Mega-CD BIOS Manual

MEGA-CD BIOS MANUAL

Sega Ozisoft

SEGA ENTERPRISES, LTD.

Ver 2.00 Feb. 24-'92

CONFIDENTIAL #161 PROPERTY OF SEGA

Sega Ozisoft

CONTENTS

1. CD SYSTEM CONTROL PROCESS SUMMARY ____ 1
   - 1-1 Hardware Summary ____ 1
   - 1-2 Process Summary ____ 2
   - 1-3 CD System Control Chart ____ 3
   - 1-4 CD System Memory Map ____ 4
2. BIOS CALL SUMMARY ____ 5
   - 2-1 BIOS Call Functions ____ 5
   - 2-2 BIOS Call List ____ 7
   - 2-3 BIOS Call Type Chart ____ 9
   - 2-4 Drive Control System Process Chart ____ 10
3. BIOS CALL REFERENCE ____ 11
   - 3-D Drive (Mechanism) ____ 12
   - 3-M CD-DA (Music) ____ 13
   - 3-R CD-ROM (Data) ____ 18
   - 3-B CD-BIOS (Other Functions) ____ 20
   - 3-F FADER (Fader) ____ 23
   - 3-C CDC (Decoder) ____ 24
   - 3-S SCD (Subcode) ____ 26
   - 3-L LED (LED Display) ____ 29
4. SYSTEM BOOTSTRAP SEQUENCE ____ 30
   - 4-1 CD Boot System Mode 1 (Boot from cartridge) ____ 30
   - 4-2 CD Boot System Mode 2 (Boot from CD-DISC) ____ 30
   - 4-3 Booting Application Programs ____ 31
5. JUMP TABLE AND USER CALLS ____ 33
   - 5-1 Structure of the Jump Table ____ 33
   - 5-2 BIOS Call Sequence ____ 33
   - 5-3 User Call Sequence ____ 34
   - 5-4 Exception Processing Sequence ____ 35
6. BOOT SYSTEM ____ 36
   - 6-1 CD-Boot ____ 36
7. BACK-UP RAM ____ 38
   - 7-1 Back-up RAM ____ 38

segaOzisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

Sega Ozisoft

1. CD SYSTEM CONTROL PROCESS SUMMARY

The CD system control process operates on top of the SUB-CPU. It mediates between direct hardware control processes and user processes.

1-1 Hardware Summary (ref. 1-3)

- **[CDD] Compact Disk Driver**

  Hardware to reproduce music, data.
  A 4 bit CPU is dedicated exclusively to controlling hardware.

- **[CDC] CD Data Controller**

  Chip to output data after standard CD-ROM error correction has been made.
  Contains 16K bytes buffer memory which can contain up to 5 frames of decoded data.

- **[SCD] SubCoDe (Functions contained within the MEGA-CD Gate Array)**

  Functions to read supplementary data (Subcode) contained in the CD.
  Contains 128 bytes of buffer memory in MEGA-CD Gate Array.

- **[FDR] FaDeR**

  Controls the volume of music data (CD-DA) being directly output by the CD drive.

- **[LED] Light Emitting Diode**

  Functions to show the status of the CD drive.
  Consists of one green and one red LED.

- **[Back-up RAM]**

  Battery backed back-up RAM
  Holds up to 8K bytes of information.

- **[SDC]**

  PCM sound generator chip.

segaozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

1-2 Process Summary (ref. 1-3)

- **Initial Process**

  This operation is initiated when the CPU is reset. Initializes required hardware and control processes.
  Requires over 100 msec while interrupts are disabled.

- **System Control Process**

  Controls the execution of system and user processes.
  Manages interrupt vectors and BIOS call entries.

- **CD-Boot Process**

  Loads programs from the CD and boots the system.

- **CD-BIOS Control Process**

  Operates the control processes of hardware according to the commands made by the application.

- **CDD Control Process**

  Controls the CD drive hardware.

- **CDC Control Process**

  Error correction is made to data output from the CD drive.

- **SCD Control Process**

  Reads out the Subcode data which is output from the CD drive and stored in the MEGA-CD Gate Array buffer. Error checking and corrections are performed.

- **FDR Control Process**

  Controls the audio volume of direct music data output from the CD drive.

- **LED Control Process**

  Controls the on/off state of the CD drive's status LED's.

- **Back-up RAM I/O Process**

  In order to allow the partitioned usage of the RAM by multiple applications, a common I/O process is used for input/output.

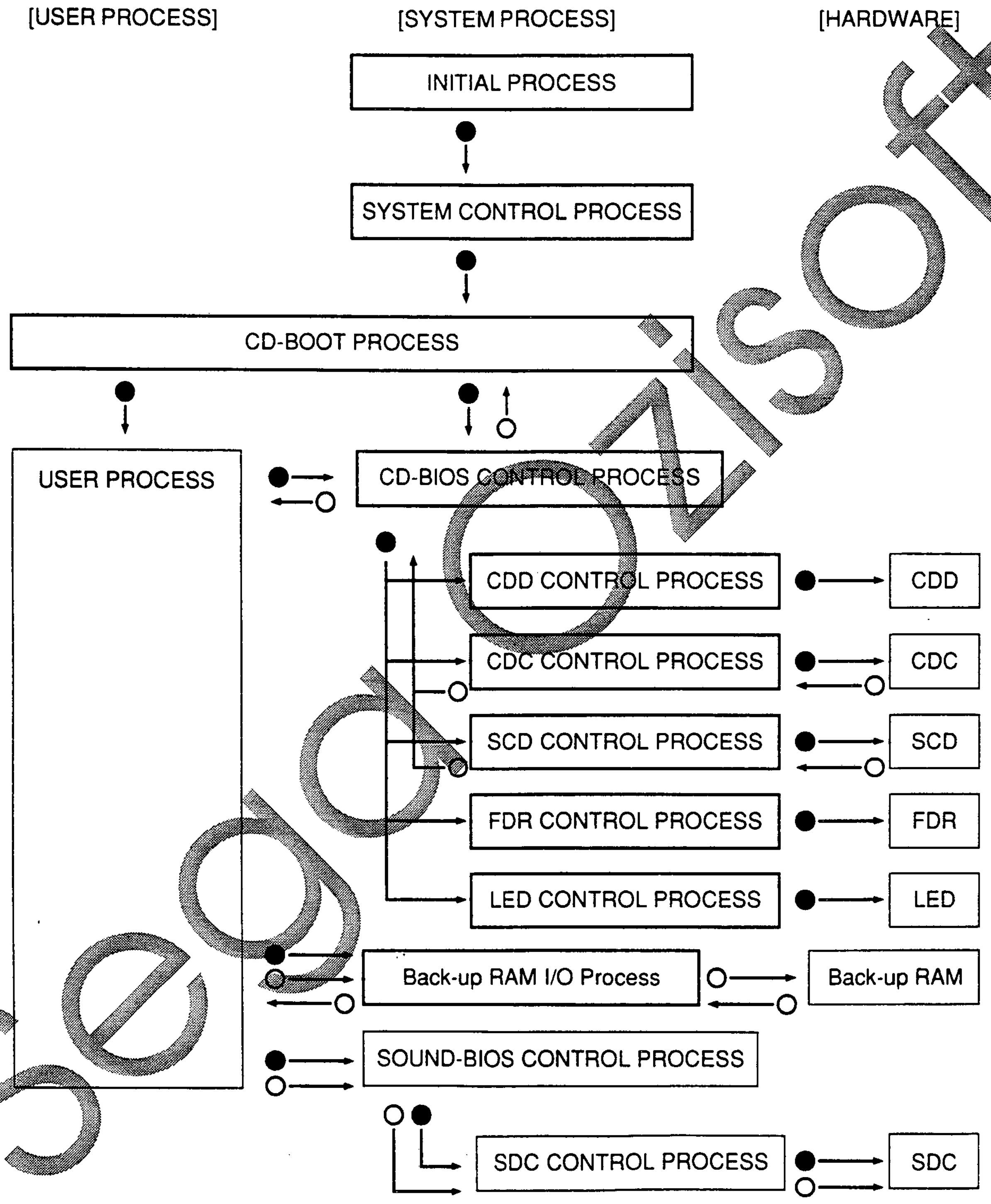
- **User Process**

  BIOS calls are made according to the needs of the application.

