   db       'のロードを終了しました$'
      83 68 82 F0 8F 49                               [ Loading is completed$]
      97 B9 82 B5 82 DC
      82 B5 82 BD 24
011E  83 70 83 89 83 81          msg_fail:   db       'パラメータに誤りがあります$'
      81 5B 83 5E 82 C9                               [Invalid parameter$]
      8C EB 82 E8 82 AA
      82 A0 82 E8 82 DC
      82 B7 24
0139  83 41 83 4E 83 56          msg_error_01:  db    'アクション・コードが無効です$'
      83 87 83 93 81 45                                [Invalid action code$]
      83 52 81 5B 83 68
      82 AA 96 B3 8C F8
      82 C5 82 B7 24
0156  83 74 83 40 83 43          msg_error_02:  db    'ファイルが見つかりません$'
      83 8B 82 AA 8C A9                                [File not found$]
      82 C2 82 A9 82 E8
      82 DC 82 B9 82 F1
      24
```

```
016F  83 70 83 58 82 AA    msg_error_03:   db    'パスが見つかりません$'
      8C A9 82 C2 82 A9                          [Bus not found$]
      82 E8 82 DC 82 B9
      82 F1 24
0184  83 49 81 5B 83 76    msg_error_04:   db    'オープンされているファイルが多すぎます$'
      83 93 82 B3 82 EA                          [Too many open files$]
      82 C4 82 A2 82 E9
      83 74 83 40 83 43
      83 8B 82 AA 91 BD
      82 B7 82 AC 82 DC
      82 B7 24
01AB  83 41 83 4E 83 5A    msg_error_05:   db    'アクセスが否定されました$'
      83 58 82 AA 94 DB                          [Access denied$]
      92 E8 82 B3 82 EA
      82 DC 82 B5 82 BD
      24
01C4  83 49 81 5B 83 76    msg_error_06:   db    'オープンされていないハンドルが使用されました$'
      83 93 82 B3 82 EA                          [Unopened handle was used$]
      82 C4 82 A2 82 C8
      82 A2 83 6E 83 93
      83 68 83 8B 82 AA
      8E 67 97 70 82 B3
      82 EA 82 DC 82 B5
      82 BD 24
01F1  83 81 83 82 83 8A    msg_error_07:   db    'メモリ内のデータが破壊されています$'
      93 E0 82 CC 83 66                          [Data in memory was destroyed$]
      81 5B 83 5E 82 AA
      94 6A 89 F3 82 B3
      82 EA 82 C4 82 A2
      82 DC 82 B7 24
0214  83 81 83 82 83 8A    msg_error_08:   db    'メモリが足りません$'
      82 AA 91 AB 82 E8                          [Insufficient memory space$]
      82 DC 82 B9 82 F1
      24
0227  8E 77 92 E8 82 B3    msg_error_09:   db    '指定されたブロックは解放できません$'
      82 EA 82 BD 83 75                          [Specified block cannot be released$]
      83 8D 83 62 83 4E
      82 CD 89 F0 95 FA
      82 C5 82 AB 82 DC
      82 B9 82 F1 24
024A  8E 77 92 E8 82 B3    msg_error_10:   db    '指定された環境ストリングが長すぎます$'
      82 EA 82 BD 8A C2                          [Too long environment string was specified$]
      8B AB 83 58 83 67
      83 8A 83 93 83 4F
      82 AA 92 B7 82 B7
      82 AC 82 DC 82 B7
      24
026F  82 64 82 77 82 64    msg_error_11:   db    'ＥＸＥファイルの情報が不正です$'
      83 74 83 40 83 43                          [Illegal information on EXE file$]
      83 8B 82 CC 8F EE
      95 F1 82 AA 95 73
      90 B3 82 C5 82 B7
      24
—028E 83 41 83 4E 83 5A    msg_error_12:   db    'アクセス・コードが無効です$'
      83 58 81 45 83 52                          [Invalid access code$]
      81 5B 83 68 82 AA
```

```
        96 B3 8C F8 82 C5
        82 B7 24
02A9    83 66 81 5B 83 5E       msg_error_13:   db      'データが無効です$'
        82 AA 96 B3 8C F8                                [Invalid data$]
        82 C5 82 B7 24
02BA    83 68 83 89 83 43       msg_error_15:   db      'ドライブ指定が無効です$'
        83 75 8E 77 92 E8                                [Invalid drive assignment$]
        82 AA 96 B3 8C F8
        82 C5 82 B7 24
02D1    83 4A 83 8C 83 93       msg_error_16:   db      'カレント・ディレクトリは削除できません$'
        83 67 81 45 83 66                                [Current drive cannot be deleted$]
        83 42 83 8C 83 4E
        83 67 83 8A 82 CD
        8D ED 8F 9C 82 C5
        82 AB 82 DC 82 B9
        82 F1 24
02F8    88 D9 82 C8 82 C1       msg_error_17:   db      '異なった装置のパスが指定されました$'
        82 BD 91 95 92 75                                [Different drive path was specified$]
        82 CC 83 70 83 58
        82 AA 8E 77 92 E8
        82 B3 82 EA 82 DC
        82 B5 82 BD 24
031B    82 B1 82 EA 88 C8       msg_error_18:   db      'これ以上ファイルが存在しません$'
        8F E3 83 74 83 40                                [No more files exist$]
        83 43 83 8B 82 AA
        91 B6 8D DD 82 B5
        82 DC 82 B9 82 F1
        24
                                ;
033A    00                      $033d:          db      0
033B    00                      $033e:          db      0
033C    00                      $033f:          db      0
                                ;
033D    43 4F 4E                devcon:         db      'CON'
0340    41 55 58                devaux:         db      'AUX'
                                ;
0343                            errtbl:
0343    0139 R                                  dw      msg_error_01
0345    0156 R                                  dw      msg_error_02
0347    016F R                                  dw      msg_error_03
0349    0184 R                                  dw      msg_error_04
034B    01AB R                                  dw      msg_error_05
034D    01C4 R                                  dw      msg_error_06
034F    01F1 R                                  dw      msg_error_07
0351    0214 R                                  dw      msg_error_08
0353    0227 R                                  dw      msg_error_09
0355    024A R                                  dw      msg_error_10
0357    026F R                                  dw      msg_error_11
0359    028E R                                  dw      msg_error_12
035B    02A9 R                                  dw      msg_error_13
035D    0000                                    dw      0
035F    02BA R                                  dw      msg_error_15
0361    02D1 R                                  dw      msg_error_16
0363    02F8 R                                  dw      msg_error_17
0365    031B R                                  dw      msg_error_18
```

