Programmer Board Set (DTL-H2000)

Developer

Reference

Series

September 1995

The Development System Tool to which this manual relates is supplied pursuant to and subject to the terms of the Sony PlayStation Licensed Developer Agreement.

This manual is intended for distribution to and use only by Sony PlayStation Licensed Developers in accordance with the Sony PlayStation Licensed Developer Agreement. The information in this manual is subject to change without notice.

Published December 1994.

Sony Electronic Publishing Ltd
13 Great Marlborough Street
London
W1V 2LP

Tel No: +44 (0) 171 911 8900 (switchboard)

Contents

6 KANJI ROM SPECIFICATION

7 CPE2X

8 PSXCONS

DTL-H2000

1 DTL-H2000

1.1 About this Document

Based on an extract taken from the DTL-D2130 Psy-Q Programmer's Guide Version 2.0, this document gives information about porting the development environment from DTL-H500 (target box) to DTL-H2000.

1.2 Porting Operation

If you are using our development environment with the DTL—H500 (target box), please carry out the following operation in order to port to DTL-H2000.

① **Installing DTL-H2000**
Please refer to the attached manual in detail.

② **Installing Runtime Library Version 2.0**
Please refer to the attached manual in detail.

③ **Installing Psy-Q Update Disk, which is in the same package as DTL-S2160**
Please replace PSYBIOS.COM, which has been set up to now, with DEXBIOS.COM, which is on the same disk. Please also change the option name according to the H2000 setting.

④ **Installing Library Update in this Disk**
Please copy the file according to the instructions in README.DOC.

You can use software such as Compiler Debugger without any change.

1.3 Registration of DEXBIOS

Option setting of DEXBIOS, which is based on an ex-factory setting, is as follows. If you change the board setting , please change the Options as explained in number [5] below.

```
DEXBIOS /a1340
```

1.4 Confirming Installation

① **Rebooting the PC**
Please reboot the PC when installation is completed.

② **Confirmation of Hardware Function**
Please execute RUN.EXE without the argument and confirm whether or not the hardware is functioning correctly. The following message will be displayed when the target board is recognised.

[Example]

```
C:\>RUN
RUN.EXE Executable/Binary file downloader version 2.23
Copyright (c)  1992,93 S N Systems Ltd.
Target 0 is R3000 - SONY_PSX4.00
No file to proceed.
```

1.5 System Driver DEXBIOS.COM

DEXBIOS.COM is a permanently resident program that handles the target board installed in the PC, the PC-AT itself and the Psy-Q system. DEXBIOS.COM must be permanently resident in order to use the Psy-Q system.

It is usually executed within AUTOEXEC.BAT and loaded automatically when the PC is activated.

Form

Input the following against the MS-DOS Prompt in order to activate DEXBIOS.

```
DEXBIOS [/option]
```

The option is given after the "/" and in the case of multiple options, each option is separated by a space.

Options

The following options can be given in DEXBIOS.COM after "/". In the case of multiple options, each option is separated by a space. The default value is used if nothing is input.

In its ex-factory condition, the target board has been set as far as possible so as not to clash with other boards and interfaces etc. Therefore, you normally designate as follows.

[Example]

Designation based on the ex-factory setting

```
DEXBIOS /a1340
```

If you change the board setting and install it in the PC, please input the additional option line to AUTOEXEC.BAT as described below.

Card Address (/a)

Input the card address after "/a" with numbers in base sixteen. The 12bit above the card address can be input with the dipswitch on top of the board. The I/O address needs 32 bytes of space.

[Example]

```
DEXBIOS /a1300
```

File Transmission Buffer Size (/b)

Please designate the file transmission buffer size after "/b" by numbers in base of ten. The maximum size that may be given is 232 (KB units).

This option designates the size of the transmission buffer used when the target board accesses a file on the PC. Although access becomes faster if you enlarge the size of the buffer, the amount of memory used for that part increases. There is normally no need to make a special designation.

[Example]

```
DEXBIOS /b2
```

IRQ Number (/i)

Please give the IRQ number after "/i". IRQ numbers can be designated as 10, 11, 12 or 15, or base ten thereof. If you designate "0" it becomes OFF and functions without using IRQ.

[Example]

```
DEXBIOS /i15
```

Note

If you reactivate DEXBIOS without an Option, the permanently resident image will be deleted once from the memory (released from permanent residence). In this way you can change the Option without rebooting the PC.

1.6 H2000 Activation Mode

H2000 will function in the PlayStation Platform Mode if you switch on the PC/AT (or reset). This is the mode to execute the boot sequence from the CD-ROM drive, and is a convenient design for the purpose of demonstration. For the purpose of development, the target board must be changed to a condition suitable for debugging. Therefore, DEXBIOS automatically resets the H2000 and reactivates in the debug mode.

Mode Number	Action
0	**PlayStation Platform Mode** Runs the boot sequence from CD-ROM
1	**Debug Mode** Please designate this mode for the purpose of development.
2	**Debug Monitor Mode** It is possible to use various commands of the debug monitor, which is inside the target board.
3	**Color Bar Display Mode** The color bar is displayed on the video screen.

1.7 Target Board Boot Program RESETPS

The RESETPS.EXE program is used to carry out this reactivation clearly from the DOS command mode. Because of this the mode can be changed simply from DOS without touching the dipswitch or other hardware.

Input the following in order to put the target board into debug mode.

```
RESETPS 1
```

DEXBIOS must be permanently resident in order to execute RESTPS. When adding to AUTOEXEC.BAT, please make sure you add this after the DEXBIOS line.

Form

Please input the following at the MS-DOS Prompt in order to execute RESETPS. Also, DEXBIOS must be permanently resident in order to execute RESETPS.

```
RESETPS <mode number> [/a<address>]
```

The target board address number can be given as an option after the execution mode number. This is done in the same way as the “/a” option of DEXBIOS. However, this is a special option for designating only when DEXBIOS is not permanently resident, and there is normally no need to do so.

DTL-H2000 Usage Guide

2 DTL-H2000 USAGE GUIDE

2.1 Specifications

DTL-H2000 (hereafter "H2000") is a development environment equipped with all functions necessary for the PlayStation(tm) Game Machine (hereafter "Game Machine"). It comprises two full size ISA boards for PC compatible machine. It has the following specifications and functions.

CPU-GTE	Same specification as the Game Machine
Main memory	8MByte (access can be limited to 2MByte)
Graphics	Same specification as the Game Machine
Sound	Same specification as the Game Machine
CD-ROM sub-system	Equivalent to the Game Machine (without copy protection)
Controller terminal	Equivalent to the Game Machine (connector shape is different)
SIO/PIO extension	Equivalent to the Game Machine (connector shape is different)
CD-ROM drive terminal	For DTL-H2010

2.2 Warning with regard to the use of software

Please do not execute PCinit() in LIBSN. Automatic execution takes place on initialisation.

The kernel that is in ROM is host compatible with the one which is installed in DTL-H500.

The following devices have been specially added for the purpose of extending the development environment. These functions are not available in the Game Machine.