CONFIDENTIAL #161 PROPERTY OF SEGA

Sega Ozisoft

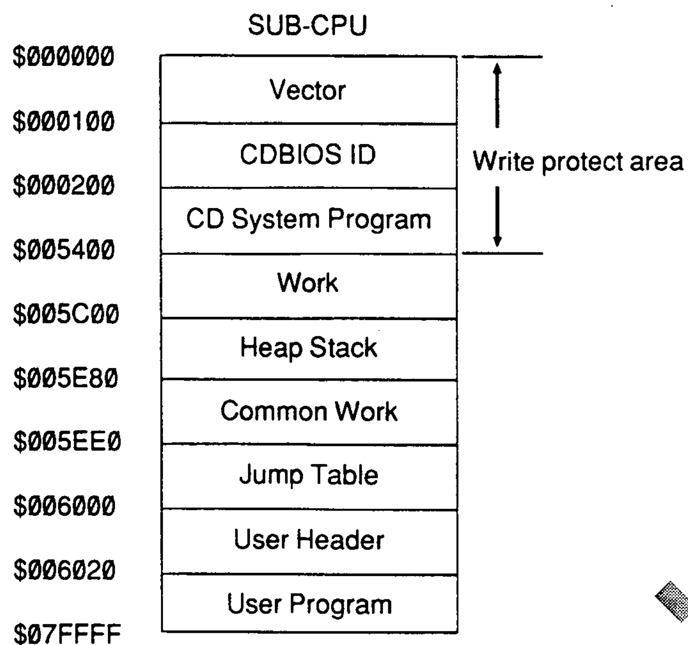
1-3 CD System Control Chart



segaXzisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

1-4 CD System Memory Map



Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

2. BIOS CALL SUMMARY

2-1 BIOS Call Functions

● Purpose

The programmer issues commands to the program (BIOS) that controls and manages the operations of hardware necessary to carry out tasks such as playing music or reading data with the CD drive. In addition, there are programs which can manage back-up RAM and boot from the CD disc.

● Caution

After the initialization process,
    level 2 (software V-INT) and
    level 4 (CDD) interrupts become enabled.
After that, depending on the CD-ROM related BIOS calls,
    level 5 (CDC) and
    level 6 (SCD) interrupts either become enabled or disabled.

If these interrupts are disabled, the BIOS cannot operate normally. Therefore, when dealing with the interrupt mask control port, never change the enable status of the levels mentioned above. (Use bit operation commands.)

In addition, the interrupt disable status created by changing the SR register will not be problem for short time periods (on the microsecond order). However, when extended for longer time periods (on the millisecond order), operations such as reading data will run into difficulties.

On the other hand, never set an interrupt at lower level than the current level. After changing the interrupt level, it is necessary to return the level to the previous position.

The CD-BIOS is managed within level 4 interrupt management. Therefore, BIOS calls should not be made while interrupt management above level 4 is in effect.(Interrupts above level 4 are CD system exclusive. Access is prohibited to the user.)

BIOS, LED, FADER, SCD, and CDC related calls are executed immediately. However, for DRIVE, CD-DA, and CD-ROM related calls, there is a lag time between the receipt of the command and the execution of the command.(This is managed by a level 4 interrupts with 13.3 msec interval.)

It is necessary to design on the application level in order to handle power failures during the execution of back-up RAM related functions. This is especially critical when directories are being changed, a power failure can destroy data from other applications. In addition, we encourage the use of memory protect functions. By using these functions, it may be possible to recover partly damaged data. Design the application software to protect the memory when not using the memory protect functions.

LED, Subcode, and boot system related calls should not normally be used on the application level.

CONFIDENTIAL #161 PROPERTY OF SEGA

SegaXtreme/Zisoft

● Call Method

A parameter selected by a command is set in a register and then the BIOS call entry is called.

(1) Parameters important to several functions are set up in a table, and the parameter table's header address is set in register a0.l. In addition, parameters necessary for the functions are set in the other registers.

(2) The function numbers are set register d0.w.

(3) The CD-BIOS entry address is called.
Registers other than d0/d1/a0/a1 are stored if they are not utilized as return values.
The entry addresses and the function numbers are provided by an "include" file.
(For details regarding each function, refer to the BIOS call reference.)