```
                            ;
0367  FFFF                  $036a:        dw      -1
0369  0000                  $036c:        dw      0
036B  0000                  $036e:        dw      0
036D  0000                  $0370:        dw      0
036F  0000                  $0372:        dw      0
0371  0000                  $0374:        dw      0
0373  0000                  $0376:        dw      0
0375  0000                  $0378:        dw      0
                            ;
0377  0000 0000 0000 0000                         dw      0,0,0,0,0,0,0,0
      0000 0000 0000 0000
0387  0000 0000 0000 0000                 dw      0,0,0,0,0,0,0,0
      0000 0000 0000 0000
0397  0000 0000 0000 0000                 dw      0,0,0,0,0,0,0,0
      0000 0000 0000 0000
03A7  0000 0000 0000 0000                 dw      0,0,0,0,0,0,0,0
      0000 0000 0000 0000
03B7  0000 0000 0000 0000                 dw      0,0,0,0,0,0,0,0
      0000 0000 0000 0000
03C7  0000 0000 0000 0000                 dw      0,0,0,0,0,0,0,0
      0000 0000 0000 0000
03D7  0000 0000 0000 0000                 dw      0,0,0,0,0,0,0,0
      0000 0000 0000 0000
03E7  0000 0000 0000 0000                 dw      0,0,0,0,0,0,0,0
      0000 0000 0000 0000

                            ;
03F7                        main:
03F7  BC 03F7 R                           mov     sp,offset main
03FA  BB 05B4 R                               mov     bx,offset bottom
03FD  83 C3 0F                            add     bx,15
0400  B1 04                               mov     cl,04h
0402  D3 EB                               shr     bx,cl
0404  B4 4A                               mov     ah,4ah
0406  CD 21                               int     21h
0408  73 03                               jnb     $0410
040A  E9 0549 R                           jmp     $05c0
040D                        $0410:
040D  BB 1000                             mov     bx,1000h
0410  B4 48                               mov     ah,48h
0412  CD 21                               int     21h
0414  73 13                               jnb     $042c
0416  0B DB                               or      bx,bx
0418  75 03                               jnz     $0420
041A  E9 0549 R                           jmp     $05c0
041D                        $0420:
041D  B4 48                               mov     ah,48h
041F  CD 21                               int     21h
                            ;
0421  B1 04                               mov     cl,4
0423  D3 E3                               shl     bx,cl
0425  89 1E 0367 R                        mov     word ptr $036a,bx
0429                        $042c:
0429  A3 0369 R                           mov     word ptr $036c,ax
042C  BE 0081 R                           mov     si,offset cmdln[1]
042F  E8 0566 R                           call    $05df
0432  89 1E 036B R                        mov     word ptr $036e,bx
```

```
0436  72 44                                 jb      $0467
0438  E8 059B R                             call    $060e
043B  73 3F                                 jnb     $0467
043D  E9 0555 R                             jmp     $05cc

0440                     msg_not_ready:
0440  82 68 82 62 82 63                     db      'ＩＣＤ側の用意が出来ていません',07,'$'
      91 A4 82 CC 97 70                              [Not ready ICD]
      88 D3 82 AA 8F 6F
      97 88 82 C4 82 A2
      82 DC 82 B9 82 F1
      07 24
0460                     msg_ready:
0460  0D 1B 5B 4B 0D 24                     db      13,27,'[K',13,'$'
0466                     msg_about:
0466  0D 0A 93 5D 91 97                     db      13,10,'転送を中止しました',07,'$'
      82 F0 92 86 8E 7E                                  [Stop transfer]
      82 B5 82 DC 82 B5
      82 BD 07 24

047C                     $0467:
047C  E4 42                                 in      al,42h
047E  A8 04                                 test    al,4
0480  75 1B                                 jnz     p_skip

0482  BA 0440 R                             mov     dx,offset msg_not_ready
0485  B4 09                                 mov     ah,09h
0487  CD 21                                 int     21h
0489                           p_loop:
0489  B4 06                                 mov     ah,06h
048B  B2 FF                                 mov     dl,0ffh
048D  CD 21                                 int     21h
048F  75 1D                                 jnz     k_check

0491  E4 42                                 in      al,42h
0493  A8 04                                 test    al,4
0495  74 F2                                 jz      p_loop
0497  BA 0460 R                             mov     dx,offset msg_ready
049A  E8 055A R                             call    $05d1
049D                           p_skip:
049D  B8 3D00                               mov             ax,3d00h
04A0  8B 16 036B R                          mov     dx,word ptr $036e
04A4  CD 21                                 int     21h
04A6  A3 036F R                             mov     word ptr $0372,ax
04A9  73 0D                                 jnb     $04e6
04AB  E9 0549 R                             jmp     $05c0
04AE                           k_check:
04AE  3C 03                                 cmp     al,3
04B0  75 D7                                 jnz     p_loop

04B2  BA 0466 R                             mov     dx,offset msg_about
04B5  E9 0541 R                             jmp     $05bb

04B8                     $04e6:
04B8  B4 3F                                 mov     ah,3fh
04BA  33 D2                                 xor     dx,dx
```

```
04BC  8B 0E 0367 R                       mov     cx,word ptr $036a
04C0  8B 1E 036F R                       mov     bx,word ptr $0372
04C4  1E                                 push    ds
04C5  8E 1E 0369 R                       mov     ds,word ptr $036c
04C9  CD 21                              int     21h
04CB  1F                                 pop     ds
04CC  73 03                              jnb     $052b
04CE  EB 79 90                           jmp     $05c0
04D1                          $052b:
04D1  0B C0                              or      ax,ax
04D3  75 0A                              jnz     $0539
04D5  C6 06 033A R FF                    mov     byte ptr $033d,0ffh
04DA  A3 0371 R                          mov     word ptr $0374,ax
04DD  EB 28                              jmp     short $056b
04DF                          $0539:
04DF  8B C8                              mov     cx,ax
04E1  1E                                   push    ds
04E2' 8E 1E 0369 R                       mov     ds,word ptr $036c
04E6  2E: 3B 06 0367 R                   cmp     ax,cs:word ptr $036a
04EB  74 06                              jz      $0555
04ED  8B F0                              mov     si,ax
04EF  C6 04 1A                           mov     byte ptr [si],1ah
04F2  41                                 inc     cx
04F3                          $0555:
04F3  33 F6                              xor     si,si
04F5                          $0557:
04F5  AC                                 lodsb
04F6  3C 1A                              cmp     al,1ah
04F8  74 04                              jz      $0560
04FA  E2 F9                              loop    $0557
04FC  EB 07                              jmp     short $0566
04FE                          $0560:
04FE  2E: C6 06 033A R FF                mov     byte ptr cs:$033d,0ffh
0504  4E                                 dec     si
0505                          $0566:
0505  1F                                 pop     ds
0506  89 36 0371 R                       mov     word ptr $0374,si
050A                          $056b:
050A  8B 0E 0371 R                       mov     cx,word ptr $0374
050E  E3 1D                              jcxz    $058c

