CD-ROM SYSTEM

BIOS Ver1.00

TABLE OF CONTENTS

; IPL Information Block Data Format (Data Record Top+1) 5

IPL Information 6

Memory Layout Normal 7

PSG Driver Load Time....................... 8

Graphic Driver Load Time 9

Kanji Rom Load Time 10

RAM 11

[$00] CD_BOOT: Boot CD-ROM BIOS 12

[$01] CD_RESET: Reset CD Drive 12

[$02] CD_BASE: Set Base Address of Data Record No....... 12

[$03] CD_READ: Read Data from CD 13

[$04] CD_SEEK: Seek CD to a Specified Address 13

[$05] CD_EXEC: Read Data from CD to Specified Address... 14

[$06] CD_PLAY: Search & Playback CD Audio 15

[$07] CD_SEARCH: Search CD Audio Track 16

[$08] CD_PAUSE: Pause CD Audio Playback 16

[$09] CD_STAT: Check CD Drive Status 17

[$0A] CD_SUBQ: Read Play Status & Subcode Q 17

[$0B] CD_DINFO: Read Table of Contents Data 18

[$0C] CD_CONTNTS: Initialize System Work 18

[$0D] CD_SUBRD: Read All Channel Bits of Playing Subcode. 19

[$0E] CD_PCMRD: Read Playing CD Linear PCM Data 19

[$0F] CD_FADE: Starting & Cancelling of Linear PCM and ADPCM Fade Out 19

[$10] AD_RESET: Reset ADPCM Controller 19

[$11] AD_TRANS: Transfer Data from CD to ADPCM Buffer ... 20

[$12] AD_READ: Read Data from ADPCM Buffer to Memory ... 20

[$13] AD_WRITE: Write Data from Memory to ADPCM Buffer... 21

[$14] AD_PLAY: Partial Playback of ADPCM Buffer Data ... 21

[$15] AD_CPLAY: Continue ADPCM playback 22

[$16] AD_STOP: Stop Data Playback ADPCM Buffer 22

[$17] AD_STAT: Read ADPCM Controller Status 22

Back Up Memory Format 23

[$18] BM_FORMAT: Initialize Back Up Memory 24

[$19] BM_FREE: Check Free Back Up Memory 24

[$1A] BM_READ: Read Data from Back Up Memory 24

[$1B] BM_WRITE: Write Data to Back Up Memory 24

[$1C] BM_DELETE: Delete Data from Back Up Memory 25

[$1D] BM_FILES: Search Back Up Memory File 25

[$1E] EX_GETVER: Get BIOS Version No. 25

[$1F] EX_SETVEC: Set User Interrupt Vector Address 26

[$20] EX_GETFNT: Transfer Kanji ROM Font Data 27

[$21] EX_JOYSNS: Sense Joy Pad Status 28

[$23] EX_SCRSIZ: Set No. of Characters in Virtual Screen.. 28

[$24] EX_DOTMOD: Set VRAM Access Dot Width 29

[$25] EX_SCRMOD: Set Screen Mode 29

[$26] EX_IMODE: Set Increment Width for Memory Access ... 29

[$27] EX_VMODE: Set VRAM Address Increment by EX-SCRSIZ.. 29

[$28] EX_HMODE: Set VRAM Address Increment Width by 1 ... 30

[$29] EX_VSYNC: WAIT VSYNC 30

[$2A] EX_RCRON: RASTER HIT IRQ ON 30

[$2B] EX_RCROFF: RASTER HIT IRQ OFF 30

[$2C] EX_IRQON: VSYNC IRQ ON 30
[$2D] EX_IRQOFF: VSYNC IRQ OFF 31
[$2E] EX_BGON: Background Display ON 31
[$2F] EX_BGOFF: Background Display OFF 31
[$30] EX_SPRON: Sprite Display ON 31
[$31] EX_SPROFF: Sprite Display OFF 31
[$32] EX_DSPON: Background/Sprite Display ON 31
[$33] EX_DSPOFF: Background/Sprite Display OFF 32
[$34] EX_DMAMOD: DMA Transfer Mode Set 32
[$35] EX_SPRDMA: DMA Transfer between VRAM and SATB 32
[$36] EX_SATCLR: Clear Sprite Attribute in SAT 32
[$37] EX_SPRPUT: Set Sprite Attribute in SAT 32
[$38] EX_SETRCR: Set Raster Detect Register Value 33
[$39] EX_SETRED: Set VRAM Read Mode 33
[$3A] EX_SETWRT: Set VRAM Write Mode 33
[$3B] EX_SETDMA: Set DMA Parameters 33
[$4C] EX_COLORCMD: Set & Read Color Palette 34
[$3C] EX_BINBCD: Transfer Binary Value to BCD Value 34
[$3D] EX_BCDBIN: Transfer BCD Value to Binary Value 35
[$3E] EX_RND: Get Random Number 35
[$3F] MA_MUL8U: Unsigned 8 bit Multiplication 35
[$40] MA_MUL8S: Signed 8 bit Multiplication 35
[$41] MA_MUL16U: Unsigned 16 bit Multiplication 35
[$42] MA_DIV16U: Unsigned 16 bit Division 36
[$43] MA_DIV16S: Signed 16 bit Division 36
[$44] MA_SQRT: Square Root 36
[$45] MA_SIN: Get Sine 36
[$46] MA_COS: Get Cosine 36
[$47] MA_ATNI: Get ATN 36
[$48] PSG_BIOS: PSG Driver 37
[$49] GRP_BIOS: Graphic Driver 37
[SUB ERROR CODE LIST] 41
BIOSW.H 42-46

```
;
;       IPL INFORMATION BLOCK DATA FORMAT (DATA RECORD TOP + 1)
;
        db      ?       ;00 IPLBLK H    ;load start record no. of CD
        db      ?       ;01 IPLBLK M    ;load start record no. of CD
        db      ?       ;02 IPLBLK L    ;load start record no. of CD
        db      ?       ;03 IPLBLN      ;load block length of CD
        db      ?       ;04 IPLSTA L    ;program load address L
        db      ?       ;05 IPLSTA H    ;program load address H
        db      ?       ;06 IPLJMP L    ;program execute address L
        db      ?       ;07 IPLJMP H    ;program execute address H
;
        db      ?       ;08 IPLMPR2     ;ipl set mpr2 (+ max_mapping)
        db      ?       ;09 IPLMPR3     ;ipl set mpr3 (+ max_mapping)
        db      ?       ;10 IPLMPR4     ;ipl set mpr4 (+ max_mapping)
        db      ?       ;11 IPLMPR5     ;ipl set mpr5 (+ max_mapping)
        db      ?       ;12 IPLMPR6     ;ipl set mpr6 (+ max_mapping)
;
        db      ?       ;13 OPENMODE    ;opening mode
                                        ;bit7654_3210
                                        ;    |||    |└── data read to vram
                                        ;    |||    |    0:not read, 1:read
                                        ;    |||    └─── data read to adpcm buffer
                                        ;    |||         0:not read, 1:read
                                        ;    ||└──────── bg display
                                        ;    ||          0:display on, 1:display off
                                        ;    |└───────── adpcm play
                                        ;    |           0:play, 1:not play
                                        ;    └────────── adpcm play mode
                                        ;                0:single, 1:repeat
;
        db      ?       ;14 GRPBLK H    ;opening graphic data record no.
        db      ?       ;15 GRPBLK M    ;opening graphic data record no.
        db      ?       ;16 GRPBLK L    ;opening graphic data record no.
        db      ?       ;17 GRPBLN      ;opening graphic data length
        db      ?       ;18 GRPADR L    ;opening graphic data read address L
        db      ?       ;19 GRPADR H    ;opening graphic data read address H
;
        db      ?       ;20 ADPBLK H    ;opening ADPCM data record no.
        db      ?       ;21 ADPBLK M    ;opening ADPCM data record no.
        db      ?       ;22 ADPBLK L    ;opening ADPCM data record no.
        db      ?       ;23 ADPBLN      ;opening ADPCM data length
        db      ?       ;24 ADPRATE     ;opening ADPCM sampling rate
;
        db      0       ;25             ;(reserve)
        db      0       ;26             ;(reserve)
        db      0       ;27             ;(reserve)
        db      0       ;28             ;(reserve)
        db      0       ;29             ;(reserve)
        db      0       ;30             ;(reserve)
        db      0       ;31             ;(reserve)
;
        db      'PC Engine CD-ROM SYSTEM',0     ;(ID string)
        db      'Copyright HUDSON SOFT / NEC Home Electronics, Ltd.',0
        db      '                '              ;program name     (16 bytes)
        db      '      '                        ;                 ( 6 bytes)
;
```