Message window driver (for standard output)
DOS file driver (for CD-ROM simulator extension)

The kernel ROM patch program, patchx.cpe, is stored on disk. The following functions are added on execution of this program. Please read the \patch\readme.doc\ for details.

memory card file system
Kanji (Chinese character) font bit pattern

The following drivers are provided by BBS.
extension SIO driver (for competition cable)
mouse driver

2.3 Q & A

① **What library versions can be used on the H2000?**
Version 2.0 and all later library versions will operate. Some functions may be used only after execution of the patch program.

② **What are the differences between the H2000 and the Game Machine?**
After execution of "patchx.cpe", they are the same from the library specifications.
In addition, copy protection test and the SCE company name logo are activated immediately after resetting or power on. There are also plans to provide on ROM a CD player screen for listening to audio CD and a copy screen for managing the memory card (operating in the RAM user area). These are not loaded on the H2000.

③ **How do you move from the previous target box (DTL-H500) to the H2000?**
The compiler, debugger and then the program loader (RUN.EXE) can still be used. The only thing that needs to be changed is the resident program which mediates between the module and the boards and boxes. Please replace the previous PSYBIOS.COM with DEXBIOS.COM which is included in the DTL-S2100.

④ Does the H2000 have a dipswitch like the target box?
Once installation is completed, there is no need to operate any switches on the board. All the options formerly set by switches can now be operated either from within DOS or the R3000 program.

Valid memory size

SetMem() has been added to the kernel library. Please adjust the setting in the following way.

```
SetMem(limit)
unsigned long limit; /*Mbyte unit RAM size (must be 2or8) */
```

Function Designates the valid memory size.

Booting from CD-ROM

When DTL-H2010 is connected, the PlayStation is booted from there immediately after power on. The disk format which can carry out boot is standard ISO-9660 level 1 and has root directory PSX.EXE format (refer to the manual for Runtime Liberary Version 2.0) file PSX.EXE;1 in the root directory. Convertor CPE2X.EXE (operating on DOS), from CPE to PS-X EXE, is stored “H2000 Update disk #1”.

Debug mode

DEXBIOS.COM resets H2000 and switches to debug mode. This can also be carried out in RESETPS.EXE (included in DTL-S2100).

Can printf() be used in H2000?

It can be used. There is a “message window” function in the Psy-Q debugger. The active window can be made into a message window by selecting MSG (message) from Set Type of the pulldown window. This window can be opened multiple times and the message can be displayed using PCwrite from within the program. H2000’s standard output (Descriptor 1) is assigned to the first message window. Therefore the character string can be output by printf() etc. The following device driver, which calls PSwrite internally, has been added in order to mount the above function.

Device name	:	mwin
File format	:	mwinXY: X*16+Y (window number - 1)
Block size	:	1 byte
Note	:	Only write is valid Output data is lost if no corresponding window is provided.
Example of OPEN	:	open(“mwin02:”,O_WRONLY); open message window #3

What happens if printf() is executed in the Game Machine?

Nothing happens. The printf() code itself is loaded in the Game Machine, but standard output is assigned to NULL DEVICE, so there is “no harm”.

How do you access DOS files?

Previous access by PCopen() etc. can be used just as it is. The following device driver has also been added. The usual open()/read()/write()/close() can be used by using this driver, which calls PCopen() etc. internally. Therefore, the greatest data volume that can be read and written at one time is 65,534 bytes. Moreover, PCinit does not need to be executed. The above two methods are different in open mode. PCopen() receives the macro from within libsn.h and open() receives it from within sys\file.h.

Device name	:	sim
File format	:	sim:<DOS path name> no distinction between case
Directory structure	:	same as DOS
Block size	:	1 byte
Note	:	write is possible. If O_CREAT has not been designated and if the target file does not exist, ERROR is returned by open().
Example of open	:	open(“sim:a:\test\data.txt”,O_RDONLY); opens DOS file a:\test\data.txt.

How do you change the system configuration?

By definition, the system configuration is read from the file SYSTEM.CNF;1 at the time of boot. As this is incovenient at the time of the test run, the following group of functions has been added to the kernel library from the H2000. The configuration can be changed by using these.

```
SetConf( ev tcb, sp )
   unsigned long ev;    /*number of EvCB*/
   unsigned long tcb;    /*number of TCB*/
   unsigned long sp;    /*initial value of stack pointer*/
```

Function Setting the system configuration

```
GetConf( ev tcb, sp )
   unsigned long *ev;    /*number of EvCB*/
   unsigned long *tcb;   /*number of TCB*/
   unsigned long *sp;    /*initial value of stack pointer*/
```

Function Getting the system configuration

What is the upper limit of number of files that can be open at the same time?

16 files. This is fixed.

What system resources are used by the system?

Root Counter

It was stated in the Version 1.0 manual that both root counter # 2 (built-in counter) and root counter # 3 (vertical synchronization) are used for controller and memory card management. However, it has now been decided to use only root counter # 3 for managing the controller (including the mouse) and memory card, and this is applied retroactively to Version 1.0.

Events

The system module, which takes possession of events immediately after activation, is the CD-ROM file system (device driver format). It absorbs 6 events (events brought about by LIBCD apart from UNKNOWN). There are no other drivers that take possession of events or that are installed by default. Therefore by default the user is left with 10 events. With drivers released after this, both the memory card file system and the extension SIO driver take possession of multiple events. These are not installed by default and are registered in the kernel by calling the initialization function. At the same time, they open the event that is to be used internally. The number of events that are used is expressly stated at the time of each release.

How do you switch the stack during the program?

By using setjmp/longjmp.

As is clear from looking at setjmp.h, these functions are implemented by clearing out the register contents to the jump buffer given as the argument and returning them to the register. The stack pointer register (SP) is included in the operation register. Therefore, by overwriting the JB_SP elements (the macro within setjmp.h) of the jump buffer that has carried out setjmp, the value of the stack pointer, which is written back in the register in the next longjmp, can be changed. This method depends on the structure of the jump buffer, but providing the differences in the contents of setjmp.h can be reflected accurately, it can be used in many processing systems, including of course the PlayStation development environment.

By using SetSp.

SetSp(), which is among the other services of LIBAPI, can be used as a method specifically for PlayStation. This function takes the new stack pointer's value as the argument and returns the preceeding value. Whichever the case, the contents of the automatic variable will be undefined if you change the stack pointer. Please use an external variable in order to keep the former value.

What memory size can be used in default?

8M bytes.

During the program it is possible to create a bus error when anything other than 2M bytes, which is the same as in the final machine, is accessed. This method is explained in (14).

The initial value of the stack pointer is 80100000.

This phenomenon appears when you use the former libsn.lib, but it can be changed to 80800000 by linking 8mbyte.obj. Although this initial value is different from that stated in the document, this is regarded as the normal initial value. Forming a pair with this object is 2mbyte.obj. When this is linked, the stack pointer's initial value becomes 80200000.

How do you carry out an operating test with the Game Machine's memory size?

First compile the following program, and link with 2mbyte.obj.

```
main()
{
SetMem(2);
}
```

The valid memory size becomes 2M bytes if you execute this module. The setting is valid until H2000 is reset or SetMem() is executed again. The module to be tested is linked with 2mbyte.obj and not 8mbyte.obj.

Patch Module for DTL-H2000

Patch Module for DTL-H2000

3 PATCH MODULE FOR DTL-H2000

3.1 Overview

The kernel which is stored in DTL-H2000 ROM has the following differences from the actual game machine, and the purpose of this patch module is to correct these.