● Example

```
parameter_table_1:
        dw          PARAMETER_10, PARAMETER_11...
bios_call_example1:
        lea.l       parameter_table_1 (pc), a0
        move.w      #FUNCTION_NUMBER_1, d0
        jsr         bios_entry_address
        rts
bios_call_example_2:
        move.w      #PARAMETER_20, d1
        move.w      #PARAMETER_21, d2
        move.w      #FUNCTION_NUMBER_2, d0
        jsr         bios_entry_address
        rts
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

2-2 BIOS Call List

Abbreviation	— Summary
DRVINIT	— Close loading tray and read TOC.
DRVOPEN	— Open loading tray.
MSCSTOP	— Stop music playing.
MSCPLAY	— Play music from designated song number.
MSCPLAY1	— Play music from designated song number once.
MSCPLAYR	— Repeat playing of designated song number.
MSCPLAYT	— Play music from designated song number at designated time.
MSCSEEK	— Stop at the beginning of designated song number.
MSCSEEK1	— Stop at the beginning of designated song number and play the song only once.
MSCSEEKT	— Stop at a designated time.
MSCPAUSEON	— Pause music playing temporarily.
MSCPAUSEOFF	— Clear pause and restart music playing.
MSCSCANFF	— Play music fast-forward.
MSCSCANFR	— Play music fast-reverse.
MSCSCANOFF	— Cancel high speed music playing and restart music playing.
ROMREAD	— Begin data read from designated logical sector.
ROMREADN	— Read data a designated number of sectors beginning at a designated logical sector.
ROMREADE	— Read data between two designated logical sectors.
ROMSEEK	— Stop at designated logical sector.
ROMPAUSEON	— Pause data read.
ROMPAUSEOFF	— Clear pause and restart data read.
CDBCHK	— Check to see if request command has been received.
CDBSTAT	— Read status.
CDBTOCREAD	— Read TOC data.
CDBTOCWRITE	— Write data to TOC table.
CDBPAUSE	— Determine when the pause mode will change to stand-by mode.
FDRSET	— Set volume.
FDRCHG	— Change value at specified speed.
CDCSTART	— Begin data read from current logical sector.
CDCSTOP	— Stop data read.
CDCSTAT	— Check to see if data is ready.
CDCREAD	— Prepare for data read.
CDCTRN	— Read data with SUB-CPU.
CDCACK	— End data read.
SCDINIT	— Initialize for Subcode read.
SCDSTART	— Start reading Subcode.
SCDSTOP	— Stop Subcode read.
SCDSTAT	— Check to see if Subcode is ready.
SCDREAD	— Read Subcode.
SCDPQ	— Get P,Q code from Subcode.
SCDPQL	— Get last P,Q code from Subcode.
LEDSET	— Set LED mode.

Sega 2 Soft

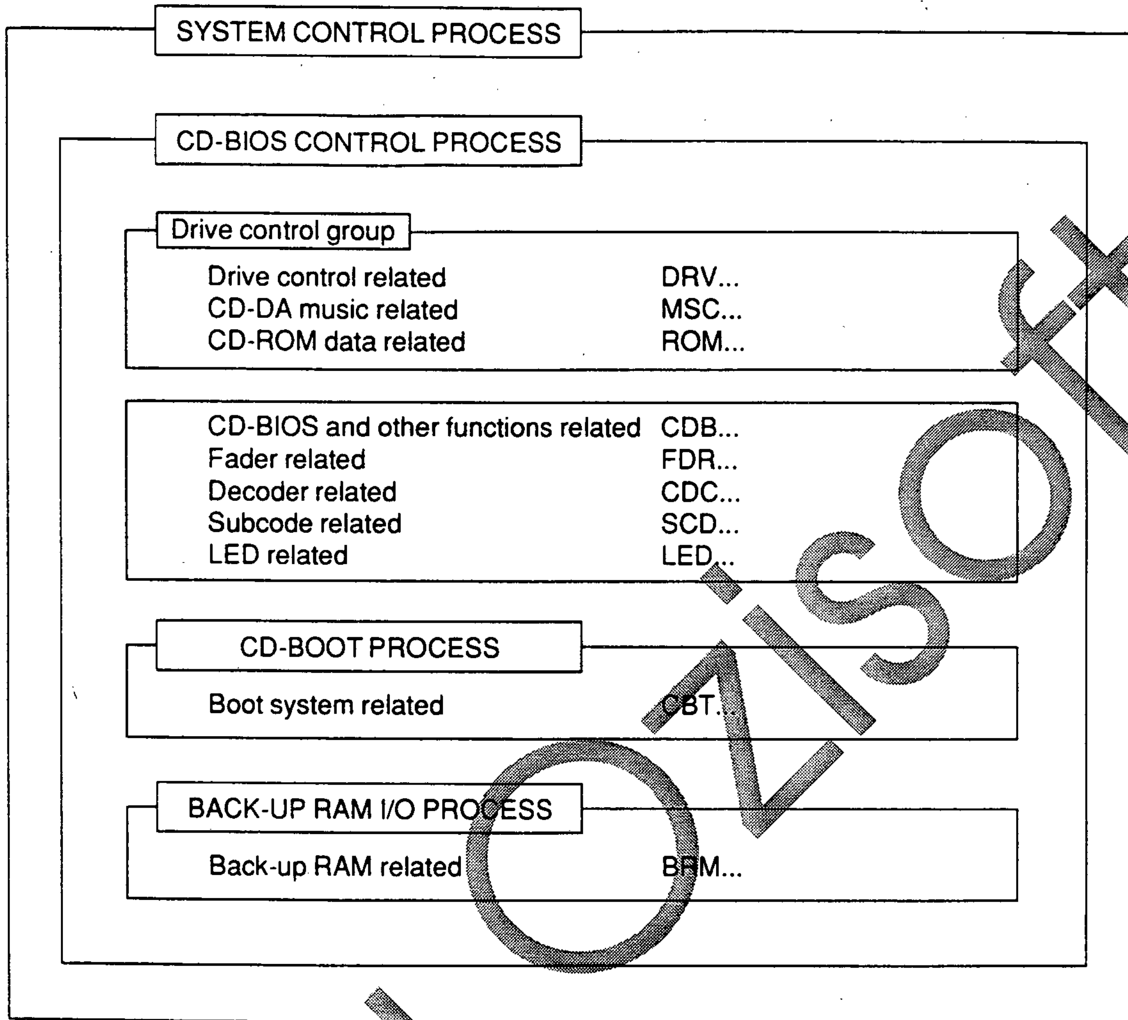
CONFIDENTIAL #161 PROPERTY OF SEGA

CBTINIT	— Initialize the boot system. Cancel booting.
CBTINT	— Call the routine to manage interrupts.
CBTOPENDISC	— Request to open the loading tray.
CBTOPENSTAT	— Check the completion of the request to open the loading tray.
CBTCHKDISC	— Request to check whether a boot can be done or not.
CBTCHKSTAT	— Check boot completion and return disc type.
BRMINT	— Prepare for read/write of back-up RAM.
BRMSTAT	— Return usage status of back-up RAM.
BRMSERCH	— Get information from back-up RAM's control file.
BRMREAD	— Read data from back-up RAM.
BRMWRITE	— Write data to back-up RAM.
BRMDEL	— Erase back-up RAM data.
BRMFROMAT	— Format
BRMDIR	— Get a directory.
BRMVERIFY	— Check the data written to the back-up RAM.

Sega Ozisoft

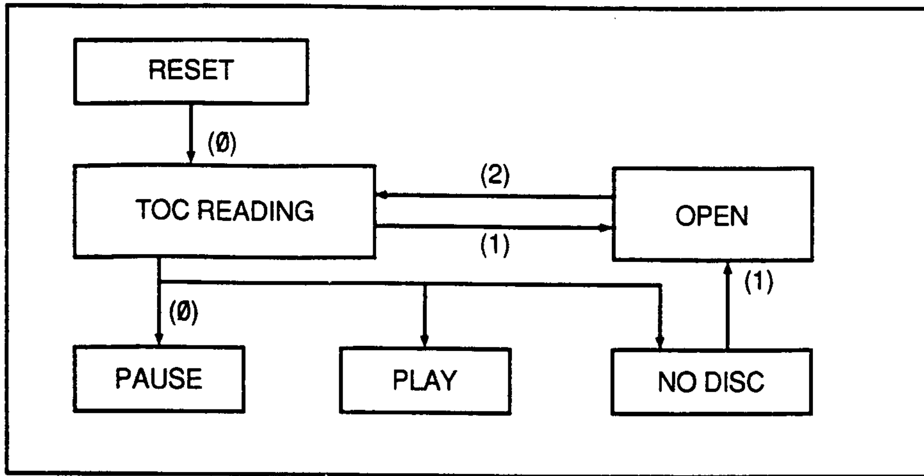
CONFIDENTIAL #161 PROPERTY OF SEGA

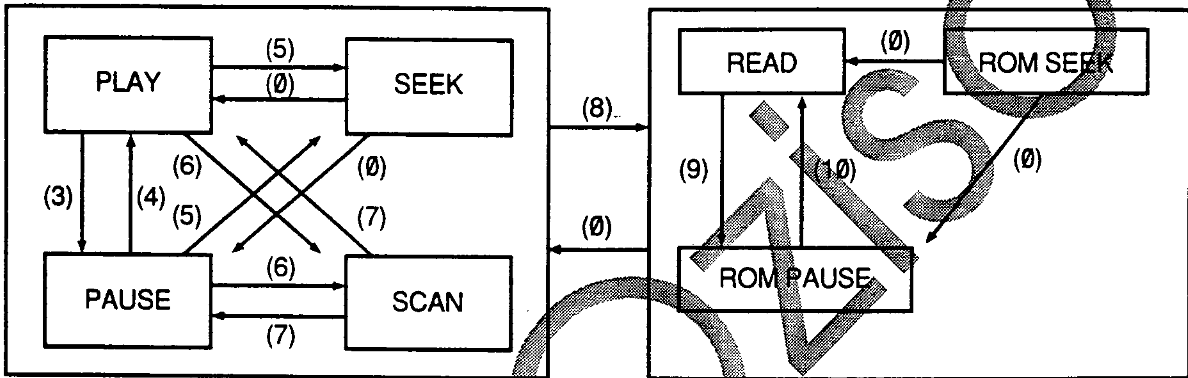
2-3 BIOS Call Type Chart

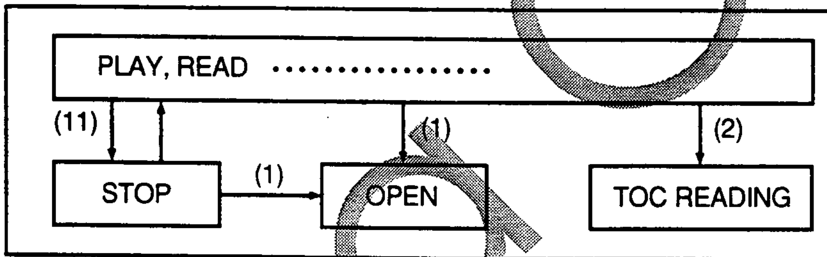


segaOzisoft

2-4 Drive Control System Process Chart







(Ø) — AUTO
(1) — DRVOPEN
(2) — DRVINIT
(3) — MSPAUSEON
(4) — MSCPAUSEOFF
(5) — MSCPLAYx, MSCSEEKx
(6) — MSCSCANFF, MSCSCANFR
(7) — MSCSCANOFF
(8) — ROMREADx, ROMSEEK
(9) — ROMPAUSEON
(10) — ROMPAUSEOFF
(11) — MSCSTOP

CONFIDENTIAL #161 PROPERTY OF SEGA

segaoziso ft

3. BIOS CALL REFERENCE

● How to read this reference:

Abbreviation **Name**

ENTRY:	Entry address-	Address to call.
IN:	Function number-	The abbreviation of the function is defined. Other parameters to be set when called. Contents of the parameter table.
→	Contents of the parameter table	
OUT:	Return value.	
BREAK:	Non-stored register.	
FUNC:	Explanation of function.	
NOTE:	Notes.	
BUG:	Known bugs.	
SAMPLE:	Examples.	

● Note

(HEX):	Hexadecimal values	($00, $01...$09, $0a...$0f, $10...)
(BCD):	BCD values	($00, $01...$09, $10...$15, $16...)
cc/cs:	carry clear/carry set	

The constants found this reference are defined in the include file CDBIOS.I.

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

3-D Drive (Mechanism)

DRVINIT — Initialize drive

ENTRY: _cdbios
IN: dØ.w function number (DRVINIT)
aØ.l table address
→ startTNO.b The first track number where the TOC data in the TOC resides. Normally "$Ø1" (hex).
endTNO.b Last track number. Normally, if "$ff" (hex) is designated, all of the tracks on the disk will be read to the last track.
OUT: none
BREAK: dØ/d1/aØ/a1
FUNC: Reads the designated TOC after the loading tray is closed. A 2 second pause occurs after the TOC read. If bit 7 of startTNO is set, the first song is automatically played after the TOC read. If there is no disk, it waits until DRVOPEN is requested.
NOTE: At a minimum, more than one track of TOC data must be read.
SAMPLE:

```
table:
        db          $Ø1, $ff
subroutine:
        lea.l       table (pc), aØ
        move.w      #DRVINIT, dØ
        jsr         _cdbios
        rts
```

DRVOPEN — Open drive

ENTRY: _cdbios
IN: dØ.w function number (DRVOPEN)
OUT: None
BREAK: dØ/d1/aØ/a1
FUNC: Opens loading tray.
NOTE:
SAMPLE:

```
        move.w      #DRVOPEN, dØ
        jsr         _cdbios
```

Segaozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

3-M CD-DA (Music)

MSCSTOP — Stop music

ENTRY: _cdbios
IN: d0.w function number (MSCSTOP)
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Stop music playing.
NOTE:
SAMPLE:

```
        move.w      #MSCSTOP, d0
        jsr         _cdbios
```

MSCPLAY — Play music

ENTRY: _cdbios
IN: d0.w function number (MSCPLAY)
a0.l table address
→ TNO.w Song number to play.
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Plays music from designated song number.
NOTE:
SAMPLE:

```
table:
        dw          $0002
subroutine:
        lea.l       table (pc), a0
        move.w      #MSCPLAY, d0
        jsr         _cdbios
        rts
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

MSCPLAY1 — Play music 1

ENTRY: _cdbios
IN: d0.w function number (MSPLAY1)
a0.l table address
→ TNO.w Song number play
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Replays the music of the designated song number once.
NOTE:
SAMPLE:

```
table:
        dw          $0002
subroutine:
        lea.l       table (pc), a0
        move.w      #MSCPLAY1, d0
        jsr         _cdbios
        rts
```

MSCPLAYR — Repeat music play

ENTRY: _cdbios
IN: d0.w function number (MSCPLAYR)
a0.l table address
→ TNO.w Song number to play
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Repeats the music playing of the designated song number.
NOTE:
SAMPLE:

```
table:
        dw          $0002
subroutine:
        lea.l       table (pc), a0
        move.w      #MSCPLAYR, d0
        jsr         _cdbios
        rts
```

MSCPLALYT — Music playing time

ENTRY: _cdbios
IN: d0.w function number (MSCPLAYT)
a0.l table address
→ time.l Time to begin playing mm:ss:ff:00 (BCD)
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Plays music beginning at the designated time.
NOTE:
SAMPLE:

```
table:
        d1          $00020000
subroutine:
        lea.l       table (pc), a0
        move.w      #MSPLAYT, d0
        jsr         _cdbios
        rts
```


CONFIDENTIAL #161 PROPERTY OF SEGA

Sega Ozisoft

MSCSEEK

Seek music

ENTRY: _cdbios
IN: d0.w function number (MSCSEEK)
a0.l table address
→ TNO.w Song number to seek
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Goes to the beginning of the designated song number and pauses.
NOTE:
SAMPLE:

```
table:
        dw          $0002
subroutine:
        lea.l       table (pc), a0
        move.w      #MSCSEEK, d0
        jsr         _cdbios
        rts
```

MSCSEEK1

Seek one song

ENTRY: _cdbios
IN: d0.w function number (MSCSEEK1)
a0.l table address
→ TNO.W Song number to seek.
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Goes to the beginning of the designated song number and pausing. After pausing, plays the song only once.
NOTE:
SAMPLE:

```
table:
        dw          $0002
subroutine:
        lea.l       table (pc), a0
        move.w      #MSCSEEK1, d0
        jsr         _cdbios
        rts
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

MSCSEEKT — Seek music time

ENTRY: _cdbios
IN: d0.w function number (MSCSEEKT)
a0.l table address
→ time.l Time to seek mm:ss:ff:00 (BCD)
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Stops at the designated time.
NOTE:
SAMPLE:

```
table:
        d1          $00020000
subroutine:
        lea.l       table (pc), a0
        move.w      #MSCSEEKT, d0
        jsr         _cdbios
        rts
```

MSCPAUSEON — Music pause on

ENTRY: _cdbios
IN: d0.w function number (MSCPAUSEON)
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Stops playing of music and pauses.
NOTE: Since an extended pause is harmful to the drive, after a set period of time, the drive will stop. (The same applies when an automatic pause occurs).
SAMPLE:

```
        move.w      #MSCPAUSEON, d0
        jsr         _cdbios
```

MSCPAUSEOFF — Music pause off

ENTRY: _cdbios
IN: d0.w function number (MSCPAUSEOFF)
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Clears pause and restarts playing of music.
NOTE: When the music playing is stopped and while in stand-by mode, this subroutine seeks the pause time and starts playing the music again.
SAMPLE:

```
        move.w      #MSCPAUSEOFF, d0
        jsr         _cdbios
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

MSCSCANFF — Music forward scan

ENTRY: _cdbios
IN: dØ.w function number (MSCSCANFF)
OUT: None
BREAK: dØ/d1/aØ/a1
FUNC: Plays fast forward scan of music.
NOTE:
SAMPLE:

```
        move.w      #MSCSCANFF, dØ
        jsr         _cdbios
```

MSCSCANFR — Music reverse scan

ENTRY: _cdbios
IN: dØ.w function number (MSCSCANFR)
OUT: None
BREAK: dØ/d1/aØ/a1
FUNC: Play fast reverse scan of music
NOTE:
SAMPLE:

```
        move.w      #MSCSCANFR, dØ
        jsr         _cdbios
```

MSCSCANOFF — Music scan off

ENTRY: _cdbios
IN: dØ.w function number (MSCSCANOFF)
OUT: None
BREAK: dØ/d1/aØ/a1
FUNC: Stop fast scan playing of music and returns to normal playing. If scan was initiated from a pause, the drive will pause.
NOTE:
SAMPLE:

```
        move.w      #MSCSCANOFF, dØ
        jsr         _cdbios
```

segaozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

3-R CD-ROM (Data)

ROMREAD — Read ROM

ENTRY: _cdbios
IN: d0.w function number (ROMREAD)
a0.l table address
→ sector.l Logical sector number to begin read
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Reads data from the designated logical sector.
NOTE: Since CDCSTART is automatically executed, in order to stop the read, it is necessary to execute CDCSTOP.
SAMPLE:
※ See BIOS load sample

ROMREADN — Read ROM number

ENTRY: _cdbios
IN: d0.w function number (ROMREADN)
a0.l table address
→ sector.l Logical sector number to begin read
number.l Number of sectors to read.
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Reads the designated number of sector beginning at designated logical sector.
NOTE: CDSTART is automatically executed. After the designated number of sectors are read, the music automatically pauses, and CDCSTOP is executed.
SAMPLE:
※ See BIOS load sample

ROMREADE — End ROM read

ENTRY: _cdbios
IN: d0.w function number (ROMREADE)
a0.l table address
→ startsector.l Logical sector number to begin read
endsector.l Logical sector number where read ends
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Reads data between one designated logical sector and another designated logical sector.
NOTE: CDSTART is automatically executed. After the sectors are read, CDCSTOP is automatically executed.
SAMPLE:
※ See BIOS load sample

SegaXtreme / Zisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

ROMSEEK

ROM seek

ENTRY: _cdbios
IN: d0.w function number (ROMSEEK)
a0.l table address
→ sector.l Logical sector number to seek
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Seeks designated logical sector and pause.
NOTE:
SAMPLE:

ROMPAUSEON

ROM pause on

ENTRY: _cdbios
IN: d0.w function number (ROMPAUSEON)
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Stop data read and pause.
NOTE:
SAMPLE:

```
move.w    #ROMPAUSEON, d0
jsr       _cdbios
```

ROMPAUSEOFF

ROM pause off

ENTRY: _cdbios
IN: d0.w function number (ROMPAUSEOFF)
OUT: None
BREAK: d0/d1/a0/a1
FUNC: Clears pause and restarts data read.
NOTE:
SAMPLE:

```
move.w    #ROMPAUSEOFF, d0
jsr       _cdbios
```

segaozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

3-B CD-BIOS (Other Functions)

CDBCHK — Check CD-BIOS

ENTRY: _cdbios
IN: d0.w function number (CDBCHK)
OUT: cc/cs COMPLETE/BUSY
BREAK: d0/d1/a0/a1
FUNC: Checks to see if request commands to the CD-BIOS control (DRIVE, CD-DA, CD-ROM related) have been completed.
NOTE:
SAMPLE:

```
loop:
        move.w  #CDBCHK,d0
        jsr     _cdbios
        bcs.s   loop
```

CDBSTAT — CD-BIOS status

ENTRY: _cdbios
IN: d0.w function number (CDBSTAT)
OUT: a0.l pointer of status table (32byte)

```
→ bios_status.w   [drv | msc | 0 | rom] drive control status
  led.w           [real (0000 00GR) | mode (#0~7)] LED status
  cdd_status.20   [status | report | control | TNO
                  | absolute time (mmssffFL)
                  | relative time (mmssffFL)
                  | start track# | end track# | version | FLAGS
                  | readout start time] drive status
  volume.l        [master volume | volume] fader volume
  header.l        [min | sec | frame | mode (realtime)] data read status
```

※Contents of bios_status

```
1xxx xxxx xxxx xxxx : not ready
0100 0000 xxxx xxxx : open tray
0010 0000 xxxx xxxx : reading TOC
0001 0000 xxxx xxxx : no disk
0000 xxxx xxxx xxxx : finished reading TOC
0000 0000 xxxx xxxx : stopped
0000 0001 xxxx xxxx : playing music, data
0000 0011 xxxx xxxx : scanning music
0000 0101 xxxx xxxx : paused
0000 1000 xxxx xxxx : seeking for play
0xxx xxxx 0000 0000 : data read complete
0xxx xxxx 0000 0001 : reading data
0xxx xxxx 0000 0101 : reading data paused
0xxx xxxx 0000 1000 : seek for read
```

The status will change if the drive goes into "not ready" status after a drive control BIOS call is requested.