IPL INFORMATION

IPLBLK	Top record no. where the program is contained
IPLBLN	No. of records for program to read
IPLSTA	Main memory address for program read
IPLJMP	Starting address of execution after program read
IPLMPR	Bank no. to set to MPR before program read
OPENMODE	Opening mode

By setting the bits (bit5,0) to the value specified by OPENMODE in IPL INFORMATION, data can be loaded into VRAM and displayed before loading the user program.

```
bit 7654_3210
      |     |
      |     L__ 0: Do not load
      |         1: Load
        ------- 0: Display ON
                1: Display OFF
```

GRPBLK H, GRPBLK M, and GRPBLK L specifies the top record of data to load.

GRPADR L and GRPADR H specifies the top VRAM address into which BG font data is read.

GRPBLN specifies the total record that contains color palette data, BAT data, and BG font data in the format shown below.

```
                IPL GRAPHIC DATA FORMAT
GRPBLK      +--------------------+.........................
            |   COLOR PALETTE    |        1 RECORD       :
            +--------------------+.......................:
            |   BAT DATA         |        1 RECORD       :
            +--------------------+..................   GRPBLN
            |                    |                       :
            |   BG FONT DATA     |  GLPBLN - 2 RECORD    :
            |                    |                       :
            +--------------------+.......................:...
```

By setting the bits (bit7,6,1) to the value specified by OPENMODE in IPL INFORMATION, data can be loaded into ADPCM buffer and displayed before loading the user program.

```
bit 7654_3210
    ||     |
    ||     L__ 0: Load
    ||         1: Do not load
    |L-------- 0: Playback
    |          1: Do not playback
    L--------- 0: Playback once
               1: Repeat playback
```

ADPBLK H, ADPBLK M, and ADPBLK L specifies the top record of data to load.
ADPBLN specifies the number of ADPCM data record.
ADPRATE specifies the ADPCM sampling rate.

Normal Memory Layout

Logical Address

$FFFF $E000	BIOS ROM	MPR7=00
$DFFF $C000	USER AREA	
$BFFF $A000	USER AREA	
$9FFF $8000	USER AREA	
$7FFF $6000	USER AREA	
$5FFF $4000	USER AREA	
$3FFF $2000	WORK RAM	MPR1=F8
$1FFF $0000	I/O	MPR0=FF

When calling BIOS or using an interrupt routine from BIOS, MPR0,1,7 cannot be changed.

Memory Layout At PSG Driver Load Time

Logical Address

$FFFF $E000	BIOS ROM	MPR7=00
$DFFF $C000	PSG DRIVER	MPR6=02
$BFFF $A000	PSG DATA 1	MPR5=??
$9FFF $8000	PSG DATA 0	MPR4=??
$7FFF $6000	USER AREA	
$5FFF $4000	USER AREA	
$3FFF $2000	WORK RAM	MPR1=F8
$1FFF $0000	I/O	MPR0=FF

When the PSG driver is loaded, MPR 4,5,6 are set as shown at left. After completion of the PSG driver, they will be reset to the user set status. A user interrupt routine cannot be set here.

Memory Layout At Graphic Driver Load Time

Logical Address

$FFFF $E000	BIOS ROM	MPR7=00
$DFFF $C000	GRAPHIC DRIVER	MPR6=03
$BFFF $A000	USER AREA	
$9FFF $8000	USER AREA	
$7FFF $6000	USER AREA	
$5FFF $4000	USER AREA	
$3FFF $2000	WORK RAM	MPR1=F8
$1FFF $0000	I/O	MPR0=FF

When GRAPHIC DRIVER is loaded, MPR 6 is set as shown at left. After completion of the GRAPHIC DRIVER, it will be reset to the user set status. A user interrupt routine cannot be set here.

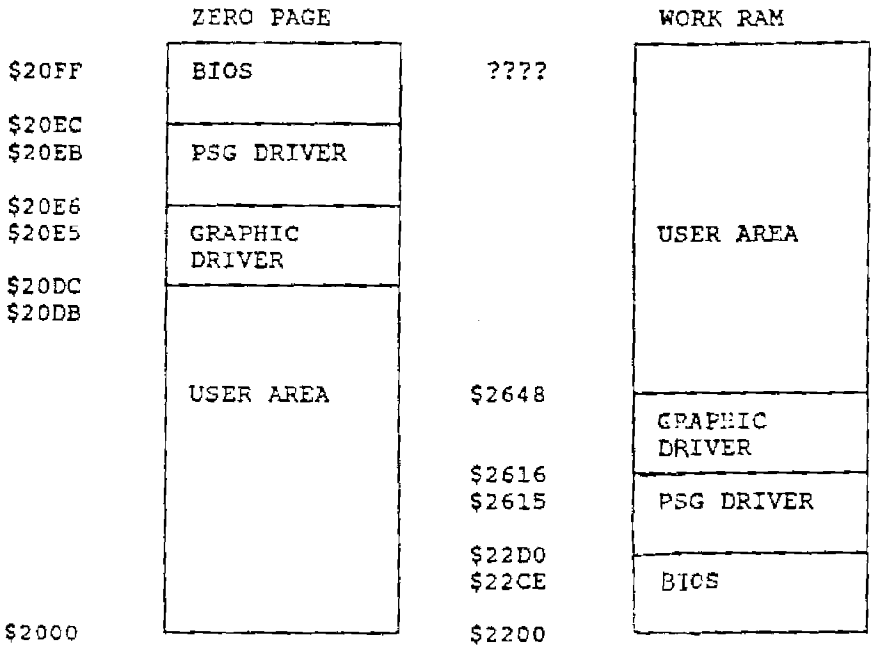
Memory Layout At Kanji ROM Load Time

Logical Address				
$FFFF $E000	BIOS ROM	MPR7=00		
$DFFF $C000	KANJI ROM 1	MPR6=??	USER AREA	
$BFFF $A000	KANJI ROM 0	MPR5=??	USER AREA	
$9FFF $8000	USER AREA		KANJI ROM 1	MPR4 =??
$7FFF $6000	USER AREA		KANJI ROM 0	MPR3 =??
$5FFF $4000	USER AREA			
$3FFF $2000	WORK RAM	MPR1=F8		
$1FFF $0000	I/O	MPR0=FF		

Two kinds of mapping shown above are possible. Which map is used depends on the specified Kanji font data destination address.

Destination address: below $A000, Kanji ROM $A000 ~ $DFFF
Destination address: above $A000, Kanji ROM $6000 ~ $9FFF

RAM Memory Layout



Working area of the PSG DRIVER and the GRAPHIC DRIVER can be used as a user area when each driver is not used. Do not destroy the BIOS area.