① The code related to the Memory Card is not installed
② Kanji (Chinese character) fonts are not installed
③ The undisclosed system call related to the trial disk is not installed
④ The ISO9660 file system and libcd cannot coexist

3.2 File Structure

patchX.cpe	the patch module itself
patchX.doc	this document
libapi.libkernel	library update
kernel.hheader	update including kernel library
errno.h header	update including error code macro definition

3.3 Contents of Change

The following specification changes are applied. The patch is carried out in the kernel reserved area and the patch exclusive memory inside the board. There is no change in the user area.

- Correction of the malfunction whereby DEXBIOS is removed on execution of exit()
- Correction of the ResetRCnt() malfunction
- Addition of kanji (Chinese character) font data
- Correction of the kernel.h macro definition bug (related to the root counter)
- Addition of functions, such as those related to Memory Card, which appear as system call
- Correction of the malfunction whereby the ISO9660 file system and libcd cannot co-exist
  However the following process is necessary for them to coexist

```
CdInit();
_96_init();
```

Following this, open()/read/firstfile() etc. can be used together with the libcd function for device “cdrom”,

Validity of patch

Patch achieves its objective by changing the code in RAM. Therefore, RESET (by resetps and dexbios) and POWER OFF become invalid.

Relationship with other patch modules

No changes are carried out apart from these above. Therefore standard output is assigned to the debugger's message window immediately after patch. Because of such problems as action speed, if the action of standard output is stopped, please proceed with execution of the patch module noprint.cpe, whose release process has already been completed.

Information lost by patch

This module carries out a restructuring of the kernel. Therefore, all descriptors, heap management information and changes in kernel structure that were applied before execution become invalid.
Please carry out adjustments with SetConf() immediately after execution of this module, if this is being used.

3.4 Procedure Given a Non-recoverable Error

Please call exit() if an error is detected on the actual game action machine that cannot be dealt with. In the actual game machine kernel, SystemError() is recalled by internal exit(). This is the system call that displays the system error occurrence screen added by this patch. An exit() argument appears as a three figure number in a multiple of ten, and its value is displayed as an error recognition number. The error recognition letter is always 'S'. SCE reserves the numbers 7XX, 8XX and 9XX for error recognition numbers, and others are released for the application.
Recognition letters other than 'S' can be displayed by directly calling SystemError(). 'E' is released for the application in addition to 'S'.
When an SCE stipulated error occurs, please call exit() using the relevant recognition number as the argument.

SCE stipulated system errors

Error recognition
Letter Number Content

EXXX Released to the application

S 000-699 application errors
701 error occurrence with malloc
751 cannot open file
752 cannot read file
753 cannot write file
754 error occurrence in file operating functions
801 HwCPU/EvSpTRAP occurrence that cannot be dealt with
811 CD-ROM order register time-out
812 CD-ROM order execution time-out
813 HwCdRom/EvSpACK occurrence that cannot be dealt with
814 HwCdRom/EvSpCOMP occurrence that cannot be dealt with
815 HwCdRom/EvSpDR occurrence that cannot be dealt with
816 HwCdRom/EvSpDE occurrence that cannot be dealt with
817 HwCdRom/EvSpERROR occurrence that cannot be dealt with
818 HwCdRom/EvSpUNKNOWN occurrence that cannot be dealt with
820 SwCARD/EvSpIOE occurrence that cannot be dealt with
821 SwCARD/EvSpERROR occurrence that cannot be dealt with

822 SwCARD/EvSpNEW occurrence that cannot be dealt with
823 SwCARD/EvSpTIMOUT occurrence that cannot be dealt with
824 SwCARD/EvSpPERROR occurrence that cannot be dealt with
825 SwCARD/EvSpUNKNOWN occurrence that cannot be dealt with

Reference Material : Error codes for internal system identification

Error identification

Letter	Number	Content
B	001	adjusting conditions for copy protect
	901	error in boot sequence
	902	CD-ROM drive cannot be initialized
	903	device trouble at time of opening SYSTEM.CNF
	904	cannot restructure kernel
	905	cannot load first execution file
	906	data structure trouble with first execution file
	907	trouble during activation of first execution file
	908	cannot activate first execution file
	909	shell open during boot
	911	trouble at time of reading SYSTEM.CNF
	912	trouble at time of SYSTEM.CNF analysis
	913	irregularity in default structure data
	921	bus error related to extension PIO
X	000-015	CPU exceptions
D	011	CD-ROM order register time-out
	012	CD-ROM order execution time-out
	022	HwCdRom/EvSpERROR occurrence that cannot be recovered
	023	HwCdRom/EvSpDE occurrence that cannot be recovered
	031	CdGetStatus order register time-out
	032	CdGetStatus order execution error
	041	CD-ROM motor suspension order register time-out
	042	CD-ROM motor suspension order execution error

3.5 Concerning the Specification for Seek

It has become clear that with seek(), which is one of the input/output management services within the kernel library, action cannot be carried out to make the file end as a starting point. The current conditions are taken as the correct specification as the influence of the amendment is so extensive. Therefore the statement regarding seek() in the library reference is changed as follows.

Area of change : DTL-2150 Runtime Library Reference Vol.2 28p

Before

flag macro action

SEEK_SET	file head
SEEK_CUR	current position
SEEK_END	end of file

After

flag macro action

SEEK_SET	file head
SEEK_CUR	current position

3.6 Memory Card Driver

The memory card file system driver is incorporated in the kernel at the time of executing this patch. When the following initialization is carried out, it becomes possible to use such input/ output management services as open()/read()/write()/firstfile() for device “bu” as well as memory card BIOS functions such as _card_info. Please refer to the “Memory Card specification” for the file name and file header.

<When there is co-existence with the controller>
/* controller initialization*/
InitCARD(1)
StartCARD();
_bu_init():

<when there is no co-existence with the controller>
/* controller initialization */
InitCARD(0);
StartCARD();
_bu_init();

3.7 Additional Service Functions

Relationship with Memory Card

InitCARD

format void InitCARD(val)
long val;

argument val 0 : using without the controller 1 : using with the controller

Explanation

Initialize Memory Card BIOS and put it in a ready condition. A low level interface starting with _card can thereafter be used directly by activating BIOS with StartCARD().
As the Memory Card file system uses these interfaces internally, these functions must be carried out before _bu_init().
It does not influence the controller.

no return value

StartCARD

format void StartCARD(void)

no argument

Explanation

It moves the Memory Card BIOS initialized by InitCARD() to a ready condition.
It does not influence the controller.

no return value

StopCARD

format void StopCARD(void)

no argument

Explanation

Moves Memory Card BIOS to a stopped condition (same condition as immediately after InitCARD() execution).
It does not influence the controller.

no return value

_bu_init

format void _bu_init(void)

no argument

Explanation

It initializes the Memory Card file system.
The file system driver is registered in the kernel when this patch module is incorporated, but initialization routing is not carried out automatically. The file system must be initialized clearly with this function.

no return value

_card_info

format long _card_info(chan)
long chan;

argument chan port number*16 + card number
port number: A port _ 0 B port _ 1
card number:always 0

Explanation

Carries out the connection test of the memory card designated by “chan”.
The function returns immediately because of the asynchronous operation. Multiple returns cannot be carried out into the same card slot. Completion of the process is notified by EVENT.