※Contents of led

```
BIT15~10 : 0
BIT9     : Green LED       ON (1)/OFF (0)
BIT8     : Red LED         ON (1)/OFF (0)
BIT7~0   : LED blink mode  (refer to LEDSET)
```


CONFIDENTIAL #161 PROPERTY OF SEGA

Segaxizsoft

※Contents of cdd_status (BP: Byte position)

BPØ : Status code
BP1 : Report code
BP2 : Disk control code (*1)
BP3 : Song number (hex) (*1)
BP4~7 : Absolute time (BCD) minutes, seconds, frame, flag (*1,*2)
BP8~11 : Relative time (BCD) minutes, seconds, frame, flag (*1,*2)
BP12 : First song number
BP13 : Last song number
BP14 : Drive version
BP15 : Flag (*3) BIT3: Unused (Ø)
BIT2: Ø:Music 1:Data
BIT1: Emphasis ON (1)/OFF (Ø)
BITØ: Mute ON, -12dB (1)/OFF (Ø)
BP16~19 : Start time of read out area.
(*1) Ignored when -1.
(*2) The flag is set in real time.
(*3) The flag is in the read-in area.

※Contents of volume

BIT31~16: Master volume ($Ø ~ $4ØØ)
BIT15~Ø : Volume ($vvvf › vvv: $Ø ~ $4ØØ f: Emphasis flag)

※Contents of header

Time written in the frame which precedes the data.

BREAK: dØ/d1/aØ/a1
FUNC: Reads the status of the CD-BIOS. The _cdstat address is returned in aØ.l
NOTE: _cdstat is rewritten each time this function is executed.
SAMPLE:

```
move.w  #CDBSTAT, dØ
jsr     _cdbios
```

CDBTOCREAD — CD-BIOS TOC read

ENTRY: _cdbios
IN: dØ.w function number (CDBTOCREAD)
d1.w TOC# (hex) song number to be read
OUT: dØ.l time information (low byte is TNO)
if error, low byte is ØØ
d1.b track types $ØØ : CD-DA track
$FF : CD-ROM track
BREAK: dØ/d1/aØ/a1
FUNC: Return TOC information. If a song number is requested without it being read in the TOC, an error will occur. When an error occurs, the time information will be either that of a song previously read or the time at the beginning of the disk.
NOTE: The TOC table should not be read during a TOC read.
SAMPLE:

```
move.w  #$Ø1, d1
move.w  #CDBTOCREAD, dØ
jsr     _cdbios
```


CONFIDENTIAL #161 PROPERTY OF SEGA

SegaOziSoft

CDBTOCWRITE

CD-BIOS TOC write

ENTRY: _cdbios
IN: d0.w function number (CDBTOCWRITE)
a0.l pointer to time information table (min:sec:frame:tno (bcd))
frame-bit7 0: CD-DA track 1: CD-ROM track
※ The time information table format consists of minutes, seconds, and frames represented by 3 bytes of BCD code. A one byte song number is attached to the end. The data for one song consists of these four bytes.
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Write data to TOC table.
NOTE: On disks with large amounts of TOC information, it is possible to rewrite the TOC table directly in order to shorten the TOC read time. The TOC table should not be written while a TOC read is in progress.
SAMPLE:

```
table:
      dl      $00028001, $10000002, $20000003, ENDOFTOCTBL
subroutine:
      lea.l     table (pc), a0
      move.w    #CDBTOCWRITE, d0
      jsr       _cdbios
      rts
```

CDBPAUSE

CD-BIOS pause

ENTRY: _cdbios
IN: d0.w function number (CDBPAUSE)
d1.w max. pause time (N × 1/75 sec)
OUT: none
BREAK: d0/d1/a0/a1
FUNC: determines when the pause status will change to stand-by mode.
NOTE: When -1 is set, stand-by mode is not entered except during debugging. In the released version, set this entry within the range: 4500~65534 ($1194 ~ $FFFE).

SAMPLE:

```
      move.w    #60*60*3, d1
      move.w    #CDBPAUSE, d0
      jsr       _cdbios
```

Segasoft

CONFIDENTIAL #161 PROPERTY OF SEGA

3-F FADER (Fader)

FDRSET — Set fader

ENTRY: _cdbios
IN: d0.w function number (FDRSET)
d1.w volume $0000 (small) ~ $0400 (large) : volume
$8000 (small) ~ $8400 (large) : master volume
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Sets audio volume levels. If BIT15 of volume is set, the master volume can be changed.
NOTE: It takes several milliseconds (0 ~ 13 msec) before the audio volume actually starts to change. It takes several more milliseconds (0 ~ 23 msec) before the volume reaches the designated level. Since volume cannot be set to exceed the value set by the master volume, this command is used to enable the user to adjust the master volume or to allow the program to change the volume.
SAMPLE:

```
        move.w      #$0400, d1
        move.w      #FDRSET, d0
        jsr         _cdbios
```

FDRCHG — Fader change

ENTRY: _cdbios
IN: d0.w function number (FDRCHG)
d1.l H: att. target data (0 ~ $0400) Volume level after change
L: change speed (0 ~ $0400) (step/int) Add 13.3 msec
$0001 (slow) ~ $0200 (fast), $0400 (set once)
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Change audio volume level at set speed from one volume level to another.
NOTE: Effect time varies according to volume settings.
SAMPLE:

```
        move.l      #$04000004, d1
        move.w      #FDRCHG, d0
        jsr         _cdbios
```

Segaxxx zisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

3-C CDC (Decoder)

CDCSTART — CDC start

ENTRY: _cdbios
IN: d0.w function number (CDCSTART)
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Begins data readout from current logical sector.
NOTE: The actual read begins 2 ~ 4 sectors prior to the designated sector. When stopping the read, CDCSTOP must always be executed. Read during playing, otherwise meaningless data may be read.
SAMPLE:

CDCSTOP — CDC stop

ENTRY: _cdbios
IN: d0.w function number (CDCSTOP)
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Stops data read.
NOTE: The sector currently being read is thrown out.
SAMPLE:

CDCSTAT — CDC state

ENTRY: _cdbios
IN: d0.w function number (CDCSTAT)
OUT: cc/cs OK/NULL (Prepared data present/not present)
BREAK: d0/d1/a0/a1
FUNC: Checks to see if data has been prepared.
NOTE: Buffers up to 5 frames.
SAMPLE:
※ See BIOS load sample

CDCREAD — CDC read

ENTRY: _cdbios
IN: d0.w function number (CDCREAD)
OUT: cc/d0.l OK/header time (mmssffMD)
Prepared data present minutes/seconds/frame/mode
cs NULL Not present
BREAK: d0/d1/a0/a1
FUNC: If data is prepared, prepares to read one frame worth of data. Before executing this function, make sure to set the CDC mode (device destination). Moreover, after reading the data, always execute CDCACK. When returned by cs, it is not necessary to execute CDCACK.
NOTE:
SAMPLE:
※ See BIOS load sample

CONFIDENTIAL #161 PROPERTY OF SEGA

Segaozisoft

CDCTRN — CDC transfer

ENTRY: _cdbios
IN: d0.w function number (CDCTRN)
a0.l point to destination data buffer (size: 2336 bytes)
a1.l pointer to header buffer (size: 4 bytes)
OUT: cc/cs ok/not complete
(all data transferred/data transfer not complete)
a0.l next pointer to destination data buffer
a1.l next pointer to header buffer
BREAK: d0/d1/a0/a1
FUNC: Reads one frame worth of data with the SUB-CPU.
NOTE: This function can only be called when the CDC mode (device destination) is set to the SUB-CPU read.
SAMPLE:
※ See BIOS load sample

CDCACK — CDC acknowledge

ENTRY: _cdbios
IN: d0.w function number (CDCACK)
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Ends one frame worth of data read.
NOTE: It is necessary to call this function after each frame is read.
SAMPLE:
※ See BIOS load sample

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

3-S SCD (Subcode)

SCDINIT — Subcode initialization

ENTRY: _cdbios
IN: d0.w function number (SCDINIT)
a0.l pointer to scratch RAM ($750)
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Initialization is done before reading Subcode.
NOTE:
SAMPLE:

```
        lea.l       scratchRAM, a0
        move.w      #SCDINIT, d0
        jsr         _cdbios
```