[$00] CD_BOOT: Boot CD-ROM BIOS

IN None
OUT None

Description Shifts to CD-ROM BIOS boot status.
Displays the title picture and waits for the RUN command.

[$01] CD_RESET: Reset CD drive

IN None
OUT AREG: Execution result
$00 OK
ELSE SUB ERROR CODE
(See SUB ERROR CODE list)

Description Brings CD-ROM drive to the power up state.
If drive is reset, the drive will check DISC and read TOC when next command is issued. It will take a few seconds to reach ready status, therefore, make sure to check CD_STAT to see if the drive is ready before issuing the next command. Do not use this call except for a DISC change. (Not necessary to call this command for initialization of application.)

[$02] CD_BASE: Set Base Address of Data Record No.

IN

_BH:	Base address type		
	%0000_00XX	%0100_00XX	%1000_00XX
	LOG.BLOCK	CD-ATIME	CD-TNO
_AL:	A_REC H	AMIN(BCD)	TNO(BCD)
_AH:	A_REC M	ASEC(BCD)	NO USE
_BL:	A_REC L	AFRAME(BCD)	NO USE

_CL: Set mode
bit 7 6 5 4 3 2 1 0
XXXXXX00 Set both
XXXXXX01 Set first
XXXXXX10 Set second
XXXXXX11 Do not set

OUT _AL: A_REC H
_AH: A_REC M
_BL: A_REC L

Description There are 2 sets of base addresses. Normally, the system sets the head record (IPL RECORD) of the top data track to both addresses during IPL. At data read, based on the base address the user will specify an offset read address. Normally the first set base address is used but when a not recoverable DISC damage occurs, the second set is used. Specify the top of the alternate track prepared by the user for errors as the second base address. The first base address is changeable. If an alternate track is not used, use the same address for the first second base address.

[$03] CD_READ: Read Data from CD

IN

```
_CL: REC H
_CH: REC M
_DL: REC L

_DH: Data read address type
     01:LOCAL  FE:VRAM      FF:VRAM   2~6:MPR NO.
_BL: ADR L     ADR L        ADR L     BANK NO. ($80~$87)
_BH: ADR H     ADR H        ADR H     NO USE

_AL: REC LEN   BYTE LEN L   REC LEN   REC LENGTH
_AH: NO USE    BYTE LEN H   NO USE    NO USE
```

OUT

```
AREG:     Execution result
          $00       OK
          ELSE      SUB ERROR CODE
```

Description Length of a record is 2KB. VDTIN_FLG is set to 1 during a read to VRAM. Normal operation is not guaranteed if MAWR or MARR is changed by interrupt, etc. during this time. Read using an MPR NO. will read to the specified RAM BANK using the specified MPR.NO as an access window. Example, MPR=3, $6000 ~ $7FFF If the number of record is more than 4 (8KB), the specified bank number will be changed by incrementing one. The current bank number will be saved and restored.

[$04] CD_SEEK: Seek CD to a Specified Address

IN

```
_CL: REC H
_CH: REC M
_DL: REC L
```

OUT None

Description Moves the head of the CD-ROM to a specified address. The head will automatically be returned to the home position and the drive motor will be turned off if no commands are received for 15 minutes. A command issued after this will take longer to complete as the motor must be restarted and the head must be moved back into position.

[$05] CD_EXEC: Read Data From CD to Specified Address

IN	_CL:	REC H	
	_CH:	REC M	
	_DL:	REC L	
	_DH:	Data read address type	
		01:LOCAL	ELSE:MPR NO. (2 ~ 6)
	_BL:	ADR L	BANK NO. ($80 ~ $87)
	_BH:	ADR H	NO USE
	_AL:	REC LENGTH	REC LENGTH
	_AH:	NO USE	NO USE

Description It will be reset if an error occurs.

[$06] CD_PLAY: Search and Playback CD Audio

IN

_BH:	Playback start address type			
	%0000_0000	%0100_0000	%1000_0000	%1100_0000
	LOG.BROCK	CD-ATIME	CD-TNO	CURRENT
_AL:	A_REC H	AMIN(BCD)	TNO(BCD)	NO USE
_AH:	A_REC M	ASEC(BCD)	NO USE	NO USE
_BL:	A_REC L	AFRAME(BCD)	NO USE	NO USE

_DH:	Playback end address type			
	%00YY_YYYY	%01YY_YYYY	%10YY_YYYY	%11YY_YYYY
	LOG.BROCK	CD-ATIME	CD-TNO	READ OUT
_CL:	A_REC H	AMIN(BCD)	TNO(BCD)	NO USE
_CH:	A_REC M	ASEC(BCD)	NO USE	NO USE
_DL:	A_REC L	AFRAME(BCD)	NO USE	NO USE

_DH: Play mode
bit 7654_3210
YY_YYYY

XX00_0000	Mute play. Status will return rignt after command is received.
XX00_0001	Infinite repeat play. Drive is busy after command is received until the next command issuance.
XX00_0010	Normal play. Drive is busy after the command is received until playback completion.
XX00_0011	Normal play. Status will return right after the command is received.
XX00_0100	Do not change from previously set mode.

OUT AREG: Execution result
$00 OK
ELSE SUB ERROR CODE

Description After fade out execution, and if not canceled, the sound circuit will be set to muting status. Cancel fade out by issuing CD_FADE in advance.

[$07] CD_SEARCH: Search CD Audio Track

```
IN    _BH: Address type of search objective
           %0000_00XX        %0100_00XX        %1000_00XX
           LOG.BROCK         CD-ATIME          CD-TNO
      _AL: A_REC H           AMIN(BCD)         TNO(BCD)
      _AH: A_REC M           ASEC(BCD)         NO USE
      _BL: A_REC L           AFRAME(BCD)       NO USE
      _BH: Search mode
           bit 7654_3210
               XXXX_XX??
                      ││
                      │└── 0 : No return until search is completed
                      │    1 : Return after search without
                      │        completing the search.
                      └─── 0 : Pause where search is completed.
                           1 : Play from where search is
                               completed. (To DISC [illegible])

OUT  AREG:      Execution result
                $00   OK
                ELSE SUB ERROR CODE
```

Description When search mode is set to 1, the execution result will always be 0. To find errors, issue CD-STAT set to drive BUSY check. After checking, SCSISTS will be set to 0 if normal termination occurred, and to a non-zero number if an error occurred. (The search operation and the contents of SCSISTS will not be guaranteed if a drive command is issued before the search command was completed.)

[$08] CD_PAUSE: Pause CD Audio Playback

IN None

OUT AREG: Execution result
$00 OK
ELSE SUB ERROR CODE

Description Stops playback at the current position during audio playback. In order to continue playback, call CD_PLAY using the current position as the playback starting address.