Content of Cause Descriptor / Event Classification

SwCARD/EvSpIOE	connected
SwCARD/EvSpTIMOUT	not connected
SwCARD/EvSpNEW	there is no writing after connection
SwCARD/EvSpERROR	communication error occurrence

Returns 1 if process registration of the return value is successful, otherwise return 0.

_card_load

format long _card_load(chan)
long chan;

argument chan port number*16 + card number
port number: A port _ 0 B port _ 1
card number:always 0

Explanation

File management information is read into the file system for asynchronous access by the input/output management service. Execution must always take place before opening the file on the Memory Card with the O_NOWAIT mode. There is no need to execute repeatedly as long as the card does not change.
The function itself returns immediately because of the asynchronous processing. Multiple calls cannot be carried out into the same card slot. Finalisation of the process is notified by EVENT The time required is 2 VSYNC.
Always execute once before carrying out open() of O_NOWAIT mode.

Content of Cause Descriptor / Event Classification

SwCARD/EvSpIOE	reading completed
SwCARD/EvSpTIMOUT	not connected
SwCARD/EvSpNEW	uninitialized card
SwCARD/EvSpERROR	communication error occurrence

Returns 1 if process registration of the return value is successful, otherwise return 0.

_card_auto

Sets automatic initialization

Format long_card_auto(val)
long val;

Argument val 0: automatic initialization prohibited
1: automatic initialization permitted

Explanation

It sets up the automatic initialization function

Return value None

_card_clear

format long _card_clear(chan)
long chan;

argument chan port number*16 + card number
port number: A port _ 0 B port _ 1
card number:always 0

Explanation

Dummy writing is carried out in the card's system management area, and the condition is cleared (there is no writing after connection).
The function returns immediately because of the asynchronous processing. Multiple returns cannot be carried out into the same card slot. Finalisation of the process is notified by EVENT.

Content of Cause Descriptor / Event Classification

SwCARD/EvSpIOE	process completed
SwCARD/EvSpTIMOUT	not connected
SwCARD/EvSpERROR	communication error occurrence

Return 1 if process registration of the return value is successful, otherwise return 0.

_new_card

format void _new_card(void)

no argument

Explanation

Masks EvSpNEW events occurring immediately after _card_read() or _card_write().
Returns immediately, even though it is a synchronous function.

no return value

_card_status

format long _card_status(drv)
long drv
argument drv slot number 0 : left hand side 1 : right hand side

Explanation

It obtains the Memory Card BIOS status in each slot.
It returns immediately, even though it is a synchronous process function.

If the return value Memory Card BIOS is in the operating condition, one of the following values is returned.

Value conditions

0x01 no registration process
0x02 READ process registration
0x04 WRITE process registration
0x08 connection test process registration
0x11 no registration process (timeout occurred immediately before)
0x21 no registration process (communication error occurred immediately before)

_card_wait

format long _card_status(drv)
long drv

argument drv slot number 0 : facing left hand side 1 : facing right hand side

Explanation

Waits until there is no registration process for the slot designated by drv.

Always returns 1.

_card_chan

format long _card_chan(void)

no argument

Explanation

Returns the device number of the Memory Card which caused the event that occurred immediately beforehand. Whether open or test is connected, the return value changes if any event occurs. (normally the event occurs during VBLANK or after BIOS activation by the next VBLANK in the channel in which no card is set.)

The return value is a 2 figure device number being a multiple of 16

System error connection

SystemError

format void SystemError(c,n)
char c;
long n;

argument c error identification letters (only English letters)
n error identification code (0-999)

Explanation

Notifies system error occurrences to the operator by screen display.
Call is carried out in the game machine by exit().

It does not return if successful.

Input/output management service related

_get_errno

format long _get_errno(void)

no argument

Explanation

Checks the code of errors through all file descriptors.

return value error code

_get_error

format long _get_error(fd)
unsigned long fd;

argument fd file descriptor

Explanation

Returns the error code which occurred close to the designated file descriptor.
The error code is defined in sys/errno.h.

return value error code

Krom2RawAdd

format unsigned long Krom2RawAdd(sjiscode)
unsigned short sjiscode;

argument sjiscode shift JIS code

Explanation

Obtains the head address of the font pattern which relates to kanji character designated by sjiscode.
A 16X16 dot gothic body bitmap font is loaded.
It covers JIS non kanji and number 1 standard.
The font pattern storage format is as follows.

The byte of the top left hand edge of the pattern is the head. The byte of the top right hand follows. The bit ordering is MSB to LSB, from left to right.

#	0	#	1
#	2	#	3
…		…	
…		…	
…		…	
#	30	#	31

Return value: returns the head address of the kanji font pattern.
Returns -1 if there is no font data that relates to the designated kanji.

Memory Card Operation

5 MEMORY CARD OPERATION

5.1 File Name

Please make up the file name as follows

Byte	Content	Note
0	Magic	Always 'B'
1	Area	Domestic is '1' (*1)
2-11	Title	SCE product number (*2)
12-20	User disclosure	Only ASCII used

Finalised with 0x00

*1 : no check by system

*2 : multiple disc title is the head disc

The SCE product number is decided at our company's sales decision preparatory meeting (approximately 3 weeks before publication of the master), and is communicated to the relevant sales personnel in each company. Based on this, please make your decision as follows.

Example

If the product code is 'SLPS-00001' the first 12 characters of the file name will be 'BISCEX-00001'

* The number should always consist of 5 digits, using 0 to fill the spaces.

5.2 Procedure When Volume is Deficient

If volume deficiency occurs during execution of an application, the SCE standard screen and message (e.g. display by an icon the same as OSD is needed) is not fixed. The message in the material given out in seminars etc is only one example. Moreover, the memory card management screen in OSD is also only reference material. Please mount a suitable operation and display system for the save function and save screen of each title.

5.3 Terminology System

The necessary unit of memory volume is stated as 'block' in the merchandise catalogue. This is equivalent to 8K bytes. In the document released by our company's development department the same volume unit is called 'slot' and 'block' indicates 1 2 8 bytes, which is the access unit. Please use the term 'slot' if you make an enquiry by BBS etc.

5.4 Document Revisions

The following two major versions and one revision of the document about memory card have been released. They are explained here to confirm the main specification change.

File Size

The final file size unit is 8K byte 'slot'. At the time of CREAT the file, please designate the slot number to the top 16 bits of open mode (2 arguments). The file size becomes fixed after the CREAT. In the memory card management screen in OSD (ON ROM activation program), 1 icon indicates 1 slot, thereby communicating the volume condition to the user. According to the above rule, the icon and file size correspond exactly.

File Composition

At one time pad inserted one byte after the slot number. The following is the correct situation.

Item Size (bytes)

Header 128
  Magic 2 (always 'SC')
  Type 1
  Slot number 1
 Amendment point Document name 64 (shift JIS, *1)
 Amendment point pad 28
  CLUT 32

Icon image (1) 128 (16 x 16 x bits)
Icon image (2) 128 (Type==0x12,0x12,0x13 only)
Icon image (3) 128 (Type==0x13 only)

Data optional (128B x N)

* 1 : non kanji (not Chinese characters) and first standard kanji (Chinese characters) only

Type Icon image number
(automatic interchangeable animation)

0x111
0x12 2
0x13 3

5.5 Malfunction in Debugging Station

In the debugging station, which is to be supplied this year, the malfunction of the _card_clear () function inside the memory card library is confirmed. It is the bug of the event handler in the boot ROM which this function uses. There are no problems in Game Machine and PS board after patch. We intend to minimise hindrance to title development. More details will be informed later.