0510  1E                                 push    ds
0511  8E 1E 0369 R                       mov     ds,word ptr $036c
0515  33 F6                              xor     si,si
0517                          $057f:
0517  E4 42                              in      al,42h
0519  A8 04                              test    al,4
051B  74 FA                              jz      $057f
051D  AC                                 lodsb
051E  E6 40                                   out     40h,al
0520  B0 0E                              mov     al,offset p_stb
                                         ;cli                            ; di = disable interrupt
0522  E6 46                              out     46h,al
0524  FE C0                              inc     al
0526  E6 46                              out     46h,al
                                         ;sti                            ; ei = enable interrupt
```

```
0528  E2 ED                              loop    $057f
052A  1F                                 pop     ds
052B  EB 00                              jmp     short $05ae
052D                      $058c:
052D                      $05ae:
052D  80 3E 033A R 00                    cmp     byte ptr $033d,0
0532  75 02                              jnz     $05b8
0534  EB 82                              jmp     $04e6
0536                      $05b8:
0536  8B 16 036B R                       mov     dx,word ptr $036e
053A  B4 09                              mov     ah,09h
053C  CD 21                              int     21h
053E  BA 0107 R                          mov     dx,offset msg_comp
0541                      $05bb:
0541  E8 055A R                          call    $05d1
0544  FB                                 sti                        ; ei = enable interrupt
                                         ;int    20h
0545  B4 4C                              mov     ah,04ch
0547  CD 21                              int     21h                ; return command.com
0549                      $05c0:
0549  BB 0343 R                          mov     bx,offset errtbl
054C  48                                 dec     ax
054D  D1 E0                              shl     ax,1
054F  03 D8                              add     bx,ax
0551  8B 17                              mov     dx,[bx]
0553  EB EC                              jmp     $05bb
0555                      $05cc:
0555  BA 011E R                          mov     dx,offset msg_fail
0558  EB E7                              jmp     $05bb
055A                      $05d1:
055A  B4 09                              mov     ah,09h
055C  CD 21                              int     21h
055E  BA 0104 R                          mov     dx,offset msg_crlf
0561  B4 09                              mov     ah,09h
0563  CD 21                              int     21h
0565  C3                                 ret
0566                      $05df:
0566  E8 05AF R                          call    $0622
0569  8B DE                              mov     bx,si
056B                      $05e4:
056B  AC                                 lodsb
056C  3C 61                              cmp     al,'a'
056E  72 09                              jb      $05f2
0570  3C 7A                              cmp     al,'z'
0572  77 05                              ja      $05f2
0574  2C 20                              sub     al,20h
0576  88 44 FF                           mov     [si-1],al
0579                      $05f2:
0579  3C 0D                              cmp     al,0dh
057B  74 15                              jz      $0608
057D  3C 20                              cmp     al,' '
057F  74 08                              jz      $0602
0581  3C 09                              cmp     al,9
0583  74 04                              jz      $0602
0585  3C 3D                              cmp     al,'='
0587  75 E2                              jnz     $05e4
```

```
0589                    $0602:
0589  C6 44 FF 00                    mov     byte ptr [si-1],0
058D  C6 04 24                       mov     byte ptr [si],'$'

0590  F8                             clc
0591  C3                             ret
0592                    $0608:
0592  C6 44 FF 00                    mov     byte ptr [si-1],0
0596  C6 04 24                       mov     byte ptr [si],'$'
0599  F9                             stc
059A  C3                             ret
059B                    $060e:
059B  E8 05AF R                      call    $0622
059E                    $0611:
059E  AC                             lodsb
059F  3C 0D                          cmp     al,0dh
05A1  74 0A                          jz      $0620
05A3  3C 20                              cmp     al,' '
05A5  74 F7                          jz      $0611
05A7  3C 09                          cmp     al,9
05A9  74 F3                          jz      $0611
05AB  F9                             stc
05AC  C3                             ret
05AD                    $0620:
05AD  F8                             clc
05AE  C3                             ret
05AF                    $0622:
05AF  E8 059E R                      call    $0611
05B2  4E                             dec     si
05B3  C3                             ret
05B4                    bottom:

05B4                    cseg         ends
                                     end     entry
```

Appendix 2

```
                                    **************************************************
                                    *       Loader Program s28 & s37
                                    *       Programed By Y.Sake      27.Feb.1989
                                    **************************************************

00FF0000                                          org      $ff0000
            00A1 0007               port          equ      $a10007
            00A1 000D               cont          equ      $a1000d
            0000 0004               stb           equ      $4
            0000 0020               g             equ      $20
            0000 0040               sel           equ      $40
            0000 0000               s2_format     equ      0
            FFFF FFFF               s3_format     equ      -1

00FF0000    41F9 00A1 0007                        lea      port,a0
00FF0006    43F9 00A1 000D                        lea      cont,a1
00FF000C    12BC 0060                             move.b   #g+sel,(a1)      ; b5 & b6 set output
00FF0010    10BC 0000                             move.b   #0,(a0)
00FF0014    10BC 0040                             move.b   #sel,(a0)

                                    ;take character 'S'
00FF0018                            start:
00FF0018    0810 0004                             btst.b   #stb,(a0)
00FF001C    66FA                                  bne.b    start

00FF001E    1010                                  move.b   (a0),d0
00FF0020    10BC 0000                             move.b   #0,(a0)
00FF0024    1210                                  move.b   (a0),d1
00FF0026    10BC 0040                             move.b   #sel,(a0)
00FF002A    0200 000F                             and.b    #$0f,d0
00FF002E    E909                                  lsl.b    #4,d1
00FF0030    8200                                  or.b     d0,d1
00FF0032    0C01 0053                             cmp.b    #'S',d1
00FF0036    66E0                                  bne.b    start

