# PDA Kernel Specification

Publication date: November 1998

Sony Computer Entertainment America
919 E. Hillsdale Blvd., 2nd floor
Foster City, CA 94404

Sony Computer Entertainment Europe
Waverley House
7-12 Noel Street
London W1V 4HH, England

# Table of Contents

# About This Manual

This manual is the 1.5 release of the PDA Kernel Specification documentation.

## Changes Since Last Release

This is the first English release of this documentation.

## Related Documentation

This manual should be read in conjunction with the following documents:

PDA Hardware Specification

## Developer Reference Series

This manual is part of the *Developer Reference Series*, a series of technical reference volumes covering all aspects of PlayStation development. The complete series is listed below:

| Manual | Description |
|---|---|
| PlayStation Hardware | Describes the PlayStation hardware architecture and overviews its subsystems. |
| PlayStation Operating System | Describes the PlayStation operating system and related programming fundamentals. |
| Run-Time Library Overview | Describes the structure and purpose of the run-time libraries provided for PlayStation software development. |
| Run-Time Library Reference | Defines all available PlayStation run-time library functions, macros and structures. |
| Inline Programming Reference | Describes in-line programming using DMPSX, GTE inline macro and GTE register information. |
| SDevTC Development Environment | Describes the SDevTC (formerly "Psy-Q") Development Environment for PlayStation software development. |
| 3D Graphics Tools | Describes how to use the PlayStation 3D Graphics Tools, including the animation and material editors. |
| Sprite Editor | Describes the Sprite Editor tool for creating sprite data and background picture components. |
| Sound Artist Tool | Provides installation and operation instructions for the DTL-H800 Sound Artist Board and explains how to use the Sound Artist Tool software. |
| File Formats | Describes all native PlayStation data formats. |
| Data Conversion Utilities | Describes all available PlayStation data conversion utilities, including both stand-alone and plug-in programs. |
| CD Emulator | Provides installation and operation instructions for the CD Emulator subsystem and related software. |
| CD-ROM Generator | Describes how to use the CD-ROM Generator software to write CD-R discs. |

| | |
|---|---|
| Performance Analyzer User Guide | Provides general instructions for using the Performance Analyzer software. |
| Performance Analyzer Technical Reference | Describes how to measure software performance and interpret the results using the Performance Analyzer. |
| DTL-H2000 Installation and Operation | Provides installation and operation instructions for the DTL-H2000 Development System. |
| DTL-H2500/2700 Installation and Operation | Provides installation and operation instructions for the DTL-H2500/H2700 Development Systems. |

## Typographic Conventions

Certain Typographic Conventions are used through out this manual to clarify the meaning of the text. The following conventions apply to all narrative text except for structure and function descriptions:

| *Convention* | *Meaning* |
|---|---|
| `courier` | Indicates literal program code. |
| **Bold** | Indicates a document, chapter or section title. |

The following conventions apply within structure and function descriptions only:

| *Convention* | *Meaning* |
|---|---|
| **Medium Bold** | Denotes structure or function types and names. |
| *Italic* | Denotes function arguments and structure members. |

## Developer Support

### Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

| Order Information | Developer Support |
|---|---|
| **In North America** | **In North America** |
| Attn: Developer Tools Coordinator<br>Sony Computer Entertainment America<br>919 East Hillsdale Blvd., 2nd floor<br>Foster City, CA 94404<br>Tel: (650) 655-8000 | E-mail: DevTech_Support@playstation.sony.com<br>Web: http://www.scea.sony.com/dev<br>Developer Support Hotline: (650) 655-8181<br>(Call Monday through Friday, 8 a.m. to 5 p.m., PST/PDT) |

### Sony Computer Entertainment Europe (SCEE)

SCEE developer support is available to licensees in Europe only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

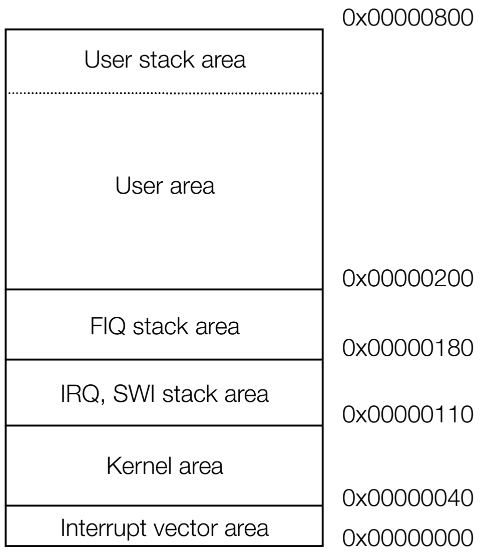
| Order Information | Developer Support |
|---|---|
| **In Europe** | **In Europe** |
| Attn: Production Coordinator<br>Sony Computer Entertainment Europe<br>Waverley House<br>7-12 Noel Street<br>London W1V 4HH<br>Tel: +44 (0) 171 447 1600 | E-mail: dev_support@playstation.co.uk<br>Web: https://www-s.playstation.co.uk<br>Developer Support Hotline:<br>+44 (0) 171 447 1680<br>(Call Monday through Friday, 9 a.m. to 6 p.m., GMT or BST/BDT) |