Memory Card Specification

4 MEMORY CARD SPECIFICATION

4.1 List of Specifications

Capacity

120K bytes at the time of format
(access by sector unit of 128 bytes)

Access Speed

(1) access cannot be made for 20ms after 1 sector has been written
(2) maximum continuous reading speed is approximately 10K bytes per second

Other Information

It can be inserted and removed without lowering power supply
Writing is guaranteed for 100,000 writes
20 replacement sectors (held within file system)

4.2 BIOS (on ROM)

BIOS, which handles one sector in 2 VBLANK, has been provided so that controller reading and access to the two connectors A & B may co-exist. Its weakness is that it cannot carry out continuous reading of one connector at maximum speed.

Activation timing	Immediately after VBLANK start (controller reading)
Effective access speed	30 sectors per second = 3.75K bytes per second
CPU loading	Continuous read time from the 2 cards 2.5%
	Continuous write time to the 2 cards 3.2%

4.3 File System (on ROM)

Device name	buX0 X:connector number (0 or 1)
File name	ASCII characters up to 21 characters
	No directory structure
Management unit	slot 8K bytes (64 sectors) file size unit
Number of slots	15 / card (maximum number of files is 15)

Number of internal events used 0 (amended from 4 as previously stated)
Initialization function _bu_init() (not initialized at time of cut-in)
Simultaneous access limit Each slot can take only one process at a time
(execution failure in the case of multiple functions)

Others

Designate the size by slot units at the time of file create
Size change is not possible afterwards
Volume management is based on slot units
Automatic sector replacement function
Automatic initialization function at the time of opening (optional)

4.4 Service Functions

Library File

libapi.lib

Event usage

Cause descriptor : SwCARD

Event types :	EvSpIOE	process completed
	EvSpERROR	reject card
	EvSpTIMOUT	no card
	EvSpNEW	new card or uninitialized

Function

InitCARD

Format void InitCARD(val)
long val;

Argument val 0 : using without the controller 1 : using with the controller

Explanation

Initialize Memory Card BIOS and put it in a terminated condition. A low level interface starting with _card can thereafter be used directly by activating BIOS with StartCARD().
As the Memory Card file system uses these interfaces internally, these functions must be carried out before _bu_init().
It does not influence the controller.

No return value

StartCARD

Format void StartCARD(void)

No argument

Explanation

It moves the Memory Card BIOS initialized by InitCARD() to a terminated condition.
It does not influence the controller.

No return value

StopCARD

Format void StopCARD(void)

No argument

Explanation

Moves Memory Card BIOS to a stopped condition (same condition as immediately after InitCARD() execution).
It does not influence the controller.

No return value

_bu_init

Format void _bu_init(void)

No argument

Explanation

It initializes the Memory Card file system.
The file system driver is registered in the kernel when this patch module is incorporated, but initialization routing is not carried out automatically. The file system must be initialized clearly with this function.

No return value

_card_info

Format long _card_info(chan)
long chan;

Argument chan port number*16 + card number
port number: A port _ 0 B port _ 1
card number:always 0

Explanation

Carries out the connection test of the memory card designated by “chan”.
The function returns immediately because of the uasychronous operation. Multiple returns cannot be carried out into the same card slot. Completion of the process is notified by EVENT.

Content of Cause Descriptor / Event Classification

SwCARD/EvSpIOE	connected
SwCARD/EvSpTIMOUT	not connected
SwCARD/EvSpNEW	there is no writing after connection
SwCARD/EvSpERROR	communication error occurrence

Returns 1 if process registration of the return value is successful, otherwise return 0.

_card_load

Format long _card_load(chan)
long chan;

Argument chan port number*16 + card number
port number: A port _ 0 B port _ 1
card number:always 0

Explanation

File management information is read into the file system for asynchronous access by the input/output management service. Execution must always take place before opening the file on the Memory Card with the O_NOWAIT mode. There is no need to execute repeatedly as long as the card does not change.
The function itself returns immediately because of the asynchronous processing. Multiple calls cannot be carried out into the same card slot. Finalisation of the process is notified by EVENT The time required is 2 VSYNC.
Always execute once before carrying out open() of O_NOWAIT mode.

Content of Cause Descriptor / Event Classification

SwCARD/EvSpIOE	reading completed
SwCARD/EvSpTIMOUT	not connected
SwCARD/EvSpNEW	uninitialized card
SwCARD/EvSpERROR	communication error occurrence

Returns 1 if process registration of the return value is successful, otherwise return 0.

_card_auto

Sets automatic initialization

Format long_card_auto(val)
long val;

Argument val 0: automatic initialization prohibited
1: automatic initialization permitted

Explanation

It sets up the automatic initialization function

Return value None

_card_clear

Format long _card_clear(chan)
long chan;

Argument chan port number*16 + card number
port number: A port _ 0 B port _ 1
card number:always 0

Explanation

Dummy writing is carried out in the card's system management area, and the condition is cleared (there is no writing after connection).
The function returns immediately because of the asynchronous processing. Multiple returns cannot be carried out into the same card slot. Finalisation of the process is notified by EVENT.

Content of Cause Descriptor / Event Classification

SwCARD/EvSpIOE	process completed
SwCARD/EvSpTIMOUT	not connected
SwCARD/EvSpERROR	communication error occurrence

Return 1 if process registration of the return value is successful, otherwise return 0.

_new_card

Format void _new_card(void)

No argument

Explanation

Masks EvSpNEW events occurring immediately after _card_read() or _card_write().
Returns immediately, even though it is a synchronous function.

No return value

_card_status

format long _card_status(drv)
long drv

argument drv slot number 0 : facing left hand side 1 : facing right hand side

Explanation

It obtains the Memory Card BIOS status in each slot.
It returns immediately, even though it is a synchronous process function.

If the return value Memory Card BIOS is in the operating condition, one of the following values is returned.

Value conditions

0x01	no registration process
0x02	READ process registration
0x04	WRITE process registration
0x08	connection test process registration
0x11	no registration process (timeout occurred immediately before)
0x21	no registration process (communication error occurred immediately before)

_card_wait

Format long _card_status(drv)
long drv

Argument drv slot number 0 : facing left hand side 1 : facing right hand side

Explanation

Waits until there is no registration process for the slot designated by drv.

Always returns 1.

_card_chan

Format long _card_chan(void)

No argument

Explanation

Returns the device number of the Memory Card which caused the event that occurred immediately beforehand. Whether open or test is connected, the return value changes if any event occurs. (normally the event occurs during VBLANK or after BIOS activation by the next VBLANK in the channel in which no card is set.)

The return value is a 2 figure device number being a multiple of 16.