SCDSTART — Subcode start

ENTRY: _cdbios
IN: d0.w function number (SCDSTART)
d1.w Subcode processing mode (0~3)
0: _ _ _ _ _ _ _ _
1: _ _ RSTUVW
2: PQ_ _ _ _ _ _
3: PQRSTUVW
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Starts reading subcode.
NOTE:
SAMPLE:

```
        move.w      #SCDSTART, d0
        jsr         _cdbios
```

SCDSTOP — Subcode stop

ENTRY: _cdbios
IN: d0.w function number (SCDSTOP)
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Stop subcode read.
NOTE:
SAMPLE:

```
        move.w      #SCDSTOP, d0
        jsr         _cdbios
```


CONFIDENTIAL #161 PROPERTY OF SEGA

SegaOzisoft

SCDSTAT — Subcode state

ENTRY: _cdbios
IN: d0.w function number (SCDSTAT)
OUT: d0.l errqcodecrc / errpackcirc /scdflag / restrcnt
d1.l erroverrun / errpacketbufful / errqcodebufful / errpackbufful
BREAK: d0/d1/a0/a1
FUNC: Returns error status of Subcode.
NOTE:
SAMPLE:

```
        move.w      #SCDSTAT, d0
        jsr         _cdbios
```

SCDREAD — Subcode read

ENTRY: _cdbios
IN: d0.w function number (SCDREAD)
a0.l pointer to subcode buffer (size: 24 bytes)
OUT: cc/cs OK/NULL prepared data available/no data present
a0.l pointer of subcode buffer
BREAK: d0/d1/a1
FUNC: Reads R~W codes from Subcode (1 pack)
NOTE:
SAMPLE:

```
      buffer:
        ds          24
subroutine:
        lea.l       buffer, a0
        move.w      #SCDREAD, d0
        jsr         _cdbios
        bcs.s       not ready
        bcc.s       ready
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

SCDPQ — Subcode PQ

ENTRY: _cdbios
IN: dØ.w function number (SCDPQ)
a0.l pointer to Q code buffer (size: 12 bytes)
OUT: cc/cs OK/NULL
a0.l pointer to Q code buffer
BREAK: dØ/d1/a1
FUNC: Retrieves P and Q codes from the subcode.
NOTE: Ø 1 2 3 4 5 6 7 8 9 A B C

CONTROL	ADDRESS1	TNO	X	MIN	SEC	FRAME	P	AMIN	ASEC	AFRAME	CRC
CONTROL	ADDRESS2	DATA						Ø	P	AFRAME	CRC
CONTROL	ADDRESS3	DATA							P	AFRAME	CRC

CONTROL	Bit2	Ø : music	1 : data
	BitØ	Ø : normal	1 : pre-emphasis
ADDRESS	Ø	reserved	
	1	mode-1	time
	2	mode-2	catalog number
	3	mode-3	International-Standard-Recording-Code
	4~	reserved	
P		Ø : music or data	
		1 : start flag (pause)	
CRC	Ø :	no error	
	other:	error exist	

SAMPLE:

```
qbuffer:
        ds          12
subroutine:
        lea.l       qbuffer, aØ
        move.w      #SCDPQ, dØ
        jsr         _cdbios
        bcs.s       not ready
        bcc.s       ready
```

SCDPQL — Subcode PQL

ENTRY: _cdbios
IN: dØ.w function number (SCDPQL)
aØ.l pointer to Q code buffer (size: 12 bytes)
OUT: cc/cs OK/NULL
aØ.l pointer to Q code buffer
BREAK: dØ/d1/a1
FUNC: Gets the last P Q code.
NOTE:
SAMPLE:

```
qbuffer:
        ds          12
subroutine:
        lea.l       qbuffer, aØ
        move.w      #SCDPQL, dØ
        jsr         _cdbios
        bcs.s       not ready
        bcc.s       ready
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

3-L LED (LED Display)

LEDSET — LED SET

ENTRY: _cdbios
IN: d0.w function number (LEDSET)
d1.w mode number blinking mode of LED
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Sets a LED mode
When the mode number is set to 0 through 7, system control is cancelled.

mode#	Green	Red	system control
(reset	x	x	not used CD-ROM driver)
LEDREADY(0)	o	*	CD ready & no disc
LEDDISCIN(1)	o	x	CD ready & disc OK
LEDACCESS(2)	o	o	CD access
LEDSTANDBY(3)	*	x	standby mode
LEDERROR(4)	*	*	(reserved)
LEDMODE5(5)	*	o	(reserved)
LEDMODE6(6)	x	*	(reserved)
LEDMODE7(7)	x	o	(reserved)
LEDSYSTEM			return to system control

o: on x: off *: blinking

NOTE: Since the status of the LEDs show the status of CD the drive, do not use this subroutine except for debugging.

SAMPLE:

```
        move.w    #LEDREADY, d1
        move.w    #LEDSET, d0
        jsr       _cdbios
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

4. SYSTEM BOOTSTRAP SEQUENCE

4-1 CD Boot System Mode 1 (Boot from cartridge)

● MAIN-CPU

(0) Power On Boot cartridge.
(1) Initialize MEGA-DRIVE hardware.
(2) Transfer CD-SYSTEM program to SUB-CPU side.
(3) Transfer SUB-CPU application program.
(4) Reset release SUB-CPU
(5) Initialize application program.
(6) Wait for SUB-CPU application program to run.
(Wait for initialization on SUB-CPU side.)
(7) Execute application program.

● SUP-CPU

(0) Power on reset.
(1) After transferring the CD-SYSTEM program and the SUB-CPU application program, reset released by the MAIN-CPU.
(2) Initialize CD-SYSTEM program.
(3) Initialize SUB-CPU application program.
(4) Run CD-SYSTEM program. (Begin interrupt processing.)
(5) Run SUB-CPU application program.
The choice to reboot can be selected through the application. When rebooting, the same process at power on is carried out.

4-2 CD Boot System Mode 2 (Boot from CD-DISC)

※ At Power ON

● MAIN-CPU

(0) Power On. Boot from BOOT-ROM.
(1) Initialize MEGA-DRIVE hardware.
(2) Transfer CD-SYSTEM program to SUB-CPU side.
(3) Transfer CD-BOOT program to SUB-CPU.
(Initialization program (IP) and system program (SP) load program)
(4) Reset release SUB-CPU.
(5) Wait for IP to be transferred.
(6) After IP transfer is complete, execute.
(7) Wait for application program transfer.
(8) Execute application program.

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

- **SUB-CPU**

(Ø) Power on reset.
(1) After transferring the CD-SYSTEM program and the CD-BOOT program, reset is released by the MAIN-CPU.
(2) Initialize CD-SYSTEM program.
(3) Initialize CD-BOOT program.
(4) Run CD-SYSTEM program. (Begin interrupt processing.)
(5) Run CD-BOOT program. After loading IP, transfer to MAIN side.
(6) Execute SP after loading.
(7) Load application program. Also transfer to MAIN side.
(8) Execute application program.

4-3 Booting Application Programs

MAIN-CPU Side

- In CD boot system mode 1 (boot from cartridge), the application program will assume control from the boot program.
- In the case of CD boot system mode 2 (boot from CD), a subprogram header is attached to the initial program's header.

SUB-CPU Side

- In CD boot system mode 1, a user header is attached to the header of the SUB-CPU application program.
- In CD boot system mode 2, a user header is attached to the header of the CD-BOOT program. A subprogram header is attached to the system program header.

User Header Specifications

```
(Ø)  __L_TEXT:
(1)     db      'MAIN________', Ø        ; module name, flag
(2)     dw      $0100,Ø                  ; version, type
(3)     dl      Ø                        ; ptr. next module
(4)     dl      __S_TEXT+__Sdata         ; module size
(5)     dl      _start-__L_TEXT          ; start address
(6)     dl      __Sdata+__Sbss           ; work RAM size
(7)  _start:
        dw      usercallØ-_start
        dw      usercall1-_start
        dw      usercall2-_start
        dw      usercall3-_start
        dw      Ø                        ; end mark (zero)
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

(1) Module name, flag.
11 character ASCII code. The first four letters are set to "MAIN."
The flag is fixed at 0.

(2) Version

$0000~$0099:	prerelease version
$0100~ :	release version

Type : 0: normal type

(3) Link module pointer
$00000000: no link module

(4) Module size (ROM size)
Total number of bytes of data to be initialized with the program code.

(5) Start address
A relative address given by a subprogram header in order to set a jump table.

(6) Work RAM address
Total number of bytes of initialized and uninitialized data.