                                    ;take character '2'
00FF0038                                   ?loop:
00FF0038    0810 0004                             btst.b   #stb,(a0)
00FF003C    66FA                                  bne.b    ?loop

00FF003E    1010                                  move.b   (a0),d0
00FF0040    10BC 0000                             move.b   #0,(a0)
00FF0044    1210                                  move.b   (a0),d1
00FF0046    10BC 0040                             move.b   #sel,(a0)
00FF004A    0200 000F                             and.b    #$0f,d0
00FF004E    E909                                  lsl.b    #4,d1
00FF0050    8200                                  or.b     d0,d1
00FF0052    0C01 0032                             cmp.b    #'2',d1
00FF0056    6604                                  bne.b    ?s3_check
```

```
00FF0058  7800                               moveq.l #s2_format,d4
00FF005A  6008                               bra.b   take_data_wide

00FF005C                         ?s3_check:
00FF005C  0C01 0033                          cmp.b   #'3',d1
00FF0060  66B6                               bne.b   start

00FF0062  78FF                               moveq.l #s3_format,d4


                         ;take data wide
00FF0064                 take_data_wide:
00FF0064  7A01                               moveq.l #2-1,d5
00FF0066                 take_wide:
00FF0066  E90F                               lsl.b   #4,d7
00FF0068                         ?loop:
00FF0068  0810 0004                          btst.b  #stb,(a0)
00FF006C  66FA                               bne.b   ?loop

00FF006E  1010                               move.b  (a0),d0
00FF0070  10BC 0000                          move.b  #0,(a0)
00FF0074  1210                               move.b  (a0),d1
00FF0076  10BC 0040                          move.b  #sel,(a0)
00FF007A  0200 000F                          and.b   #$0f,d0
00FF007E  E909                               lsl.b   #4,d1
00FF0080  8200                               or.b    d0,d1
00FF0082  0401 0030                          sub.b   #'0',d1
00FF0086  0C01 0009                          cmp.b   #9,d1
00FF008A  6302                               bls.b   ?jump

00FF008C  5F01                               sub.b   #7,d1
00FF008E                         ?jump:
00FF008E  8E01                               or.b    d1,d7
00FF0090  51CD FFD4                          dbra    d5,take_wide

00FF0094  0C04 0000                          cmp.b   #s2_format,d4
00FF0098  6706                               beq.b   take_address_s2

00FF009A  5B07                               subq.b  #5,d7
00FF009C  7A07                               moveq.l #8-1,d5
00FF009E  6004                               bra.b   take_address_s3


                         ;take address
00FF00A0                 take_address_s2:
00FF00A0  5907                               subq.b  #4,d7
00FF00A2  7A05                               moveq.l #6-1,d5

00FF00A4                 take_address_s3:
00FF00A4                 take_adrs:
00FF00A4  E98E                               lsl.l   #4,d6
00FF00A6                         ?loop:
00FF00A6  0810 0004                          btst.b  #stb,(a0)
00FF00AA  66FA                               bne.b   ?loop

00FF00AC  1010                               move.b  (a0),d0
00FF00AE  10BC 0000                          move.b  #0,(a0)
00FF00B2  1210                               move.b  (a0),d1
```

```
00FF00B4    10BC 0040                move.b   #sel,(a0)
00FF00B8    0200 000F                and.b    #$0f,d0
00FF00BC    E909                     lsl.b    #4,d1
00FF00BE    8200                     or.b     d0,d1
00FF00C0    0401 0030                sub.b    #'0',d1
00FF00C4    0C01 0009                cmp.b    #9,d1
00FF00C8    6302                     bls.b    ?jump

00FF00CA    5F01                     sub.b    #7,d1
00FF00CC                    ?jump:
00FF00CC    8C01                     or.b     d1,d6
00FF00CE    51CD FFD4                dbra     d5,take_adrs

00FF00D2    2446                     move.l   d6,a2

                   ;take data
00FF00D4           take_data:
00FF00D4    7A01                     moveq.l  #2-1,d5
00FF00D6           take_digit:
00FF00D6    E90E                     lsl.b    #4,d6
00FF00D8                    ?loop:
00FF00D8    0810 0004                btst.b   #stb,(a0)
00FF00DC    66FA                     bne.b    ?loop

00FF00DE    1010                     move.b   (a0),d0
00FF00E0    10BC 0000                move.b   #0,(a0)
00FF00E4    1210                     move.b   (a0),d1
00FF00E6    10BC 0040                move.b   #sel,(a0)
00FF00EA    0200 000F                and.b    #$0f,d0
00FF00EE    E909                     lsl.b    #4,d1
00FF00F0    8200                     or.b     d0,d1
00FF00F2    0401 0030                sub.b    #'0',d1
00FF00F6    0C01 0009                cmp.b    #9,d1
00FF00FA    6302                     bls.b    ?jump

00FF00FC    5F01                     sub.b    #7,d1
00FF00FE                    ?jump:
00FF00FE    8C01                     or.b     d1,d6
00FF0100    51CD FFD4                dbra     d5,take_digit

00FF0104    14C6                     move.b   d6,(a2)+
00FF0106    5307                     subq.b   #1,d7
00FF0108    66CA                     bne.b    take_data