[$09] CD_STAT: Check CD Drive Status

```
IN   AREG:     Get status mode
               $00   Drive busy check
               ELSE  Drive unit ready check

OUT  AREG:     Execution result
               BUSY CHECK        READY CHECK
               $00   NOT BUSY    $00   READY
               ELSE  BUSY        ELSE  SUB ERROR CODE
```

[$0A] CD_SUBQ: Read Play Status and Sub Code Q

```
IN   _BX:      Read buffer (10 bytes) address

OUT  AREG:     Execution result
               $00        OK
               ELSE       SUB ERROR CODE
Description    SUBCODE-Q 10bytes
               0: PLAYING STATUS
                          $00 PLAYING
                          $01 STILL
                          $02 PAUSE
                          $03 NOT PLAYING

               1: CONTROL/ADDRESS
               bit 3 2 1 0
                   0 0 X 0  2 AUDIO CHANNELS WITHOUT PRE-EMPHASIS
                   0 0 X 1  2 AUDIO CHANNELS WITH PRE-EMPHASIS
                   1 0 X 0  4 AUDIO CHANNELS WITHOUT PRE-EMPHASIS
                   1 0 X 1  4 AUDIO CHANNELS WITH PRE-EMPHASIS
                   0 1 X 0  DATA TRACK
                   0 1 X 1  RESERVED
                   1 1 X X  RESERVED
                   X X 0 X  DIGITAL COPY PROHIBITED
                   X X 1 X  DIGITAL COPY PERMITTED

               2: TRACK NO(BCD)
               3: INDEX(BCD)
               4: REAL MIN(BCD)
               5: REAL SEC(BCD)
               6: REAL FRAME(BCD)
               7: TOTAL MIN(BCD)
               8: TOTAL SEC(BCD)
               9: TOTAL FRAME(BCD)
```

Data contents are guaranteed if the drive is ready and the disk is rotating.

[$0B] CD_DINFO: Read TOC (Table Of Contents) Data.

IN _BX: Read buffer (4 bytes) address

_AL: Read data type

%0000_0000	%0000_0001	%0000_0010	%0000_0011
MIN TNO(BCD)	READOUT AMIN	AMIN(BCD)	A_REC H
MAX TNO(BCD)	READOUT ASEC	ASEC(BCD)	A_REC M
0	READOUT AFRAME	AFRAME(BCD)	A_REC L
0	0	SUB-Q CONTROL	SUB-Q CONTROL

_AH: TNO(BCD) (Valid when _AL is 0000_0010 or 0000_0011)

OUT AREG: Execution result
$00 OK
ELSE SUB ERROR CODE

Description Reads TOC information of the DISC currently attached. Maximum track number is 99.

[$0C] CD_CONTNTS: Initialize System Work

IN None

OUT AREG: $00 OK
ELSE SUB ERROR CODE

TNOMIN:	Minimum track number (BCD)
TNOMAX:	Maximum track number (BCD)
OUTMIN:	Read out area start minute (BCD)
OUTSEC:	Read out area start second (BCD)
OUTFRM:	Read out area start frame (BCD)

Description The information listed above is normally set up during IPL. If DISC is changed during use, BIOS will not recognize this fact, so reset the system using this call. If this call is not used to notify BIOS of a change, BIOS calls will not be guaranteed.

[$0D] CD_SUBRD: Read All Channel Bits of Playing Subcode

IN None

OUT AREG: SUBCODE

```
bit 7654_3210
    |||| |||└─ W ch
    |||| ||└── V ch
    |||| |└─── U ch
    |||| └──── T ch
    |||└────── S ch
    ||└─────── R ch
    |└──────── Q ch
    └───────── P ch
```

CARRY:

$00	SUBCODE is read
$01	SUBCODE not played

Description If drive is ready and disk is rotating, data contents are guaranteed.

[$0E] CD_PCMRD: Read Playing CD Linear PCM Data

IN AREG:

$00	R-ch
$01	L-ch

OUT XREG: AUDIO DATA L
YREG: AUDIO DATA H

Description Read Linear PCM 16 bit data. One time read requires at least 25μsec.

[$0F] CD_FADE: Starting and Cancelling of Linear PCM and ADPCM Fade Out

IN AREG: Operation mode

$00	Cancel	FADE OUT
$08	PCM	FADE OUT (6.0 sec)
$0A	ADPCM	FADE OUT (6.0 sec)
$0C	PCM	FADE OUT (2.5 sec)
$0E	ADPCM	FADE OUT (2.5 sec)

OUT None

Description Once fade out is set, audio circuit will remain in a muting state until canceled. Cancel fade out before playing another song.

[$10] AD_RESET: Reset ADPCM Controller

IN None

OUT None

[$11] AD_TRANS: Transfer Data From CD to ADPCM Buffer

IN _CL: REC H
_CH: REC M
_DL: REC L

_AL: Number of transfer record

_DH: Transfer mode
$00 Set top address of destination
ELSE Transfer to current write point
(Address specification by _BX is unnecessary)

_BX: ADPCM buffer address

OUT AREG: Execution result
$00 OK
ELSE SUB ERROR CODE

[$12] AD_READ: Read Data from ADPCM Buffer to Memory

IN _CX: ADPCM Buffer address

_DH:	Read address type 00:LOCAL	FF:VRAM	2 ~ 6:MPR NO.
_BL:	ADR L	ADR L	BANK NO.($80 ~ $87)
_BH:	ADR H	ADR H	NO USE

OUT AREG: Execution result
$00 OK
ELSE ERROR

Description VDTIN_FLG changes to 1 during a read to VRAM. If MAWR or MARR is changed during this time because of an interrupt, etc., normal operation is not guaranteed.

[$15] AD_CPLAY: Continue ADPCM Playback

IN _CL: REC H
_CH: REC M
_DL: REC L

_AL: No. of playback record L
_AH: No. of playback record M
_BL: No. of playback record H

_DH: Sampling rate ($00 - $0E)
f KHz = 32 / (16 - _DH)

OUT AREG: Execution result
$00 OK
ELSE ERROR

Description Continues playback of ADPCM by regularly transferring 16 records of ADPCM data to the ADPCM buffer from the specified record. Minimum playback record number is 32 records. If read data, etc. is executed during repeated playback, head will move to a location more than 200 records away from the next ADPCM to be read or read operation will be retried. Then the ADPCM data cannot be read and playback will stop.

[$16] AD_STOP: Stop Data Playback ADPCM Buffer

IN None

OUT None

Description Stops ADPCM playback. Since playback speed is about 16KHz, a delay will occur before stopping. Check stop status through AD_STAT.

[$17] AD_STAT: Read ADPCM Controller Status

IN None

OUT AREG: ADPCM controller status
$00 ADPCM not busy (End or Not play)
ELSE ADPCM busy

XREG: ADPCM buffer and playback status
$00 Playing
More than a half buffer of data left
$01 Stop playback
$04 Playing
Less than a half buffer of data left

[$13] AD_WRITE: Write Data from Memory to ADPCM Buffer

IN _CX: ADPCM buffer address

_DH:	Write data address type		
	00:LOCAL	FF:VRAM	2 ~ 6:MPR NO.
_BL:	ADR L	ADR L	BANK NO. ($80 ~ $87)
_BH:	ADR H	ADR H	NO USE

_AX: Number of write size byte

OUT AREG: Execution result
$00 OK
ELSE ERROR

Description VDTIN_FLG changes to 1 during a read to VRAM. If MAWR or MARR is changed during this time because of an interrupt, etc., normal operation is not guaranteed.

[$14] AD_PLAY: Partial Playback of ADPCM Buffer Data

IN _BX: ADPCM buffer playback starting address

_AX: Number of playback size byte

_DH: SAMPLING RATE ($00 ~ $0E)
f KHz = 32 / (16 ~ _DH)

_DL: Mode

```
bit 7654_3210
     ?XXX_XXX?
     |       └── COUNTER MODE
     |            0: Set ADR, LENGTH, and RATE
     |            1: Set previous ADR, LENGTH, and RATE
     └────────── PLAY MODE
                  0: AUTO STOP
                  1: REPEAT
```

OUT AREG: Execution result
$00 OK
ELSE ERROR

Description This call will results in an error during AD-PCM playback. To ensure execution, stop playback by issuing AD_STOP. Then check to make sure playback is stopped using AD_STAT. If the status is okay, it is safe to issue the AD_PLAY command.