# Kernel Services Overview

## Memory map

The PDA kernel initializes static RAM as shown in the figure below. 512 bytes are reserved as a system area for interrupt vectors, the kernel, the system stack, etc. The remaining 1.5 Kbytes can be used as a user area. The user area includes the user stack.

**Figure 1: Initial Memory Map**



## System calls

Applications use system calls for predefined kernel services. The parameters of a system call are passed in registers r0, r1, r2. If the system call returns a result, it is placed in register r0.

The main functions provided by system calls are:

- Setting up and handling of interrupts
- Controlling communication with the PlayStation
- Controlling PDA applications
- Setting the system clock
- Accessing flash memory
- Accessing the real-time clock
- Accessing the serial number

Details of the various system calls are presented below.

## Interrupt handling

The PDA kernel handles interrupts by invoking callback functions previously set up by a PDA application. The callbacks handle responses to IRQ, FIQ, and software interrupts.

If a callback function uses Thumb mode, the LSB of the callback function address must be set to 1. When returning from the callback function, the previous mode that was active when the callback function was called, must be restored using an instruction such as "bx lr".

When a callback function uses ARM mode, stack push and stack pop operations must be performed using pre-decrement store (stmfd) and post-increment load (ldmfd).

Since the kernel does not clear interrupt flags, the flags must be cleared by the callback function.

### Handling of IRQ interrupts

If the IRQ interrupt source is either the PS interface power supply line (BATIRQ) or the real-time clock interrupt (RTCIRQ), the interrupt flags must not be cleared in the PDA application's callback function since these flags are cleared by the kernel. The interrupt status for these interrupt sources can be accessed from the application.

When an IRQ callback function is called, the kernel pushes the r0, r1, r12 and r14(lr) registers onto the IRQ stack. If other registers need to be saved, this must be done by the callback function.

### FIQ interrupts

An FIQ interrupt from the Synchronous Serial Communications module (SPIFIQ) cannot be used by a PDA application since it is handled by the kernel.

When an FIQ callback function is called, the kernel pushes the r0, r1, r12 and r14(lr) registers onto the FIQ stack. If other registers need to be saved, this must be done by the callback function.

If the PDA is inserted in the PlayStation, FIQ interrupts must be handled with care. If an FIQ interrupt from counter 2 (TC2FIQ) takes place during communication with the PlayStation, a communication error may result. Communication with the PlayStation must be inhibited (swi 17) to provide reliable counter 2 FIQ interrupt handling

If the PDA is not inserted in the PlayStation, FIQ interrupts can be handled faster by rewriting the FIQ interrupt vector. If this is done, however, the interrupt vector must be restored to its original state later

### Software interrupt handling

Software interrupts are used to perform operations that cannot be executed in user mode.

When a software interrupt callback function is called, the kernel pushes the r1-r12 and r14(lr) registers onto the software interrupt stack. Since the values for the r0-r10 registers are saved when the callback function is called, these can be used as parameters to the callback function. Also, the r0 register can be used to hold the return value from the callback function.

### Using kernel services in callback functions

System calls cannot be invoked directly during interrupt processing. Therefore a service is provided so that the system call jump table can be accessed. This table contains jump destinations of the various system call processing modules arranged in sequence according to system call number.

To use a kernel service during interrupt processing, its jump destinations must be retrieved from this table in advance and passed as a parameter to the interrupt processing function.

## Initial settings

When the PDA is reset, PIO01 is set up as an output port, the system clock is set to 4 MHz, the low-voltage detector is set active, and the LCD is turned on and set to a frame rate of 64 Hz. All other device registers are left in their reset states.

When an application is started, communication with the PlayStation is disabled and must be restarted using the PlayStation communication function control system call (swi 17). If the PDA is not inserted in the PlayStation, communication will not be started.

When the PDA is inserted or removed from the PlayStation, an interrupt is generated from the PS interface by the power supply voltage detector (BATIRQ). The interrupt handling for this event must include control of PlayStation communication status.