00FF010A    6000 FF0C                bra.w    start

00FF010E                             end
```

Appendix 3
ZAX ERX-318 MACRO

```
MACRO LD
ECHO KILL
LET @1=MB(0A1000D)
L LD.S28
R RES
R SSP=0
B *=0FF
G 0FF0000
EXEC ICD_LDR %1.S28
STOP
E/B/N 0A1000D=@1
```

----- MD Manual END -----

----- Contents -----

GENESIS/OUTLINE OF DEVELOPMENT TOOLS

1) SUPER MEGA DRIVE

MEGA DRIVE(GENESIS) software development target. No loader and noRAM board are required. Various functions are implemented.

Supplied components: Specifications, power supply, ICE cable, break cable and loader program

2) 1M RAM/4m ROM board

1Mbit RAM + 4Mbit ROM + 64Kbit Backup RAM
It can be used for carious purposes including sound tools.

Supplied component: Specification

3) 4M ROM board A

Test ROM board. The selection of ROM types and ROM capacities can be done by using the jumpers on the back side.

Supplied component: Specification

4) 8M ROM board 64

Test ROM board. The selection of ROM types and ROM capacities can be done by using the dip switches. The 64K bit Backup RAM is implemented.

Supplied component: Specification

5)8M ROM board 256

Test ROM board. The selection of ROM tyeps and ROM capacities can be done by using the dipswitches. The 256Kbit Backup RAM is implemented.

Supplied component: Specification

6) 8Mbit D-RAM board

This board is required when the marketed MEGA DRIVE is targeted; it is not required when the SUPER MEGA DRIVE is used.

7)ICE adaptor

This adaptor is used between the ZAX's ICE and the marketed MEGA DRIVE. This is used to reduce ICE noises (However, we cannot guarantee the proper operation of ICEs other than the ZAX ICE).

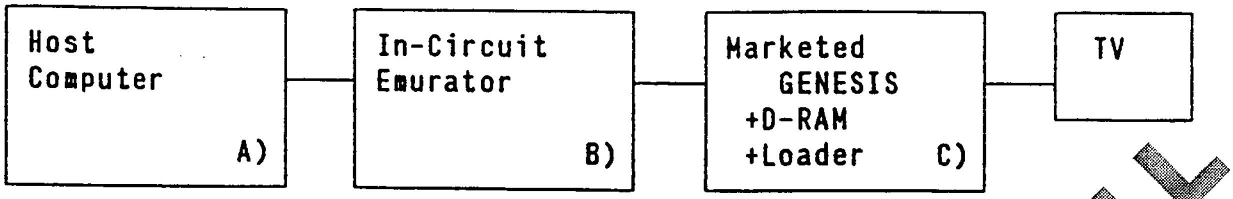
8)Loader

This board is used to down-load files from the printer port of the host computer to the MEGA DRIVE at a fast rate. This is not required when the SUPER MEGA DRIVE is used.

Supplied component: Specifications and sample software.

GENESIS software development environments (reference)

Construction example <1>



A) Host computer

There are no specified models.

B) ICE..The system may not work because of mismatch with the GENESIS.

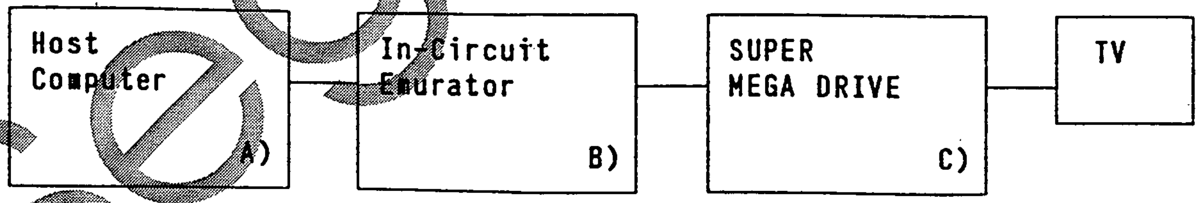
ZAX's ICE.... No problem. However, it needs to be specifified that it is used for the GENESIS development purpose, when it is purchased.

Other ICEs... Unknown sice SEGA has not checked them yet. SEGA cannot localize the problem if they do not work without any abnormality on the RAM board and GENESIS. Please ask the dealer of your ICE to localize the problem.

C) Marketed GENESIS... The following modifications are required.

- * To insert an ICE probe, the CPU should be mounted on a socket.
- * The modifications shown in figure 1 have to be made, since it is very sensitive to moises from the ICE and the power supply. It needs to be operated under the good power line environments by, for example, not putting many roads on one electric outlet, not installing large electric machines nearby, and applying a moise filter to the power supply.
- * The 8M-DRAM board and Loader is also required (option).

Construction example <2>

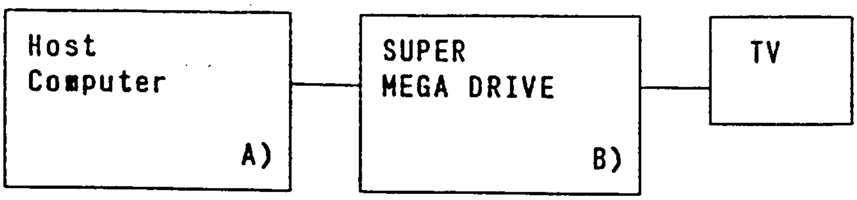


A) There are no specified models.

B) There are no specified models, since mismatch never occurs.

C) No modifications are required. No D-RAM board and loader is required.