Back Up Memory Format

2KB battery back up memory is provided for IFU-30. This memory is commonly used by multiple applications, and is managed in the following format:

RAM file top

0000-0003	"HUBM" Back up memory ID
0004.w	RAM file size
0006.w	Unused area address
0008 ~ 000f	System reserved area

FCB format

+ 00.w	USER ID
+ 02.10byte	File name

DIR format

+ 00.w	Data block length + 10H
+ 02.w	Data block and FCB check sum
+ 04.12byte	FCB data
+ 16...	Data block

$8000 ~ $dfff are used as windows for back up memory access.

[$18] BM_FORMAT: Initialize Back Up Memory

IN _AX: Password Address

Password = "!BM FORMAT!"

OUT AREG: Execution result
$00 OK
$01 Cannot format

Description Prepare the password buffer and set its top address to _AX. If the password does not match, do not format.

[$19] BM_FREE: Check Free Back Up Memory

IN None

OUT _CX: Unused capacity in bytes

AREG: Execution result
$00 OK
$FF Format error

[$1A] BM_READ: Read Data from Back Up Memory

IN _AX: FCB address
_BX: Data read address
_CX: No. of bytes to read
_DX: Offset within file from top

OUT _CX: No. of bytes actually read

AREG: Execution result
$00 OK
$01 Cannot find file
$02 Bad data (Check sum error)
$FF Format error

[$1B] BM_WRITE: Write Data to Back Up Memory

IN _AX: FCB address
_BX: Data memory address
_CX: No. of bytes to write
_DX: Offset within file from top

OUT AREG: Execution result
$00 OK
$01 Not enough back up memory
$FF Format error

[$1C] BM_DELETE: Delete Data from Back Up Memory

IN	_AX:	FCB address	
OUT	AREG:	Execution result	
		$00	OK
		$01	Cannot delete
		$FF	Format error

[$1D] BM_FILES: Search Back Up Memory File

IN	_BX:	FCB destination address	
	_AL:	Number from top of file (top = 1)	
OUT	AREG:	Execution result	
		$00	OK
		$01	Cannot find file _AL = Largest file number
		$FF	Format error

[$1E] EX_GETVER: Get BIOS Version No.

IN	None	
OUT	XREG:	Integer part of version number
	YREG:	Decimal part of version number

[$1F] EX_SETVEC: Set User Interrupt Vector Address

IN AREG: Vector number

$00	IRQ2
$01	IRQ
$02	TIMER
$03	NMI
$04	SYNC
$05	RCR
$06	SOFT RESET

XREG: Vector address low
YREG: Vector address high

OUT None

Description The above vectors are valid when its IRQ_M bit is ON. Turn on the specified bit after execution of this call. Operation [illegible] guaranteed if a bit is turned ON before setting the Vector.

```
IRQ_M
7 6 5 4 3 2 1 0
│ │ │ │ │ │ │ └── irq2  jmp vector enb
│ │ │ │ │ │ └──── irq   jmp vector enb
│ │ │ │ │ └────── timer jmp vector enb
│ │ │ │ └──────── nmi   jmp vector enb
│ │ │ └────────── if irq vector off then sync jmp vector enb
│ │ └──────────── nop sync irq
│ └────────────── if irq vector off then rcr jmp vector enb
└──────────────── nop rcr irq
```

Return from SYNC and RCR routines by using the rts instruction. Return from the others (IRQ2,IRQ, TIMER, and NMI) with the rti instruction.

To use the timer, the user must set the timer count and then turn the timer ON, after setting the vector bit on. The PSG driver in BIOS cannot be used during timer access, however, both can be used by calling PSG driver from the user timer routine. The PSG driver should be used with VSYNC access when used with the timer vector.

SOFT RESET VECTOR is the address branched out to when RUN+SELECT is pressed when IRQ in BIOS or the joy pad sense routine EX_JOYSNS is used.

[$20] EX_GETFNT: Transfer Kanji ROM Font Data (32 bytes)

IN _AX: Kanji code (SHIFT JIS CODE)

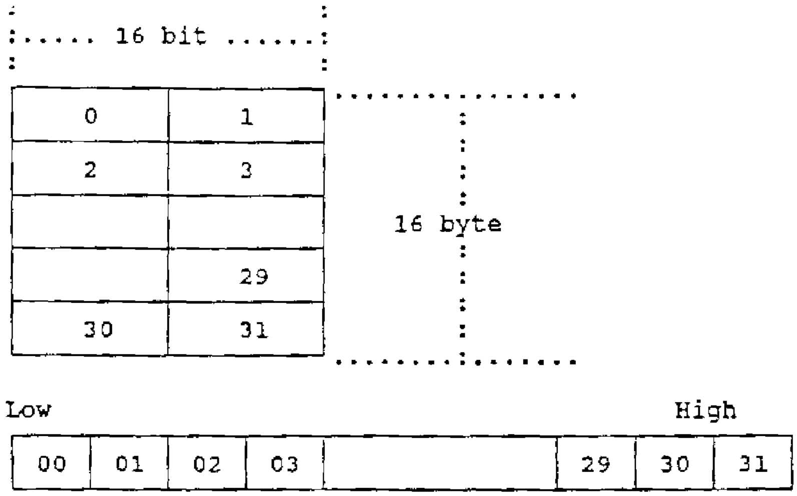
_BX: Transfer destination address

_DH: Transfer mode
$00 Transfer dot font 16x16
$01 Transfer dot font 12x12

OUT AREG: Execution result
$00 OK
$01 Kanji code error

Description This BIOS provides Kanji up to JIS level 1. $A000 ~ $DFFF or $6000 ~ $9FFF are used as the Kanji ROM window using the destination address.

Output Format for font pattern



[$21] EX_JOYSNS: Sense Joy Pad Status

IN JOYENA: Specifies software reset enable/disable.
Fixed to 1 when multi-tap is not used.
When multi-tap is used, refer to the following:

```
bit7654_3210
   XXX?_????  (Enable with bit ON)
      |  |||└─ #1 pad
      |  ||└── #2 pad
      |  |└─── #3 pad
      |  └──── #4 pad
      └─────── #5 pad
```

OUT JOY,X: Senses ON/OFF
(X = 0 to 4)
JOYTRG,X: Sense trigger (OFF -> ON)
(X = 0 to 4)
JOYOLD,X: JOY contents of work one loop before
(X = 0 to 4)

```
bit7654_3210
   XXXX_XXXX
   ||||  |||└─ ONE     0:off 1:on
   ||||  ||└── TWO     0:off 1:on
   ||||  |└─── SELECT  0:off 1:on
   ||||  └──── RUN     0:off 1:on
   |||└─────── UP      0:off 1:on
   ||└──────── RIGHT   0:off 1:on
   |└───────── DOWN    0:off 1:on
   └────────── LEFT    0:off 1:on
```

[$23] EX_SCRSIZ: Set No. of Characters in Virtual Screen

(Set to memory width register MWR)

IN AREG: Screen size (SCREEN;R09 bit4-6)

bit 2 1 0	x	y
0 0 0	32	32
0 0 1	64	32
0 1 0	128	32
0 1 1	128	32
1 0 0	32	64
1 0 1	64	64
1 1 0	128	64
1 1 1	128	64

OUT None

[$24] EX_ DOTMOD: Set VRAM Access Dot Width

(Set to memory width register MWR)

IN AREG: Dot size (VM;R09 bit0,1 SM;R09 bit 2,3)

OUT None

[$25] EX_SCRMOD: Set Screen Mode

IN AREG: Clock
$00 5MHz
$01 7MHz

XREG: Horizontal character size (do not use odd data)
10 ~ 34 (5MHz)
16 ~ 44 (7MHz)

YREG: Vertical character size
10 ~ 30 (5MHz, 7MHz)

OUT CARRY: 0 OK
1 ERROR

Description Default value is 5MHz, 32 x 30.