4.5 Asynchronous IO Management Service

Includes open mode macro O_NOWAIT in the second argument

It opens the file with the original management conditions read by _card_load().
This function itself is synchronous, but it completes at high speed.
Without O_NOWAIT (1) always calls out _card_info() internally or (2) the reading operation of the management information can be carried out internally in some cases.

read/write

The process of the read() and write() for files which are opened by designating the O_NOWAIT macro, should be registered in the kernel and returns immediately (registration is completed if the return value is 0). Completion of the process is notified by EVENT.

Event types : EvSpIOE process completed
EvSpERROR irregular finalisation

4.6 File Composition

Item	Size (bytes)
Header	128
Magic	2 (always “SC”)
Type	1
Slot number	1
pad	1
Document name	64 (shift JIS, *1)
pad	27
CLUT	32

Icon image (1) 128(16 x 16 x bits)
Icon image (2) 128 (Type==0x12,0x12,0x13 only)
Icon image (3) 128 (Type==0x13 only)

Data optional (128B x N)

* 1 : non kanji (not Chinese characters) and first standard kanji (Chinese characters) only
* 2 : file size is a multiple of 64 sectors

Type Icon image number
0x11 1
0x12 2
0x13 3

* File name

Byte	Content	Note
0	Magic	Always “B”
1	Area	Domestic is “1” (*1)
2-11	Title	SCE merchandise number (*2)
12-20	User disclosure	Only ASCII used

Finalised with 0x00

*1 : no check by system
*2 : for multiple disc titles this is the first disc’s SCE product code

The SCE product code is decided in the sales decision preparatory meeting (approximately 3 weeks before publication of the master). Until then please use the dummy character string (contents free).

4.7 Management Screen

The memory card management screen is included in the opening demonstration part of the kernel ROM. File deletion and copying between cards (file unit - card as a whole) can be carried out. The icon pattern to be placed at the head of the file is used in this screen. However, this program must be transmitted to RAM in order to run and be activated. Activation during the application is not possible because it also destroys the contents of VRAM.

The management screen displays data and code during the opening in order to support the memory card management screen or an operation to cut in the function during application.

4.8 Backup Operation Rules

Please strictly adhere to the rule of “1 file 1 save point”. 1 save point for multiple files is a very dangerous condition.

When setting multiple save points, it is recommended that this should be managed within one file independently, but multiple file operation is acceptable as long as the rule of “1 save 1 file” is not broken.

Please avoid changing the file size frequently (combining the delete and new formation). Please secure the necessary capacity first in order to avoid causing a capacity deficiency during the game.

If a volume deficiency occurs, please carry out the following procedure.

① Notify the user of the condition.
② Select one of these three: file deletion, card change or continuation without backup.
③ In the case file deletion, display and operate a screen that is similar to the card management screen during the opening (data and source code of the management screen are available.)

*Please ensure that the game can be carried out as far as possible even if no backup is chosen

4.9 Message Screen

In some cases a message needs to be sent to the user (game player), such as demanding card insertion. Although the screen display format or the control system are not set in any particular way, the following procedures are recommended.

Without Card

<u>If card is essential</u>

Display a message, and wait for confirmation by controller operation. Access flow is continued after the user’s confirmation of operation.

<u>If the card is unnecessary</u>

If the card is unnecessary, the cancel option is added to confirmation operation of the necessary circumstances. The demand message should include a warning about the operational limit of the application.

Card Full

Display the space message and wait for confirmation by controller operation. If file deletion is selected, transfer the screen which can carry out file deletion and copying (between card only).

Reject Card

Display the error message and wait for confirmation by controller operation. Continue the process without the card after the user confirms operation.

4.10 Content Guarantee of Written Data

On the application side please take steps to avoid data destruction, caused by reset during data writing, card withdrawal or power off.

Example

Duplicating the written data. Carrying out writing alternately. Also, each sector's final byte adds its own checksum. Carry out checksum test at the time of reading and use another data set at the time of error search.

* A substitute sector function in the file system is valid only for a written error of the memory card.

Sample

[1] Example of synchronized card access

```
#inc|due <sys/file.h>

char data[0x1000];
char buf[0x1000];

main()
{
     long fd;

/* driver initialization*/
InitCARD(0); /* no co-existence with controller */
StartCARD();
_bu_init();

/* deleting file L01 of card on port A */
pritf("delete\n");
delete("bu00:L01");

/* recreate file L01 with 2 slot length in the card on port A */
printf("create\n");
if((fd=open("bu00:L01",0_CREAT|(2<<16)))==-1)
     printf("error\n"):

    close(fd);
     /*Please always close once after creation*/
```

```
/* 10 sector write*/
     printf("write\n");
     if((fd=open("bu00:L01",0_WR0NLY))==-1)
        printf("error 1a\n");
     if(write(fd, &data[0], 10*128) =10*128)
        printf("error 1b\n");
     close(fd);
     /* 30 sector read*/
     printf("read\n");
     if((fd=open("bu00:L01",0_RD0NLY))==-1)
        printf("error 1a\n");
     if(write(fd, &data[0], 30*128) =30*128)
        printf("error 2b\n");
     close(fd);
```

[2] Obtaining the card condition for BIOS & asynchronous access

```
#include <r3000.h>
#include <asm.h>
#include <kernel.h>

unsigned long ev0,ev1,ev2,ev,3;

/* initialization */
_init_my_event()

     ev0 = 0penEvent(SwCARD,EvSpI0E,EvMdN0INTR,NULL);
     ev1 = 0penEvent(SwCARD,EvSpERR0R,EvMdN0INTR,NULL);
     ev2 = 0penEvent(SwCARD,EvSpTIM0UT,EvMdN0INTR,NULL);
     ev3 = 0penEvent(SwCARD,EvSpNEW,EvMdN0INTR,NULL);

     EnableEvent(ev0);
     EnableEvent(ev1);
     EnableEvent(ev2);
     EnableEvent(ev3);

/* test events about memry card */
_test_my_event()
{
if(TestEvent(ev0)==1) /*I0E*/
return 1;
if(TestEvent(ev1)==1) /*ERR0R*/
return 2;
if(TestEvent(ev2)==1) /*TIME0UT*/
return 3;
if(TestEvent(ev3)==1) /*TIME0UT*/
return 4;
```

```
return 0;
}

/*main body*/
main()
{
long i,fd,ret
     char buf[0x1000],data[0x000];
/*controller initialization */
..............

/* initialization of the driver */
     InitCARD(1); /*enable coexistence with controller */
StartCARD();
_bu_init();

/* open events*/
_init_my_event();
     _card_auto(0); /* no automatic initialization */
     ret = _card_info(0x00); /*tests condition of slot 1 card */
     printf("info(0)=%d\n",ret);
     ret = _card_event(); /* decided by this return value*/
     printf("event=%d\n",ret);
     ret = _card_info(0x10);
     printf("info(1)=%d\n",ret);
     ret = _card_event();
     printf("event=%d\n",ret);
ret = _card_load(0x00);
     printf("load(0)=%d\n",ret);
     ret = _card_event();
     printf("event=%d\n",ret);
for(i=0;i<1024;i++){
     buf[i] = 0xff;
     data[i] = i&0xff;
fd = open("bu00:L01",0_CREAT|3<<16); /* file creation (same period
only) */
close(fd);
printf("creat=%d\n",fd);
fd = open("bu00:L01",)_WRONLY|0_NOWAIT);
printf("open=%d\n",fd);
|seek(fd,8192,0);
while((ret = write(fd,data,384))!=0) /* if management registration is
OK 1 is returned */
     ;
printf("write=%d\n",ret);
```

```
ret = _card_event();
close(fd);
printf("event=%d\n",ret);

fd = open("buOO:LO1",)_RDONLY|O_NOWAIT);
printf("open=%d\n",fd);
|seek(fd,8192,O);
ret = read(fd,buf,384);
printf("read=%d\n",ret);
ret = _card_event();
close(fd);
printf("event=%d\n",ret);

for(i=O;i<384;i++){
     if(data[i]!=buf[i]){
        printf("error\n");
        break;
     }
}
CloseEvent(evO);
CloseEvent(ev1);
CloseEvent(ev2);
CloseEvent(ev3);

printf("end of test\n");
```