Construction example <3>



A) There are no specified models.

B) No modifications are required. However, the software for controlling SMD needs to be developed.

----- END -----

Notes on the development 8M D-RAM board

The development D-RAM board contains:

The D-RAM area in $00000 to $0FFFFF
The S-RAM area on the low-byte side of $200000 to $203FFF.

** How to use

- Immediately after power-on:

  Write 0100H at $A11000;
  Write 0001H at $A130F0;
  Then the D-RAM board becomes ready to start (green lights).

- During operation:

  Write 0003H at $A130F0;
  Then the D-RAM becomes a read-only memory (red lights indicating write disable); or

  Write 0001H at $A130F0;
  Then the D-RAM becomes read/write enable.

Notes:
When the "NO_MEMORY ACCESS" break occurs from the ICE, do "MA 0A13000,0A13FFF=US", and follow the operational procedures. Then write 0003H at $A130F0 before performing the wmulation with the ICE, do "MA 0A13000, 0A13FFF=NO" to completely protect the D-RAM, and perform the emulations of "r reset" and "go"

Value at $A130F0	MAPPING	WRITE PROTECT
0 Green=off, Red=odd	$400000 - $4FFFFF (D-RAM) $600000 - $603FFF (S-RAM)	OFF
1 Green=on, Red=off	$000000 - $0FFFFF (D-RAM) $200000 - $203FFF (S-RAM)	OFF
2 Green=off, red=on	$400000 - $4FFFFF (D-RAM) $200000 - $203FFF (S-RAM)	ON (ROM mode) OFF
3 Green=on, red=on	$000000 - $0FFFFF (D-RAM) $200000 - $203FFF (S-RAM)	ON (ROM mode) OFF

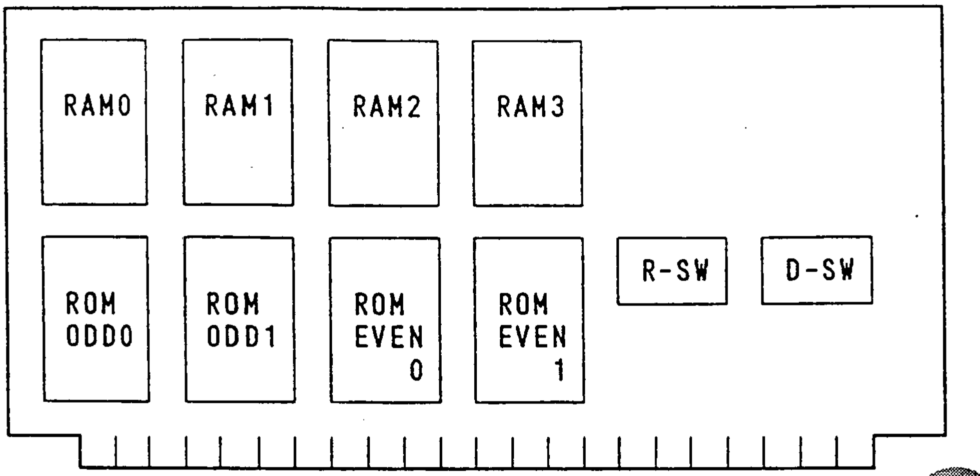
Notes:

The S-RAM comprises odd addresses of $XX0000 - $XX3FFF.

When mapping is changed to $400000 - $4FFFFF, general cartridges which can be used as the extended RAM space with FDD in use should be developed with $A130F0 = 1,3.

2.20-'91

1MRAM・4MROM BD DOCUMENTATION



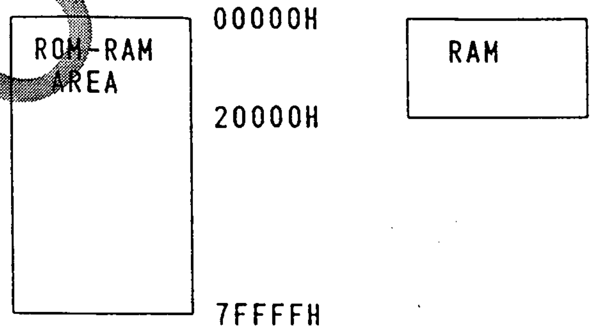
① ROM-size is 80000H(512KBYTE)

② RAM-size is 20000H(128KBYTE)

The ratary switch is used to select the address used in RAM.
(if RAM overlaps the memory in ROM. RAM memory will take priority)
RAM data will save to backup-Ram

③ Rotary switch and RAM address

SW0	00000H-1FFFFH
1	08000H-27FFFH
2	10000H-27FFFH
3	18000H-37FFFH
4	20000H-3FFFFH
5	28000H-47FFFH
6	30000H-4FFFFH
7	38000H-57FFFH
8	40000H-5FFFFH
9	48000H-57FFFH
A	50000H-6FFFFH
B	58000H-77FFFH
C	60000H-7FFFFH
D	68000H-7FFFFH
E	70000H-7FFFFH
F	200000H-21FFFFH



④RAM address is selected by rotary switch.

RT SW=0

0000H	RAM 0
8000H	RAM 1
10000H	RAM 2
18000H	RAM 3

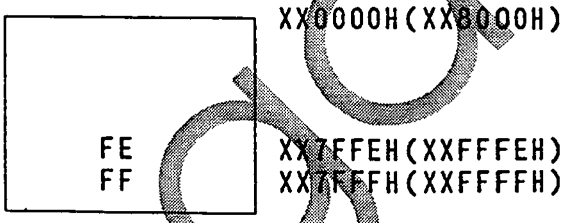
RT SW=1

8000H	RAM 1
10000H	RAM 2
18000H	RAM 3
20000H	RAM 0

⑤Control port(A130E0H-WORD)
(1)Set write-protection to RAM
BIT4=0 You cannot WRITE
BIT4=1 You can WRITE

(2)Indication of Bank-address
By setting '1' to bit0-3, you can indicate the bank-address at last word of each of the four RAMs.
If you are indicating the bank-address, you cannot write data at last word of RAM.
And the bank-address was indicated at bit15 to bit23.

Correspondence
Bit 0 ----- RAM 0
Bit 1 ----- RAM 1
Bit 2 ----- RAM 2
Bit 3 ----- RAM 3



RAM ARIA	FE							
Addrs BIT	A23	A22	A21	A20	A19	A18	A17	A16
DATA(BANK 38000H)	0	0	0	0	0	0	1	1

RAM ARIA	FF							
Addrs BIT	A15	0	0	0	0	0	0	0
DATA(BANK 38000H)	1	0	0	0	0	0	0	0

(3)Reading the "control port"
By reading the "control port", you can check :
1)A state of Protection
2)A state of indication of bank-address
3)A state of rotary sw.

A130E0H BIT MAP