[$26] EX_IMODE: Set Increment Width for Memory Access

(1W;R05 bit 11,12)

IN AREG: 0 +1
1 +32
2 +64
3 +128

OUT None

[$27] EX_VMODE: Set VRAM Address Increment Width to 32,64,128 According to the Screen with EX_SCRSIZ.

IN None

OUT None

Description Set screen mode using EX_SCRMOD before using EX_VMODE. Use IRQ routine in BIOS.

[$28] EX_HMODE: Set VRAM Address Increment Width by 1

IN None

OUT None

Description Set screen mode using EX_SCRMOD before using EX_HMODE. Use IRQ routine in BIOS.

[$29] EX_VSYNC: WAIT VSYNC

IN None

OUT None

BREAK AREG

Description Use IRQ routine in BIOS. EX_VSYNC will return without waiting for VSYNC when Bit1 of IRQ_M is ON (while IRQ). If VSYNC IRQ is OFF, it is set ON.

[$2A] EX_RCRON: RASTER HIT IRQ ON (IE;R05 bit2)

IN None

OUT None

BREAK AREG

Description Use IRQ routine in BIOS.

[$2B] EX_RCROFF: RASTER HIT IRQ OFF (IE;R05 bit2)

IN None

OUT None

BREAK AREG

Description Use IRQ routine in BIOS.

[$2C] EX_IRQON: VSYNC IRQ ON (IE;R05 bit3)

IN None

OUT None

BREAK AREG

Description Use IRQ routine in BIOS.

[$2D] EX_IRQOFF: VSYNC IRQ OFF (IE;R05 bit3)

IN None

OUT None

BREAK AREG

Description Use IRQ routine in BIOS.

[$2E] EX_BGON: Background Display ON (BB;R05 bit7)

IN None

OUT None

Description Operates at a VSYNC interrupt after call. Use IRQ routine in BIOS.

[$2F] EX_BGOFF: Background Display OFF (BB;R05 bit7)

IN None

OUT None

Description Operates at a VSYNC interrupt after call. Use IRQ routine in BIOS.

[$30] EX_SPRON: Sprite Display ON (SB;R05 bit6)

IN None

OUT None

Description Operates at a VSYNC interrupt after call. Use IRQ routine in BIOS.

[$31] EX_SPROFF: Sprite Display OFF (SB;R05 bit6)

IN None

OUT None

Description Operates at a VSYNC interrupt after call. Use IRQ routine in BIOS.

[$32] EX_DSPON: Background and Sprite Display ON (SB,BB)

IN None

OUT None

Description Operates at a VSYNC interrupt after call. Use IRQ routine in BIOS.

[$33] EX_DSPOFF: Background and Sprite Display OFF (SB,BB)

IN	None
OUT	None
Description	Operates at a VSYNC interrupt after call. Use IRQ routine in BIOS.

[$34] EX_DMAMOD: Set Operation Mode for DMA Transfer between VRAM & SATB and VRAMs. (DCR;R0F)

IN	AREG:	DMA mode
OUT	None	

[$35] EX_SPRDMA: Set Sauce Address and Transfer Between VRAM and SATB (DVSSR;R13)

IN	SAT_ADR:	SAT BUFF ADR L
	SAT_ADR+1:	SAT BUFF ADR H
OUT	None	
Description	Use IRQ routine in BIOS.	

[$36] EX_SATCLR: Clear Sprite Attribute Table (SAT) in VRAM

IN	SAT_ADR:	SAT BUFF ADR L
	SAT_ADR+1:	SAT BUFF ADR H
OUT	SPRPTR = 0	
Description	Set VRAM increment width to 1.	

[$37] EX_SPRPUT: Set Sprite Attribute to Where Specified by SAT in VRAM

IN	SPRPTR:	Definition number
	SPRYL, SPRYH:	Y-coordinate
	SPRXL, SPRXH:	X-coordinate
	SPRNL, SPRNH:	Pattern number
	SPRAL, SPRAH:	Attribute, Color
	SATADR:	SAT BUFF ADR L
	SAT_ADR+1	SAT BUFF ADR H
OUT	SPRPTR +=1	
Description	Set VRAM increment width to 1.	

[$38] EX_SETRCR: Set Raster Detection Register (RCR;R06) Value

IN	AREG:	RCR L
	XREG:	RCR H

OUT None

[$39] EX_SETRED: Set to VRAM Read Mode

IN	AREG:	VRAM READ START ADR L
	XREG:	VRAM READ START ADR H

OUT None

[$3A] EX_SETWRT: Set to VRAM Write Mode

IN	AREG:	VRAM WRITE START ADR L
	XREG:	VRAM WRITE START ADR H

OUT None

[$3B] EX_SETDMA: Set Source(SOUR;R10), Destination Address(DESR;R11), and Transfer Block Length(LENR;R12) for DMA Transfer between VRAMs.

IN	_BX:	SOURCE ADDRESS
	_DX:	DESTINATION ADDRESS
	_CX:	BLOCK LENGTH

OUT None

[$4C] EX_COLORCMD: Set and Read Color Palette

IN	COLOR_CMD:	Function	
		$00	Do nothing
		$01	Read color palette
		$02	Set color palette
	BGC_PTR:	Background color data buffer address L	
	BGC_PTR+1:	Background color data buffer address H	
	BGC_LEN:	No. of background color palette	
	SPRC_PTR:	Sprite color data buffer address L	
	SPRC_PTR+1:	Sprite color data buffer address H	
	SPRC_LEN:	No. of sprite color palette	
OUT	COLOR_CMD:	$00	

About color palette setting

If IRQ routine in BIOS is used , color palette cannot be set through the above EX_COLORCMD method but in the following method.

Set the following values in the BIOS work area:

BGC_PTR:	Background color data buffer address L	
BGC_PTR+1:	Background color data buffer address H	
BGC_LEN:	No. of background color palette	
SPRC_PTR:	Sprite color data buffer address L	
SPRC_PTR+1:	Sprite color data buffer address H	
SPRC_LEN:	No. of sprite color palette	
COLOR_CMD:	Set function code	
	0:	Do nothing
	1:	Read color palette
	2:	Set color palette

Operates at a VCYNC interrupt after COLOR_CMD is set. COLOR_CMD will be cleared to 0 when the operation is completed.

[$3C] EX_BINBCD: Switch Binary Value to BCD Value

IN	AREG:	Binary data	
OUT	AREG:	BCD data	
	CARRY:	0	OK
		1	ERROR

[$3D] EX_BCDBIN: Switch BCD Value to Binary Value

IN AREG: BCD data

OUT AREG: Binary data
CARRY: 0 OK
1 ERROR

[$3E] EX_RND: Get Random Number

IN None

OUT AREG: Random number

Description The following system values need to be asynchronously changed to get a random number using this call.

RANDSEED	It will be incremented by each VSYNC interrupt if using IRQ in BIOS.
RNDM	Must be changed by user.