## Controlling PDA applications

The kernel provides services for starting up, exiting and pausing applications. A PDA application is specified using its Memory Card block number. A single Memory Card can have 15 blocks, with block numbers 1 - 15 available. Block number 0 is reserved for the "Start-up Application".

### Starting PDA applications

PDA applications are stored in blocks of 8 Kbytes, just like Memory Card data. If a PDA application spans two or more blocks, the application may or may not be stored in consecutive blocks. When a PDA application starts, the kernel maps discrete blocks into a continuous virtual address space using the MMU (memory management unit). The start of the virtual address space is always 0x2000000.

When mapped to the virtual address space, the PDA application is started from the address specified by the "program start address" field in the file header. It is assumed that the header of the "program start address" is at 0x2000000.

The start-up sequence for PDA applications is as follows: the application to be started next is established (swi 8); the currently active application is exited (swi 9); and control transitions to the next application. The power to the PDA cannot be cut off as long as the battery is running, so one of the PDA applications, including the "start-up application" is always running.

Once a PDA application exits (swi 9), there are two ways in which the application can be restarted. Regardless of which method is used, the system clock will be set to 4 MHz and all interrupts except for BATIRQ and RTCIRQ will be disabled immediately after start-up.

Also, registered callback functions will be cleared. The following is a description of the two methods for restarting an application.

### Cold Start

The PDA application is started from the beginning.

The user area is initialized at this time.

### Hot Start

The PDA application is started from where it left off.

However, if the "file type" field in the file header specifies "MCX0", a Cold Start will be forced since the RAM save area will not be preserved.

### Exiting PDA applications

The system call for exiting a PDA application (swi 9) terminates the currently active PDA application and starts up the next application that was previously registered. The parameters to the application exit system call determine the manner in which the application is restarted.

Communication with the PlayStation must be inhibited (swi 17) immediately before exiting the PDA application.

### Arguments to the "Start-up Application"

The following values can be used for the arguments to the "Start-up Application" (set in r2 of SWI 8). Improper operation may result if values other than these are specified.

| Bit | Function |
|---|---|
| 0-3 | Specifies the block number (0x0 - 0xF). A zero means that no block is specified. |
| 4-7 | Specifies the screen to be displayed at start-up.<br>0x1: Clock B screen. Block number specification is ignored.<br>0x2: Clock A screen. If the specified block number is 0, normal display. If the block number is non-zero, clock-setting mode is enabled immediately after start-up, and pressing the "Enter" button will start up the PDA application at the specified block. (*)<br>0x3: Game selection screen. The icon for the specified block number is displayed. |

(*) This feature can be used to call the clock-setting feature of the "Start-up Application" from a user application. However, the user application will be restarted with no arguments so it is necessary to either use resume (see the description of swi 9) or to perform an operation such as writing to a flag in flash memory.

## Communication with the PlayStation

Two types of callbacks are provided to allow a PDA application to be controlled from the PlayStation: the file transfer control callback and the device entry callback.

### The file transfer control callback

The PlayStation library (libmcx) can notify a PDA application when data transfer starts and completes. The file transfer control callback is invoked when this notification takes place. This callback function is performed by setting the user callback (swi 1).

Parameters to the callback function are passed through the r0 register.

**Table 1: Parameters to the file transfer control callback function**

| Bit 23-16 | Bit 15-8 | Bit 7-0 |
|---|---|---|
| Transfer direction<br>0 : PDA -> PlayStation<br>1 : PDA <- PlayStation | Transfer control<br>0 : Start<br>non-0: Stop | Timeout interval<br>Timeout (no. of secs.) |

When an error is generated during a file transfer, the PDA application can perform error handling during a specified timeout interval. During this time, the PDA application must inform the kernel that transfer has stopped by invoking a system call (swi 11) indicating the end of file transfer with the PlayStation.

### The device entry callback

Data transfer between the PlayStation and the PDA can be performed using the device entry callback contained in the file header. The device entry callback is called from the PlayStation library (libmcx).

The following is an example of the device entry callback function.

```
typedef struct {
   char            *buf;
   unsigned long   size;
} buf_rec;

buf_rec *dev_ent_callback(unsigned long status, char *rec_buf)
{
   :
}
```

The modes from the following table are assigned to each of the bits of status, above.

**Table 2: Bit assignments of status**

| Bit | Meaning |
|---|---|
| 0 | Read |
| 1 | Write |
| 2 | Data transfer error |
| 3 | Parameter passing |
| 4 | Data transfer finished |

When a parameter passed to the libmcx functions McxReadDev() or McxWriteDev() is received, the device entry callback is called in parameter-passing mode. Once the data has been transferred, the device entry callback is called again in data-transfer-finished mode.