[3] sector checksum example

```
/*
*test check sum for 128byte block
*return     1 :OK
        0:NG
*/
_test_csum(buf)
unsigned char *buf;
{
     long i;
     unsigned char c;

     c = 0x00;
     for(i=0;i<127;i++)
         c^=*buf++;
     if(*buf==c)
         return 1;
     return 0;

/*set check sum to the last byte of 128byte block*/
_set_csum(buf)
unsigned char *buf;
{
     long i;
     unsigned char c;

     c = 0x00;
     for(i=0;i<127;i++)
         c^=*buf++;
     *buf = c;
}j


/*sample data structure*/
struct SDB {
char name[8];
long size,attr,sector,mode;
}


/*common load buffer*/
unsigned char load_buf[1024];

/*get data from memory card with check-sum test*/
get(num,data)
long num;
struct SDB *data;
{
```

```
long i,fd;

if((fd=open("bu00:L01",0_WR0NLY))<0)
return 0;
     memcpy(&load_buf[0],data,sizeof(struct SDB));
     _set_csum(&load_buf[0]);
     i = write(fd,&load_buf[0],128);
     close(fd);
return (i==128)?1:0;
}


/*get data from memory card with check-sum test*/
get()
{
long i,fd;

if((fd=open("bu00:L01",0_RD0NLY))<0)
return 0;
     if(read(fd,&load_buf[0],1024)!=1024) {
close(fd);
     return 0;
}

for(i=0;i<8;i++)
     if(_test_csum(&load_buf[128*i])==1)
     memcpy(&data[i],&load_buf[128*i],sizeof(struct SDB));
else
     memset(&data[i],0xff,sizeof(struct SDB));
close(fd);
return 1;
}
```

Kanji ROM Specification

6 KANJI ROM SPECIFICATION

6.1 Overview

2 bitmap (16 dots X 16 dots) kanji fonts are recorded in the kernel ROM of PlayStation(tm). This document is about these fonts and their usage.

Font data is not stored in the kernel ROM loaded in development systems produced before DTL-H2000 (PlayStation(tm) board). In the case of the PlayStation board, the patch program (patchx.cpe) is now provided.

Because of this situation the addresses of the font patterns in the Game Machine and the development machine do not always correspond. Therefore rather than disclosing the address of the font pattern we are disclosing the function for getting the address.

6.2 Fonts

Data format 16 dots X 16 dots 2 level bitmap
Letter size 15 dots X 15 dots

Content

JIS first standard kanji and non Kanji, gothic letters non Kanji includes top space (0x2121)

Access method

The heading address of the font pattern of 1 specific letter at the start of the ROM can be obtained by giving the shift JIS code to the service function. After that direct access of the font patterns is possible.

6.3 Data Format

As shown in the drawing below, the byte on the left hand side top of the pattern is first and the byte on the right hand side top follows it. The MSB is on the left hand side.

#	0	#	1
#	2	#	3
…		…	
…		…	
…		…	
#	30	#	31

6.4 Service Functions

The following function is provided as part of libapi.lib. This function is system call, and the code, including an address code exchange table, is located in the kernel area (0-0xffff) of RAM.

Krom2RawAdd	Getting the kanji font pattern address
format	unsigned long Krom2RawAdd(sjiscode) unsigned short sjiscode;
argument	sjiscode shift JIS code

Explanation

It obtains the head address of the font pattern which corresponds to the kanji characters designated by sjiscode.

Return value

returns the start address of the kanji font pattern.
-1 is returned if no font data is provided corresponding to the designated kanji.

6.5 Patch for the DTL-H2000

After reset, run the patchx.cpe. This will allow the service function and font pattern can be used from the program.

The address of the font pattern is different from the one in the Game Machine. (0xbfc66000 in the machine, and 0x1fa66000 in patchX.cpe.) Therefore, please avoid adding the font address directly to the code without the service function.

6.6 Sample Program

The function _get_font()

The function _get_font transmits the font pattern which responds to the designated shift JIS code to VRAM and returns the developed format which can be used as a 16 bit texture.

```
unsigned long
_get( sjis )
unsigned char *sjis;
{
       unsigned short sjiscode;

       sjiscode = *sjis<<8| *(sjis+1);
return Krom2RawAdd(sjiscode); /* getting kanji font pattern address*/
}

#define COLOR 0x4210
#define BLACK 0x3000

_get_font(s, data)
char *s;
unsigned short *data;
{
       unsigned short *p, *d, w;
       long i,j;

       if((p=(unsigned short*)_get(s))!=-1) {
              d = data;
              for(i=0;i<15;i++) {
                     for(j=7;j>=0;jñ)
                            *d++=(((*p>>j)&0x01)==0x01)?COLOR:BLACK;
                     for(j=15;j>=8;jñ)
                            *d++=(((*p>>j)&0x01)==0x01)?COLOR:BLACK;
                     p++;
                           }
              else {
                     for(d=data,i=0;i<2*16*16;i++)
                            *d++=BLACK;
              }
       }
}
```

cpe2x

7 CPE2X

7.1 About CPE2X

CPE2X translates a CPE format file created by the PSYLINK linker to a PS-X EXE format file, which is private execution format. Execution requirements are as follows.

Type of operating platform: PC AT compatible
Operating system: MS-DOS 3.3 or later

Details of CPE format are not disclosed.
Information about PS-X EXE format details and the execution method are contained in the following document.

Related manual: DTL-D2140 Library Reference Version 2.0 Vol.1

The following is an example of its use.

```
C:>CPE2X MAIN.CPE
```

MAIN.EXE is created by this.

PSXCONS

8 PSXCONS

8.1 Introduction

PSXCONS is a user interface front end program used for operating the DTL-H2000 (from now on called the “PS board”), which operates on IBM-PC/AT compatible. The operating environment is as follows.

Operating equipment:	AT compatible
Operating board:	DTL-H2000
Operating OS:	DOS-5/V, DOS-6/V
Required driver:	ANSI.SYS
Incompatible driver:	DEXBIOS.COM (Please delete from the memory.)

Operation and Support

PSXCONS is used on the basis of instructions from our support staff for the purpose of analysing the cause of hardware malfunction. Please refrain from using it for any other purpose.

Expressions Used

[Return] indicates that the Return key should be pressed.
[F1] to [F10] indicate that keys F1 to F10 on the upper part of the keyboard should be pressed.