(7) Data table set to the jump table

_usercall0:	initialization routine
_usercall1:	main routine
_usercall2:	level 2 interrupt routine
_usercall3:	user defined routine (cannot be called from the system)

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

5. JUMP TABLE AND USER CALLS

5-1 Structure of the Jump Table

The Jump Table is the area where the CD-SYSTEM program and application program commonly access, communicate, and control.

They have the following characteristics:
- By executing CD-BIOS calls through this area, it will not be necessary to change the application program when the CD-BIOS is upgraded.
- Enables the change of processing exceptions.
- By using links such as these during execution (dynamic links), the efficiency of application program development will be increased.

The common area address is defined in the include file 'cabios.i'. By including 'cabios.i' into the source program and assembling, it will be possible to use the jump table.

5-2 BIOS Call Sequence

By setting parameters in registers and calling BIOS entries such as '_cdbios' and '_buram', it is possible to use the BIOS functions.

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

5-3 User Call Sequence

● Contents

In order to call user programs from the CD-SYSTEM program (including the boot system), four entries _usercall0,_usercall1,_usercall2, and _usercall3 have been prepared in the jump table.

(0) ,_usercall0:
An initialization routine entry. It is called prior to the enabling of level 2 interrupt processing.

(1) ,_usercall1:
A main routine entry. Sets the '_usermode' value to the d0.w register and gets called. When returning, it waits for level 2 interrupt processing to finish and gets called again.

(2) ,_usercall2:
A level 2 interrupt routine entry. Gets called after CD-SYSTEM processing.

(3) ,_usercall3:
An user defined routine entry. Does not get called by the system.

● Application Methods

It is possible to either use a sub program header (refer to 4-3) or to rewrite directly. One jump table entry has 6 bytes. By writing code here, it is possible to control the execution of programs.

The codes which the CD-SYSTEM can write here are 'jmp', 'rts', and 'rte'.

● EXAMPLE

_level3: jmp $10000

$4ef9	0,1
$0001	2,3
$0000	4,5

CAUTION When rewriting directly, use one of the two methods below.
(1) Disable interrupts while writing the 6 bytes.
(2) Rewrite in the order of addresses and jump codes.

In cases where a "rte" is written, the next 2~5 bytes may have dubious addresses written in them.

CONFIDENTIAL #161 PROPERTY OF SEGA

Sega Ozisoft

5-4 Exception Processing Sequence

The exception processing vector area ($Ø~$FF) is in a write protected area. Therefore, it is set up so that jumps are made to the jump table area. The jump destination after initialization is a return. Or alternatively, it is set in the error processing routine (Error processing only goes through the reset process.) By rewriting the jump destination, it is possible to change exception processing. However, the following entries are exclusive to the CD-SYSTEM; therefore, they must not be changed.

_level2, _level4, _level5, _level6

When a fatal error occurs, call SP by doing a hard initialization. Let the MAIN-CPU know about the fatal error by means of SP.

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

6. BOOT SYSTEM

6-1 CD-Boot (Boot System)

CBTINIT — CD Boot Initial

ENTRY: _cdboot
IN: d0.w function number (CBTINIT)
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Initializes the boot system. Cancel booting.
NOTE:
SAMPLE:

```
        move.w    #CBTINIT, d0
        jsr       _cdboot
```

CBTINT — CD Boot Int

ENTRY: _cdboot
IN: d0.w function number (CBTINT)
OUT: none
BREAK: d0/d1/a0/a1
FUNC: Calls the routine to manage interrupts.
NOTE: Call this entry every 16.6 msec.
SAMPLE:

```
        move.w    #CBTINT, d0
        jsr       _cdboot
```

CBTOPENDISC — CD Boot Open Disc

ENTRY: _cdboot
IN: d0.w function number (CDTOPENDISC)
OUT: cc/cs OK/BUSY
BREAK: d0/d1/a0/a1
FUNC: Requests to open the loading tray.
NOTE: Uses DRVOPEN of _cdbios. Returns BUSY only when interrupts are being handled by CBTINT.
SAMPLE:

```
        move.w    #CBTOPENDISC, d0
        jsr       _cdboot
```


Sega Ozisoft
CONFIDENTIAL #161 PROPERTY OF SEGA

CBTOPENSTAT — CD Boot Open Status

ENTRY: _cdboot
IN: dØ.w function number (CBTOPENSTAT)
OUT: cc/cs COMPLETE/BUSY
BREAK: dØ/d1/aØ/a1
FUNC: Checks the completion of the request to open the loading tray.
NOTE:
SAMPLE:

```
loop:
        move.w    #CBTOPENSTAT, dØ
        jsr       _cdboot
        bcs.s     loop
```

CBTCHKDISC — CD Boot Check Disc

ENTRY: _cdboot
IN: dØ.w function number (CBTCHKDISC)
aØ pointer to scratch RAM($8ØØ)
OUT: cc/cs OK/BUSY
BREAK: dØ/d1/aØ/a1
FUNC: Requests to check whether a boot can be done or not.
NOTE: Returns BUSY only when interrupts are being handled by CBTINT.
SAMPLE:

```
        move.w    #CBTCHKDISC
        jsr       _cdboot
```

CBTCHKSTAT — CD Boot Check Status

ENTRY: _cdboot
IN: dØ.w function number (CBTCHKSTAT)
OUT: cc/cs COMPLETE/BUSY
dØ.w disc type
BREAK: dØ/d1/aØ/a1
FUNC: Checks boot completion and returns disk type
NOTE: disc type
- -1: not ready
- Ø: no disc
- 1: music disc
- 2: all rom disc
- 3: mixed disc
- 4: game system disc
- 5: game data disc
- 6: game boot disc
- 7: game disc

SAMPLE:

```
loop:
        move.w    #CBTCHKSTAT, dØ
        jsr       _cdboot
        bcs.s     loop
```


CONFIDENTIAL #161 PROPERTY OF SEGA

Sega Ozisoft

7. BACK-UP RAM

7-1 Back-up RAM

● MEGA-CD I/F Board (Development board)

At the time of power on, WRITE PROTECT is in effect.
SUB-CPU side $800000 (WORD ACCESS) ※$800001 in case of Byte Access.

bit0	2M RAM	0: Disable (Writing)
bit1	Included inside the Back-up RAM	1: Enable (Writing)

Since this function is implemented only for the development board, please do not use it in the released version.

Precaution

1. If BIOS is called when WRITE PROTECT is on, the following occur.
   ① BRAMINIT
      If not formatted, treated as if no RAM were present.
   ② BRAMFORMAT
      Returns error.
   ③ BRAMVERIFY
      Returns "no match" unless the same filename and contents have been written.
2. A file of more than 8 KB size can be used for the Back-up RAM cartridge.

BRMINIT — **Back-up RAM Initialization**

```
ENTRY:  _buram
IN:     d0.w   function number (BRMINIT)
        a0.l   pointer to scratch RAM (size: $640 bytes)
               Temporary work pointer
        a1.l   pointer to display strings buffer (size: 12 bytes)
               Pointer to the buffer for display strings.
OUT:    cc/cs  OK/ERROR
          cc:  SEGA formatted RAM is present
               d0.w       size of back-up ram
                          $2(000)~$100(000) bytes
          cs:  Not formatted or no RAM
               d0.w       size of back-up ram
               d1.w       0: No RAM
                          1: Not formatted
                          2: Other format
        a1.l   pointer to display strings
BREAK:  d0/d1/a0
FUNC:   Prepares to write into and read from Back-up RAM. Makes the table for the data
        protect function.
NOTE:   Do not destroy the scratch RAM during the BRMXXX functions.
SAMPLE:
          lea.l     scratch_RAM, a0
          lea.l     systemname, a1
          move.w    #BRAMINIT, d0
          jsr       _buram
          bcs.s     backupRAMformat
          bcc.s     OK
```


CONFIDENTIAL #161 PROPERTY OF SEGA

SegaXtreme Zisoft

BRMSTAT — Back-up RAM Status

ENTRY: _buram
IN: d0.w function number (BRMSTAT)
a1.l pointer to display string buffer (size: 12 bytes)
OUT: d0.w number of blocks of free area
d1.w number of files in directory
BREAK: none
FUNC: Returns how much Back-up RAM has been used.
NOTE: The free block are treated as one file. When treated as more than two files, an overflow error may occur when writing the directory. When the number of blocks and files are negative, no file can be accessed, formatting is needed.
SAMPLE:

```
        move.w      #BRAMSTAT.d0
        jsr         _buram
```