[$3F] MA_MUL8U: 8bit Multiplication with Unsigned Bit

8bit * 8bit = 16bit

IN _AL: Multiplicand
_BL: Multiplier

OUT _CX: Result

[$40] MA_MUL8U: 8bit Multiplication with Signed Bit

8bit * 8bit = 16bit

IN _AL: Multiplicand
_BL: Multiplier

OUT _CX: Result

[$41] MA_MUL16U: 16bit Multiplication with Unsigned Bit

16bit * 16bit = 32bit

IN _AX: Multiplicand
_BX: Multiplier

OUT _CX(L):_DX(H)

[$42] MA_DIV16U: 16bit Division with Unsigned Bit

16bit / 16bit = 16bit

IN	_AX:	Dividend
	_BX:	Divisor
OUT	_CX:	Quotient
	_DX:	Remainder

[$43] MA_DIV16S: 16bit Division with Signed Bit

16bit / 16bit = 16bit

IN	_AX:	Dividend
	_BX:	Divisor
OUT	_CX:	Quotient
	_DX:	Remainder

[$44] MA_SQRT: Square Root

IN	_AX:	Cardinal number
OUT	_CL:	Root

[$45] MA_SIN: Get Sine

IN	AREG:	Degree (DEGREE: 0° ~ 90°)	
OUT	CARRY:	0	AREG = AREG/256 Value
		1	RESULT = 256

[$46] MA_COS: Get Cosine

IN	AREG:	Degree (DEGREE: 0° ~ 90°)	
OUT	CARRY:	0	AREG = AREG/256 Value
		1	RESULT = 256

[$47] MA_ATNI: Get ATN

IN	AREG:	Y/X*64
OUT	AREG:	Value between 0° ~ 45°

[$48] PSG_BIOS: PSG Driver

IN _DH: Function number ($00 ~ $14)

Description See PSG driver document.

[$49] GRP_BIOS: Graphic Driver

IN _DH: Function number ($00 ~ $0F)

_DH

$00:VI_GINIT Initialize graphic BIOS

IN _AX: Graphic starting address
_BX: Paint stack area
OUT None

$01:VI_CASHCLR Clear graphic pointer cash. This is usually not used if initialized by VI_GINIT, shown above, which clears cash automatically.

IN None
OUT None

$02:VI_STRTADR Get graphic starting address and stack pointer.

IN None
OUT _AX: Graphic starting address
_BX: Stack pointer

$03:VI_GETADRS Get actual address of the coordinate address

IN _AH: X-coordinate
_AL: Y-coordinate
OUT _AX: Actual address
_BH: Bit number of byte (Count from 7th bit)

$04:VI_CLS Clear a 4 screen area currently specified by BAT top address for graphics.

IN None
OUT None

$05:VI_PSET Dot at the specified coordinates

IN	_AH:	X-coordinate
	_AL:	Y-coordinate
	_DL:	Color code (0 ~ 15)
OUT	None	

$06:VI_POINT Get color of a specified coordinate

IN	_AH:	X-coordinate
	_AL:	Y-coordinate
OUT	AREG:	Color code of the coordinate

$07:VI_LINE Draw a line between specified coordinates

IN	_AH:	X-coordinate of start point
	_AL:	Y-coordinate of start point
	_BH:	X-coordinate of end point
	_BL:	Y-coordinate of end point
	_DL:	Color code of line
OUT	None	

$08:VI_BOX Draw a box using the specified coordinates as a diagonal line

IN	_AH:	X-coordinate of start point
	_AL:	Y-coordinate of start point
	_BH:	X-coordinate of end point
	_BL:	Y-coordinate of end point
	_DL:	Color code of line
OUT	None	

$09:VI_BOXF Draw a box using the specified coordinates as a diagonal line and fill

IN	_AH:	X-coordinate of start point
	_AL:	Y-coordinate of start point
	_BH:	X-coordinate of end point
	_BL:	Y-coordinate of end point
	_DL:	Color code of line and fill
OUT	None	

$0A:VI_FLOOD Paint from the specified coordinate

IN _AH: X-coordinate of start point
_AL: Y-coordinate of start point
_DL: Color code of paint

OUT None

$0B:VI_PAINT Paint from the specified coordinate (different from the above FLOOD). Border line is specified and filled.

IN _AH: X-coordinate
_AL: Y-coordinate
_BH: Border line bit (H)
_BL: Border line bit (L)
_DL: Color code of paint

OUT None

$0C:VI_GWINDOW Set window at the specified coordinate

IN _AH: X-coordinate of start point (character unit)
_AL: Y-coordinate of start point (character unit)
_BH:
_BL:
_CH: No. of characters in X direction
_CL: No. of characters in Y direction
_DL: Color bank (not code)

OUT None

$0D:VI_GFONT Select font type for PUTFONT and SYMBOL output

IN _AL: Select type
0: 16 dot font
1: 12 dot font
2: 8 dot font
4: User defined character

OUT None

$0E:VI_PUTFONT Display Kanji at the specified coordinate

IN _AH: X-coordinate
_AL: Y-coordinate
_BX: Character font number
_CL: Select back color and mode
bit 0 ¯ 3: Back color
bit 6: Back transparent (1/0)
bit 7: Fore transparent (1/0)
_DL: Fore color

OUT None

$0F:VI_SYMBOL Display magnified specified character font

IN _AH: X-coordinate
_AL: Y-coordinate
_BX: Character font number
_CL: Select back color and mode
bit 0 ¯ 3: Back color
bit 4 ¯ 5: Font (00:16dot, 01:12dot, 10:8dot, 11:user)
bit 6: Back transparent (1/0)
bit 7: Fore transparent (1/0)
_CH: XY magnifying rate (x,y = 4/4bit, LOW/HI)
_DL: Fore color

OUT None

[SUB ERROR CODE LIST]

$00 NO SENS
Does not have a valid sense byte. (Does not contain error information)

$04 NOT READY
CD-ROM drive not ready.

$0B NO DISC
CD not inserted.

$0D OPEN
Cover is open.

$11 DATA FIELD INCORRECT
More than one uncorrectable error occurred in one block.

$15 SEEK ERROR
Seek operation dod not complete within the specified time.

$16 HEADER READ ERROR
Cannot detect objective CD-ROM header.

$1C NOT DIGITAL AUDIO TRACK
Seek object address of the AUDIO TRACK SEARCH command is located within the data area.

$1D NOT CDROM DATA TRACK
Read address is not located within the CD-ROM data area.

$20 INVALID COMMAND
Received an unsupported command. (CDB top byte)

$21 INVALID ADDRESS
Invalid address or value within the CDB. (BCD check)

$22 INVALID PARAMETER
Invalid parameter within the CDB. (Other than an unused value or a specified value)

$25 END OF VOLUME
Specified address exceeds disk end location.

$2A INVALID PARAMETER LIST
Invalid value within the parameter list.

$2C NOT AUDIO PLAY STATE
Audio track not in play.