```
           :  Rotary SW   : WRITE PROTECT : RAM AREA BANK DISP
           :              :  1:ON/0:OFF   : 1:DISP/0:MASK
F  E  D  C : B  A  9  8   : 7  6  5  4    : 3  2  1  0
           :              :               : :  :  :  :
           :              :               : :  :  :  RAM0
           :              :               : :  :  RAM1
           :              :               : :  RAM2
           :              :               : RAM3
```

⑥DIP SWITCH

	ON	OFF
1	2MROM	1MROM
2	Not JEDEC	JEDEC
3	Not used	
4	Not used	

Example:

		1	2
NEC	27C1000 ‥	OFF	ON
	27C1001 ‥	OFF	OFF
	27C2001 ‥	ON	OFF

--- END ---

Genesis

1991/2/20

ROM BOARD SPECIFICATIONS

Ⅰ.4M ROM BOARD

Basic construction of ROM size
- Four 1M EP-ROMs (4M bit)

However, the jumpers on the back side can change the setting.
- Four 2M EP-ROM (8M bit)

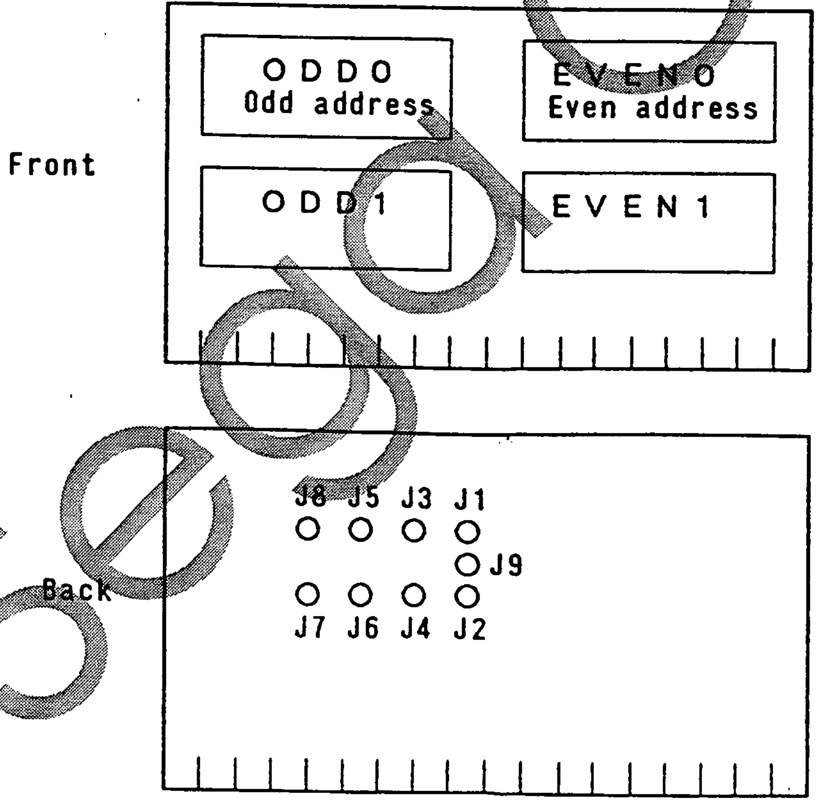
Basic construction of ROM type
- The ROM type is non-JEDEC (27C1000 for Fujitsu)

However, the jumpers on the back side can change the setting.
- The ROM type is JEDEC (27C1001 for Fujitsu)

① 1M EP-ROM of non-JEDEC type (initial setting)
J1,J3,J5 and J7 are connected. The others are all cut.

② 1M EP-ROM of JEDEC type
J1,J3,J6 and J8 are connected. The others are all cut.

③ 2M EP-ROM of JEDEC type
J2,J4,J6 and J8 are connected. The others are all cut.



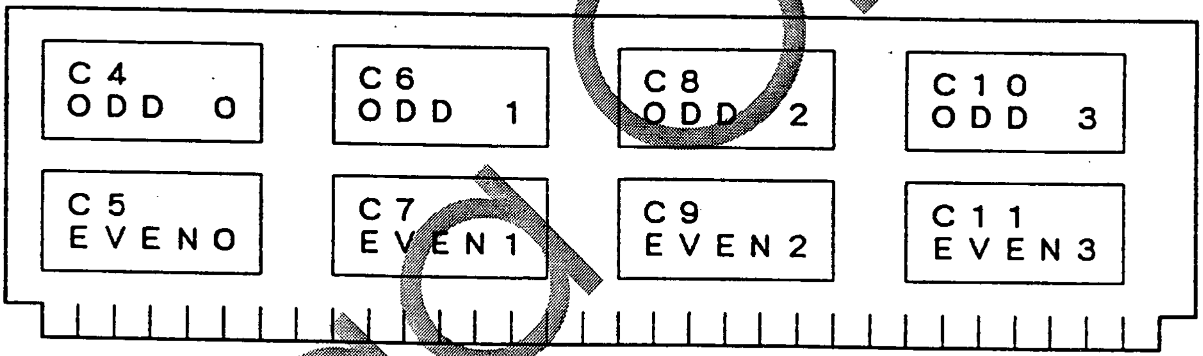
II .8M ROM board 64

Basic construction
- With the 64k bit S-RAM
- Eight 1M EP-ROMs (8M bit)
- The ROM type is non-JEDEC (27C1000 for Fujitsu)

However, the dip switch can change the setting.
- Eight 2M EP-ROM (16M bit)
- The ROM type is JEDEC (27C1001 and 27C2001 for FUjitsu)

Dip switch

1.....Backup RAM	On...Write Off..Pritect
2.....On	
3.....ROM type	On...non-JEDEC Off..JEDEC
4.....ROM capacity	On...2M EP-ROM Off..1M EP-ROM



II .8M ROM board 256

Basic construction
- With the 256k bit S-RAM
- Eight 1M EP-ROMs (8M bit)
- The ROM type is non-JEDEC (27C1000 for Fujitsu)

However, the switch can change the setting.
- The ROM type is JEDEC.

Note:Unlike the 8M board 64, the 2M EP-ROM cannot be used.

Switch

On....JEDEC Off....non-JEDEC

○ The configuration of the ROM on the board is the same as that of the 8M ROM board 64.

--- END ---

1991/6/14

Master system ROM board specifications

Dip switch

I . 1M EP-ROM (non-JEDEC type only)
1 and 2..... On
The others are all off.

II . 512K EP-ROM
3 .... On
The others are all off.

--------- END ----------

The following is information for the Super Mega Drive development station. The NMI (green) button on the front panel must be configured before it will work.

This document contains information on setting up the NMI button.

Explanation for Dip Switch #4.

I. Board Setting

Two settings exist: MEGA DRIVE setting and SUPER TARGET setting. They are switched by DIP.SW4.

A) setting for MEGA DRIVE (DIP.SW4=OFF)
Functions as a MEGA DRIVE

B) setting for SUPER TARGET (DIP.SW4=ON)
Two modes exist: the EMURATE mode and the MONITOR mode.
- EMURATE mode
A GAME_program for MEGA_DRIVE is being executed.
- MONITOR mode
A GAME_program for MEGA_DRIVE is stopped and the debugging program put in the shadow ROM is executed.

When power is on, it is in the MONITOR mode.

To turn into the EMURATE mode, execute the following instructions:

```
nop
nop
move.w  SR,-(A7)
move.l  PC,-(A7)
move.w  D7,$A81000
rte
```

Above small program is very important, so type it exactly like this and execute it.

When EMURATE mode, those functions become available:
- break functions necessary as ICE:
  1) MEMORY SIZE.break function
  2) WRITE PROTECT.break function
  3) TIME OUT.break function
  4) SWITCH.break function (push SW3 "BREAK")
  5) ADDRESS CHECK.break function
  6) HARD BREAK.break function

The mode is changed to the MONITOR mode when any break happens.

When MONITOR mode,
- I/O etc. for development can be used.
- HISTORY function RAM can be READ/WRITE.

II. Extended I/O function (when MONITOR mode)

memory map control (memctl):
mapping changes (SRAM mapping address can be changeable)

```
                    this bit is important!
$A82001            (write only)
          7 6 5 4 3 2 1 0
          | | | | | | | +---map_change   switch "ctrg" and "fdd" (valid when bit3=0)
          | | | | | | |         bit0=1 -- ctrg -> (fdd), fdd -> (ctrg)
          | | | | | | |         bit0=0 -- ctrg -> ctrg, fdd -> fdd
          | | | | | | |
          | | | | | +-+-----memory map control
          | | | | |
          | | | | +---------valid when bit3=1.  switch "ctrg" and "fdd" depending on
          | | | |               pin.B32 of cartridge. (when pin.B32=LOW it switches)
          | | | |
          | | | +-----------emulator mapping bit   temporarily swtch to EMURATE mode
          | | |                                    mapping
          | | |
          | | +-------------auto_vector no_BANK(0), auto_vector BANK(1)
          | |                   auto_vector address can be switched by BANK
          | |
          +-+---------------auto_vector BANK.page(vector bank change)
                                switch vector_address when auto_vector BANK
```

１２・ 拡張Ｉ／Ｏ機能（モニターｍｏｄｅ時）

12-a-1) memory map control (memctl)

・マッピングの変更(SRAMのマッピング番地が変えられる)

() memory map control

```
$a82001          (write_only)
         7 6 5 4 3 2 1 0                    bit0=1 ctrg->(fdd).fdd->(ctrg)
         | | | | | | | +----------map_change bit0=0 ctrg->ctrg ,fdd-> fdd
         | | | | | | |           「ctrg」と「fdd」の入れ替え(bit3=0の時有効)
         | | | | | | |
         | | | | | +-+-----------memory map control
         | | | | |               マッピングの切り換え
         | | | | |
         | | | | +---------------bit3=1の時、カートリッジのpin.B32の状態で「ctrg」と「fdd」を入れ替え.
         | | | |                 カートリッジのpin.B32=LOWの状態で[ctrg]と[fdc]の入れ替え(bit0は
         | | | |
         | | | +-----------------emulator mapping bit
         | | |                   MONITOR.mode内で、一時的にEMURATE.modeのマッピングにする
         | | |                    ＊以上「12-a-2) memory map control」を参照
         | | |
         | | +-------------------auto_vector no_BANK(0),auto_vector BANK(1)
         | |                     auto_vectorの番地をBANKで切り換えられる
         | |
         +-+---------------------auto_vector BANK.page(vecter bank change)
                                 auto_vector BANK時のvector_addressの切り換え
```

~~＊以上「12-a-3) memory map control」を参照~~

~~refer to.~~

```
           BOOT
$800000 +-------+
        | rom   |
        |       |
$880000 +-------+
        | bram  |(back_up)
$8c0000 +-------+
        |       |
$900000 +-------+
        | ram   |(emu_ram)
        |       |
        |       |
        |       |
$a00000 +-------+
```

cart	:	cartridge
fdd	:	floppy disk unit
BOOT	:	boot_rom.emulation_ram & back_up_ram
ram	:	emulation_ram(8Mbits)
bram	:	back_up_ram(2Mbits)
ex cart	:	cartridge slot [cart] line
work	:	work_ram (extnded)
rom	:	boot_rom

12-a-2) memory map control (memctl)

```
                        (( memory map control ))
+----------------++-----------------------------------+-----------------------------------+
|   hard mode    ||            emulate                |              monitor              |
|                ||(if bit4=1 in monitor_mode)        |                                   |
+----------------++--------+--------+--------+--------+--------+--------+--------+--------+
|   bit 2,1      || 0  0   | 0  1   | 1  0   | 1  1   | 0  0   | 0  1   | 1  0   | 1  1   |
|  ($a82001)     ||        |        |        |        |        |        |        |        |
     $000000      +--------+--------+--------+--------+--------+--------+--------+--------+
                  | boot   | ram    |        |        | boot   | boot   | boot   | ram    |
                  |        +--------+        |        |        |        |        +--------+
                  |        |        | cart   | cart   |        |        |        |        |
     $200000      +--------+        |(fdd)   |(fdd)   +--------+--------+--------+        |
                  |        |        |        |        |        |        |        |        |
                  |        |        |        |        |        |        |        |        |
     $400000      +--------+--------+--------+--------+--------+--------+--------+--------+
                  |        |        | ram    |        |        |        |        |        |
                  |        | fdd    +--------+ fdd    |        |        |        |        |
                  |        |(cart)  |        |(cart)  |        |        |        |        |
                  |        |        |        |        |        |        |        |        |
     $800000      +--------+--------+--------+--------+--------+--------+--------+--------+
                  | boot   | boot   | boot   | ram    | boot   | boot   | boot   | ram    |
                  |        |        |        +--------+        |        |        +--------+
     $a00000      +--------+--------+--------+--------+--------+--------+--------+--------+
                  |        |        |        |        |        |        |        |        |
     $c00000      +--------+--------+--------+--------+--------+--------+--------+--------+
                  |        |        |        |        |        |        |        |        |
     $e00000      |        |        |        |        |        |        |        |        |
     $fc0000      |        |        |        |        +--------+--------+--------+--------+
     $ff0000      |--------+--------+--------+--------+ work   | work   | work   | work   |
                  +--------+--------+--------+--------+--------+--------+--------+--------+
```

MEGA DRIVE設定の時（DIP.SW4=OFF）

mapping

・通常のMEGA DRIVEのマッピング＋開発用WORK($FC0000-FEFFF)

for develop

・開発用I/Oアドレス

for develop address

SUPER TARGET設定（DIP.SW4=ON）

a）EMURATE.modeの状態

・EMURATE.modeのマッピング

b）MONITOR.modeの状態

mapping

・MONITOR.modeのマッピング＋開発用WORK($FC0000-FEFFF)

・開発用I/Oアドレス

１２。拡張Ｉ／Ｏ機能（モニターｍｏｄｅ時）

12-C) Change to EMULATE (MONITOR)

・エミュレーターmodeに移る。

```
() monitor

$a81001          (write)
        * * * * * * * *---------change to EMULATE

$a81001 (read)
        7 6 5 4 3 2 1 0
        | | | | | | | +---------in bgack
        | | | | | | +-----------in monitor
        | | | | | +-------------by write protect
        | | | | +---------------by 232c_hunt
        | | | +-----------------by dtack time_out
        | | +-------------------by break_chip
        | +---------------------by address_checker
        +-----------------------by break_switch

                      * all_bit active low
```