BRMSERCH — Back-up RAM Search

ENTRY: _buram
IN: d0.w function number (BRMSERCH)
a0.l pointer to parameter table
→ file_name. 11 file name (ASCII code [0~9 A~Z _])
zero. l 0
OUT: cc/cs found/not found
d0.w number of blocks
d1.b mode 0: normal -1: data protected (with protect function)
a0.l backup ram start address for search
BREAK: d0/d1/a0/a1
FUNC: Search for the desired file in back-up RAM
NOTE: 11 ASCII characters are used for the file name.
SAMPLE:

```
    table:
        db          'FILE_NAME__', 0
    subroutine:
        lea.l       table(pc), a0
        move.w      #BRMSERCH, d0
        jsr         _buram
        bcs.s       not found
        bcc.s       found
```

SegaXtreme

CONFIDENTIAL #161 PROPERTY OF SEGA

BRMREAD — Back-up RAM Read

ENTRY: _buram
IN: d0.w function number (BRMREAD)
a0.l pointer to parameter table
→ file_name.11 file name (ASCII code [0~9 A~Z _])
zero. l 0
a1.l pointer to write buffer
OUT: cc/cs OK/ERROR
d0.w number of blocks
d1.b mode 0: normal -1: data protected
BREAK: d0/d1/a0/a1
FUNC: Reads data form Back-up RAM
NOTE: Read size is in multiples of $20 bytes with the data protect function and in multiples of $40 bytes with no data protect function. 11 characters of ASCII code are used for the file name.
SAMPLE:

```
table:
        db          'FILE_NAME__'
subroutine:
        lea.l       table(pc), a0
        lea.l       write_buffer, a1
        move.w      #BRMREAD, d0
        jsr         _buram
```

BRMWRITE — Back-up RAM Write

ENTRY: _buram
IN: d0.w function number (BRMWRITE)
a0.l pointer to parameter table
→ file_name. 11 file name (ASCII code [0~9 A~Z _])
flag.b $00: normal $FF: encode (with protect function)
block_size.w flag=$00: 1 block=$40 byte
flag=$FF: 1 block=$20 byte
a1.l pointer to save data
OUT: cc/cs OK/ERROR complete/cannot write in the file
BREAK: d0/d1/a0/a1
FUNC: Writes data in Back-up RAM
NOTE: Save size is in multiples of $20 bytes with the data protect function and in multiples of $40 bytes with no protect function. 11 characters of ASCII code are used for the file name.
SAMPLE:

```
table:
        db          'FILE_NAME__', -1
        dw          $0008
subroutine:
        lea.l       table(pc), a0
        lea.l       save_data, a1
        move.w      #BRMWRITE, d0
        jsr         _buram
        bcs.s       error
        bcc.s       complete
```


CONFIDENTIAL #161 PROPERTY OF SEGA

Segaozisoft

BRMDEL — Back-up RAM Delete

ENTRY: _buram
IN: dØ.w function number (BRMDEL)
aØ.l pointer to parameter table
→ file_name. 11 file name (ASCII code [Ø~9 A~Z _])
zero. l Ø
OUT: cc/cs deleted/file not found
BREAK: dØ/d1/aØ/a1
FUNC: Deletes data on Back-up RAM
SAMPLE:

```
table:
        db          'FILE_NAME__', Ø
subroutine:
        lea. l      table (pc), aØ
        move.w      #BRMDEL, dØ
        jsr         _buram
```

BRMFORMAT — Back-up RAM Format

ENTRY: _buram
IN: dØ.w function number (BRMFORMAT)
OUT: cc/cs OK/ERROR (formatted/cannot format)
BREAK: dØ/d1/aØ/a1
FUNC: First initializes the directory, then formats
NOTE: Initialize first
SAMPLE:

```
        move.w      #BRAMFORMAT, dØ
        jsr         _buram
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

BRMDIR — Back-up RAM Directory

ENTRY: _buram
IN: d0.w function number (BRMDIR)
d1.l H: number of files to skip when cannot read all the files in one try.
L : size of directory buffer (number of files that can be read in the directory buffer.)
a0.l pointer to parameter table
→ file_name. 11 Template file name (ASCII code [0~9 A~Z _ *])
zero.l 0
a1.l pointer to directory buffer
OUT: cc/cs OK/FULL (complete/too much to read into directory buffer)
BREAK: d0/d1/a0/a1
FUNC: Reads directory
NOTE:
SAMPLE:

```
table:
        db          'FILE_NAME__', 0
subroutine:
        lea.l       table (pc), a0
        lea.l       directorybuffer, a1
        moveq.l     #DIRECTORYBUFFERSIZE, d1
        move.w      #BRAMDIR, d0
        jsr         _buram
        bcs.s       dirbufferfull
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

BRMVERIFY — Back-up RAM Verify

ENTRY: _buram
IN: d0.w function number (BRMVERIFY)
a0.l pointer to parameter table
→ file_name.11 File name (ASCII code [0~9 A~Z_])
flag.b $00: normal $FF: encode (with protect function)
block_size.w flag=$00: 1 block=$40 byte
flag=$FF: 1 block=$20 byte
a1.l pointer to save data
OUT: cc/cs OK/ERROR
d0.w Error No. -1: Data does not match
0: File not found
BREAK: d0/d1/a0/a1
FUNC: Checks data written on back-up RAM.

NOTE: Save size is in multiples of $20 bytes with the data protect function and in multiples of $40 bytes with no data protect function. 11 characters of ASCII code are used for the file name.

SAMPLE:

```
table:
        db          "FILE_NAME__", -1
        dw          $0008
subroutine:
        lea.l       table(pc), a0
        lea.l       save_data, a1
        move.w      #BRMVERIFY, d0
        jsr         _buram
        bcs.s       error
        bra.s       complete
error:
        tst.w       d0
        beq.s       filenotfound
        bne.s       dataerror
```

SegaOzisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

● MEGA-CD load sample

```
;-----------------------------------------------------------
;               loadsub
;       IN      d1.l    load size (byte)
;               a1      load start address on disc
;-----------------------------------------------------------

MAXERRWAITCNTØ    equ     (6Ø*1Ø)
MAXERRWAITCNT1    equ     6

loadsub:
        enter_subroutine
        lea.l           cbtreadsct(a5), aØ
        move.l          a1, dØ
        lsr.l           #8, dØ
        lsr.l           #3, dØ
        move.l          dØ, (aØ)                ; read start sector#
;
        divu.w          #75, dØ
        swap            dØ
        call            hex2bcd
        move.b          dØ, cdbfrm (a5)
;
        move.l          d1, dØ
        lsr.l           #8, dØ
        lsr.l           #3, dØ
        and.w           #$Ø7ff, d1
        beq.s           ?11
        addq.l          #1, dØ
?11:
        move.l          dØ, 4 (aØ)              ; read size (number of sector)
        move.w          dØ, cbtreadcnt(a5)

?retry:
        lea.l           cbtreadsct(a5), aØ
        move.w          #ROMREADN, dØ
        callf           _cdbios
        move.w          #MAXERRWAITCNTØ, d7
?null:
        bsr             cbtwaitd7
?loop:
        move.w          #CDCSTAT, dØ
        callf           _cdbios
        dbcc.w          d7, ?null
        bcs.s           ?retry
        move.b          #CDCMDSUBCPU, pcdcmode
        move.w          #CDCREAD, dØ
        callf           _cdbios
        bcs.s           ?retry
;check header
        lsr.w           #8, dØ
        cmp.b           cdbfrm(a5), dØ
        bne.s           ?retry
```


CONFIDENTIAL #161 PROPERTY OF SEGA

Sega Ozisoft

```
;
        movea.l     ploadbuf(a5), a0
        lea.l       cbthdrbuf(a5), a1
        move.w      #CDCTRN, d0
        callf       _cdbios
        bcs.s       ?retry
;check header
        move.l      cbthdrbuf(a5), d0
        lsr.w       #8, d0
        cmp.b       cdbfrm(a5), d0
        bne.s       ?retry
;
        moveq.l     #1, d1
        move.w      #PBITCCRZERO, ccr
        abcd        d1, d0
        cmpi.b      #$75, d0
        bcs.s       ?c1
        moveq.l     #0, d0
?cl:
        move.b      d0, cdbfrm(a5)
;
        move.l      a0, ploadbuf(a5)
        move.w      #CDCACK, d0
        callf       _cdbios
        move.w      #MAXERRWAITCNT1, d7
        addq.l      #1, cbtreadsct(a5)
        subq.l      #1, cbtreadnum(a5)
        subq.w      #1, cbtreadcnt(a5)
        bne.s       ?loop
        leave_subroutine
```

Sega Ozisoft

CONFIDENTIAL #161 PROPERTY OF SEGA

Sega Ozisoft

MEGA-CD BIOS MANUAL

Ver 2.00 Feb. 24 -'92

SEGA ENTERPRISES, LTD.
2-12, HANEDA 1-CHOME, OHTA-KU,
TOKYO 144, JAPAN

CONFIDENTIAL #161 PROPERTY OF SEGA