Note: CDB is an abbreviation of Command Data Byte.

```
;================================================================
;
;    PC Engine CD-ROM SYSTEM BIOS  version 1.00
;
;    BIOS ENTRY and WORK equ
;
;    880913  Takaki Kobayashi
;
;=================================================================
;
max_mapping    equ       $FFF5

;-----------------------------------------------------------------
;    BIOS ENTRY
;
cd_boot        equ       $E000
cd_reset       equ       $E003
cd_base        equ       $E006
cd_read        equ       $E009
cd_seek        equ       $E00C
cd_exec        equ       $E00F
cd_play        equ       $E012
cd_search      equ       $E015
cd_pause       equ       $E018
cd_stat        equ       $E01B
cd_subq        equ       $E01E
cd_dinfo       equ       $E021
cd_contnts     equ       $E024
cd_subrd       equ       $E027
cd_pcmrd       equ       $E02A
cd_fade        equ       $E02D

ad_reset       equ       $E030
ad_trans       equ       $E033
ad_read        equ       $E036
ad_write       equ       $E039
ad_play        equ       $E03C
ad_cplay       equ       $E03F
ad_stop        equ       $E042
ad_stat        equ       $E045

bm_format      equ       $E048
bm_free        equ       $E04B
bm_read        equ       $E04E
bm_write       equ       $E051
bm_delete      equ       $E054
bm_files       equ       $E057

ex_getver      equ       $E05A
ex_setvec      equ       $E05D
ex_getfnt      equ       $E060
ex_joysns      equ       $E063
ex_joyrep      equ       $E066
ex_scrsiz      equ       $E069
```

```
ex_dotmod       equ        $E06C
ex_scrmod       equ        $E06F
ex_imode        equ        $E072
ex_vmode        equ        $E075
ex_hmode        equ        $E078
ex_vsync        equ        $E07B
ex_rcron        equ        $E07E
ex_rcroff       equ        $E081
ex_irqon        equ        $E084
ex_irqoff       equ        $E087
ex_bgon         equ        $E08A
ex_bgoff        equ        $E08D
ex_spron        equ        $E090
ex_sproff       equ        $E093
ex_dspon        equ        $E096
ex_dspoff       equ        $E099
ex_dmamod       equ        $E09C
ex_sprdma       equ        $E09F
ex_satclr       equ        $E0A2
ex_sprput       equ        $E0A5
ex_setrcr       equ        $E0A8
ex_setred       equ        $E0AB
ex_setwrt       equ        $E0AE
ex_setdma       equ        $E0B1
ex_colorcmd     equ        $E0E4
ex_binbcd       equ        $E0B4
ex_bcdbin       equ        $E0B7
ex_rnd          equ        $E0BA

ma_mul8u        equ        $E0BD
ma_mul8s        equ        $E0C0
ma_mul16u       equ        $E0C3
ma_div16s       equ        $E0C6
ma_div16u       equ        $E0C9
ma_sqrt         equ        $E0CC
ma_sin          equ        $E0CF
ma_cos          equ        $E0D2
ma_atni         equ        $E0D5

psg_bios        equ        $E0D8
grp_bios        equ        $E0DB

psg_drive       equ        $E0E1

;----------------------------------------------------------------
;     ZERO PAGE WORK
;
zpgtop          equ        $20DC
;
zpg_grp_top     equ        $20DC
vi_bitpat       equ        $20DC            ;1
vi_rvbitpat     equ        $20DD            ;1
vi_ft_front     equ        $20DE
vi_padrs        equ        $20DE            ;2
```

```
vi_porg         equ         $20E0               ;1
vi_ft_back      equ         $20E1               ;3
vi_stack        equ         $20E4               ;2
;
zpg_psg_top     equ         $20E6
time_sw         equ         $20E6               ;1
main_sw         equ         $20E7               ;1
si              equ         $20E8
si_l            equ         $20E8               ;1
si_h            equ         $20E9               ;1
r0              equ         $20EA
r0_l            equ         $20EA               ;1
r0_h            equ         $20EB               ;1
;
zpg_sys_top     equ         $20EC
zx0             equ         $20EC
zl0             equ         $20EC               ;1
zh0             equ         $20ED               ;1
zx1             equ         $20EE
zl1             equ         $20EE               ;1
zh1             equ         $20EF               ;1
zx2             equ         $20F0
zl2             equ         $20F0               ;1
zh2             equ         $20F1               ;1
cdi_b           equ         $20F2               ;1
crl_m           equ         $20F3               ;1
crh_m           equ         $20F4               ;1
irq_m           equ         $20F5               ;1
str_b           equ         $20F6               ;1
reg_box         equ         $20F7               ;1
_ax             equ         $20F8
_al             equ         $20F8               ;1
_ah             equ         $20F9               ;1
_bx             equ         $20FA
_bl             equ         $20FA               ;1
_bh             equ         $20FB               ;1
_cx             equ         $20FC
_cl             equ         $20FC               ;1
_ch             equ         $20FD               ;1
_dx             equ         $20FE
_dl             equ         $20FE               ;1
_dh             equ         $20FF               ;1

;----------------------------------------------------------------
;    RAM WORK
;
ramtop          equ         $2200
;
usrvec          equ         $2200
irq2_jmp        equ         $2200               ;2
irq_jmp         equ         $2202               ;2
tim_jmp         equ         $2204               ;2
nmi_jmp         equ         $2206               ;2
sync_jmp        equ         $2208               ;2
```

```
rcr_jmp        equ       $220A          ;2
bgx1           equ       $220C          ;2
bgx2           equ       $220E          ;2
bgy1           equ       $2210          ;2
bgy2           equ       $2212          ;2
sat_adr        equ       $2214          ;2
sprptr         equ       $2216          ;1
spryl          equ       $2217          ;1
spryh          equ       $2218          ;1
sprxl          equ       $2219          ;1
sprxh          equ       $221A          ;1
sprnl          equ       $221B          ;1
sprnh          equ       $221C          ;1
spral          equ       $221D          ;1
sprah          equ       $221E          ;1
color_cmd      equ       $221F          ;1
bgc_ptr        equ       $2220          ;2
bgc_len        equ       $2222          ;1
sprc_ptr       equ       $2023          ;2
sprc_len       equ       $2025          ;1
joykeyflg      equ       $2226          ;1
joyena         equ       $2227          ;1
joy            equ       $2228          ;5
joytrg         equ       $222D          ;5
joyold         equ       $2232          ;5
irq_cnt        equ       $2241          ;1
notrdyflg      equ       $2247          ;1
rndseed        equ       $2249          ;1
rndl           equ       $2249          ;1
rndh           equ       $224A          ;1
rndm           equ       $224B          ;1
tnomin         equ       $226A          ;1
tnomax         equ       $226B          ;1
outmin         equ       $226C          ;1
outsec         equ       $226D          ;1
outfrm         equ       $226E          ;1
vdtin_flg      equ       $2272          ;1
recbase0_h     equ       $2274          ;1
recbase0_m     equ       $2275          ;1
recbase0_l     equ       $2776          ;1
recbase1_h     equ       $2277          ;1
recbase1_m     equ       $2278          ;1
recbase1_l     equ       $2279          ;1
scsists        equ       $227B          ;1
suberrc        equ       $227C          ;1
sibcode        equ       $227E          ;1

ramend         equ       $22D0

psg_work_top   equ       $22D0
graph_work_top equ       $2616
key_work_top   equ       $2649
user_work_top  equ       $267F
;-----------------------------------------------------------------
```

```
;     PSG_BIOS FUNCTION NUMBER
;
PSG_ON           equ         0
PSG_OFF          equ         1
PSG_INIT         equ         2
PSG_BANK         equ         3
PSG_TRACK        equ         4
PSG_WAVE         equ         5
PSG_ENV          equ         6
PSG_FM           equ         7
PSG_PE           equ         8
PSG_PC           equ         9
PSG_TEMPO        equ         10
PSG_PLAY         equ         11
PSG_MSTAT        equ         12
PSG_SSTAT        equ         13
PSG_MSTOP        equ         14
PSG_SSTOP        equ         15
PSG_ASTOP        equ         16
PSG_MVOFF        equ         17
PSG_CONT         equ         18
PSG_FDOUT        equ         19
PSG_DCNT         equ         20

;--------------------------------------------------------------
;     GRP_BIOS FUNCTION NUMBER
;
VI_GINIT         equ         0
VI_CASHCLR       equ         1
VI_STRTADR       equ         2
VI_GETADRS       equ         3
VI_CLS           equ         4
VI_PSET          equ         5
VI_POINT         equ         6
VI_LINE          equ         7
VI_BOX           equ         8
VI_BOXF          equ         9
VI_FLOOD         equ         10
VI_PAINT         equ         11
VI_GWINDOW       equ         12
VI_GFONT         equ         13
VI_PUTFONT       equ         14
VI_SYMBOL        equ         15
;--------------------------------------------------------------
```