Related Manual

DTL-D2120 Hardware Guide Version 2.0

8.2 Installation

Confirming the presence of ANSI.SYS

Please confirm that the following line is included within the CONFIG.SYS file, which is located in the root directory of the MS-DOS boot disk.

```
DEVICE=C:\DOS\ANSI.SYS
```

If it is not present, please edit CONFIG.SYS, add the above line, by adding ANSI.SYS, and the reboot.

Installing PSXCONS

Please copy PSXCONS.EXE to a directory defined in the execution PATH.

8.3 Activation

Form

PSXCONS [OPTION] [BATCH FILE NAME]

Option

There is now only one valid option as stated below.

-p port address [, IRQ level [, DMA channel]]

It is designated the I/O address on the PS board.

You can omit the IRQ level and DMA channel designations, but if you do so you must not use the DMA function to interrupt. If the -p option designation is missing, the value set by the DEX2000 environment variable should be adopted. If the DEX2000 environment variable is missing, '0x320,10' should be adopted .

Batch File Name

When the batch file name is designated, the contents of the batch file are read and automatic execution takes place. Details of the batch file are given in Chapter 7 Automatic Execution.

Resetting the PS board

Immediately after activation the PS board is in the mode for booting the program from the CD-ROM drive. In order to use the debug monitor via PSXCONS, it is necessary to reset, taking "2" as the value for the dummy DIPSW. All these operations can be carried out by using the keyboard via PSXCONS. Please refer to the following examples of operation.

Examples of Operation

Example 1

```
C:\> psxcons  [RETURN]
psxcons DEX2000 console program
     $Date: 1994/07/19 18:11:50 $
     type F1 -> display help
     when hung up try type Ctrl+BREAK
I/O addr = 0x320, IRQ=10(vect=0x0072,8259=A0)
[F7]2[RETURN]
[F9][F10]
PSX>
```

Example 2

```
C:\> set psxcons -p0x300 [RETURN]
psxcons DEX2000 console program
     $Date: 1994/06/02 05:03:21 $
     type F1 -> display help
     when hung up try type Ctrl+BREAK
I/O addr = 0x300
```

(Then, same as Example 1)

Example 3

```
C:\> set DEX2000=0x340 [RETURN]
C:\> PSXCONS [RETURN]
 PSXCONS DEX2000 console program
 $Date: 1994/06/02 05:03:21 $
 type F1 -> display help
 when hung up try type Ctrl+BREAK
I/O addr = 0x340
```

(Then, same as Example 1)

8.4 Basic Action

PSXCONS transfers the usual key input to PSX and displays on screen the characters transferred from PSX. However, the function key and cursor key input are not transferred to PSX, but are handled locally.

In particular, keys [F1] to [F8] activate the following local commands.

[F1] displays the help message.
[F2] inputs the DOS command and executes it.
[F3] inputs the batch file name and executes the batch file.
[F4] inputs the file name of the binary file and downloads it.
[F5] inputs the name of the operation recording file and starts recording the operation.
[F6] displays the dummy LED.
[F7] inputs and displays the dummy DIPSW data.

Additionally, [F9] activates local commands by inputting the function keys as follows.

[F9][F4] inputs the file name of the S-format file and downloads it.
[F9][F5] terminates recording the operation.
[F9][F10] resets psx.

Additionally, [F10]activates a local command by inputting the function keys as follows.

[F10][F2] terminates the PSXCONS program.

8.5 Key Operation at the Time of Local Command Execution

If you execute a local command by using the function keys as described in the preceding chapter, you may need to input data such as a file name etc.

At the time of input, editing operation can be performed by using the following keys.

Left Arrow moves the cursor to the left.
Ctrl-s moves the cursor to the left.
Right Arrow moves the cursor to the right.
Ctrl-d moves the cursor to the right.

Up Arrow	moves up the history buffer.
Ctrl-e	moves up the history buffer.
Down Arrow	moves down the history buffer.
Ctrl-x	moves down the history buffer.
Ctrl-g	deletes the character where the cursor is located.
Ctrl-y	deletes one line or quits the history buffer.
Ctrl-k	deletes from the cursor position to the end of the line.
TAB	The present input line moves up the history buffer until it meets the top of the buffer.
Ctrl-j	The present input line moves down the history buffer until it meets the top of the buffer.
ESC	stores in the history buffer without executing the current input line.

8.6 Downloading the Program

Binary data is downloaded by activating the local Bload command with the [F4] key.
Multiple pairs of file names and addresses (to a maximum of 16) can be given to the Bload command argument.

```
PSX> [F4]
Bload[x]:file1 address1 file2 address2 [RETURN]
smon
CCS Target mode

     file1   address:xxxxxxxx size:xxxxxx xxxxxx: xxsec.
     file2   address:xxxxxxxx size:xxxxxx xxxxxx: xxsec.

Exit from CCS Target mode
PSX>
```

S-format data can also be downloaded by activating the local Down2 command with the [F9][F4] keys.

```
PSX> [F9][F4]
Down2[x]: filename] [RETURN]
ld2

S-format data load start
xxx line                ← the transmission line number is sometimes displayed during transmission

"filename" send end

GP:a00xxxxxx PC:a00xxxxxx
PSX>
```

8.7 How to Terminate PSXCONS

Activate the local Quit command with the [F10][F2] key.

```
PSX> [F10][F2]

c:\>
```

8.8 Auto Execution

Outline of Auto Execution

There is a simple auto execution function in PSXCONS.EXE. If you create a text file which lists the commands that are input from the keyboard and designate the file name with the AUTO command activated by the [F3] key, it reads and executes from the file line by line each time PSX displays the prompt.

```
PSX> [F3]
Auto[x]: batchfile [RETURN]
BATCHFILE LINE 1
PSX>BATCHFILE LINE 2
PSX>BATCHFILE LINE 3
PSX>
```

Caution

As this command unconditionally executes the first line of the batch file, please make sure you activate whilst the PSX prompt is displayed.

Batch File Commands

Within the batch file it is possible to describe the commands you input to PSX. These commands are sent line by line to PSX each time PSX displays the command prompt.

Local Commands in the Batch File

To activate the PSXCONS.EXE local command in the batch file, first input the word 'local', and then the command name followed by the argument, as follows:

local local-command argument1 argument2

The following local commands may be used.

Help	displays the help message.
DOS	executes the DOS command given to the argument.
Auto	executes the batch file of the file name given to the argument. The batch file nest is permitted to a level of 16.
Bload	With the bload argument you give the file name and address, and download the binary file.
Down2	You download the S-format file, which has the file name given to the Down2 argument.
Log	If an operation recording file name is given to the log argument, operation recording is commenced. If there is no argument, operation recording is terminated.
Led	displays the dummy LED.
Dipsw	creates in the dummy DIPSW up to 16 data given to the Dipsw argument.
Reset	resets the psx.
Beep	makes the sound of a buzzer.
Pause	waits for key input. Press the appropriate key to continue.
Echo	displays the row of letters given by the Echo argument.
Sleep	stops for only the number of seconds given by the sleep argument.
Auto-again	repeats from the top the batch file that is in the process of execution. This command is a temporary specification. It may disappear.
Quit	terminates the PSXCONS program.

Batch File Example

```
local reset
local sleep 1

local bload program 80080000
local beep

sr sp 801ffff0
local echo type any key to execute program.
local pause
call 80080100
```