In parameter-passing mode, the second parameter (rec_buf) contains the buffer address in which the received parameter is stored. The PDA application is then able to interpret the parameter.

In read mode, the size and address of the data to be sent is stored in a structure (buf_rec) and sent back as the return value. In write mode, the size and address of the data to be received is stored in the buf_rec structure and sent back as the return value.

In data-transfer-finished mode, the second parameter (rec_buf) contains the starting address of the transferred data. At this point, the PDA application can perform post-transfer operations. No return value is defined for data-transfer-finished mode. If an error was generated during data transfer, the data transfer error bit in status is set.

## Notes on PDA device control

The IR communications module, speaker, and LED that are built into the PDA have high current consumption when active. Therefore, when the PDA is inserted in the PlayStation, the devices on the PDA must be activated in ways that do not exceed the PlayStation's current consumption capacity.

PDA status (described later) can be used to control devices while taking current consumption into consideration. Device status is represented as either enabled (0) or disabled (1). Device status is initialized as enabled, but is set to disabled when the PDA is inserted into the PlayStation. The status returns to enabled when the PDA is removed from the PlayStation. When the PDA is inserted in the PlayStation, the PlayStation application must grant permission to the PDA so that it can use the devices. The PlayStation application must take into account the maximum current that can be supplied to the devices before granting permission. Since the PlayStation controls PDA status in this manner, the PDA application must always check the PDA status before attempting to use a device. In other words, the PDA can use a device if its status is enabled but it must not if its status is disabled.

## System Call Details

Table 3: System Call Table

| System Call contents | Number | Argument | Return Value |
|---|---|---|---|
| Software reset | 0 | - | - |
| Set user callback | 1 | r0: Interrupt type<br>r1: Address of callback function | Previous address of callback function |
| Invoke user callback | 2 | - | Return value of callback function |
| Write to flash memory (relative sectors) | 3 | r0: Destination address<br>r1: Source address | Result of write |
| Set system clock frequency | 4 | r0: System clock frequency | Previous value of system clock frequency |
| Switch kernel mode | 5 | - | - |
| Get PDA status | 6 | - | Address of status buffer |
| Application block number control | 8 | r0: Get/set specification<br>r1: Block number<br>r2: Application argument | Get/set block number |
| Terminate/suspend application | 9 | r0: Restart method | - |
| Read serial number | 10 | - | Serial number |
| Terminate file transfer with PlayStation | 11 | - | - |
| Write real-time clock | 12 | r0: Year, month, day<br>r1: Day of week, hour, minute, second | - |
| Read calendar | 13 | - | Calendar value |
| Read clock | 14 | - | Clock value |
| Write serial number | 15 | r0: Serial number | - |
| Write flash memory (absolute number) | 16 | r0: Write destination address<br>r1: Write source address | Result of write |
| Control PlayStation communication | 17 | r0: State of communication with PlayStation | - |
| Get application status | 18 | r0 | Application state |
| Get user interface status | 19 | - | Status buffer address |
| Get system call table | 20 | - | Address where table address is stored |
| Get application block number | 22 | - | Application block number of active application |
| Get application flag | 24 | r0: Program number (1-15) | PDA application flag |

## Software reset

**Syntax**

swi  0

**Arguments**

None

**Return value**

None

## Set user callback

**Syntax**

swi 1

**Arguments**

r0: Interrupt type

- 0: Software interrupt
- 1: IRQ interrupt
- 2: FIQ interrupt
- 3: Start/stop display of file transfer in progress from the PS

r1: Address of the callback function

**Return value**

Previous address of callback function

**Remarks**

LSB is 1 if the callback function is written in ARM Thumb code.
When returning from the callback, return to the destination mode using 'BX LR', etc.

When an interrupt is generated, registers are saved as needed before the callback function is called.

**Table 4: Registers saved for each interrupt type**

| Type of interrupt | Saved register |
|---|---|
| Software interrupt | R1-R12, LR |
| IRQ interrupt | R0, R1, R12, LR |
| FIQ interrupt | R0, R1, R12, LR |
| Start/stop display of file transfer in progress from the PS | None |

## Invoke user callback

**Syntax**

swi  2

**Arguments**

None

Registers r0-r10 are passed to the user callback functions while holding the value when the system call was invoked

**Return value**

Return value of user callback function

## Write to flash memory (relative sectors)

### Overview

Each Memory Card block is 8 KBytes long and consists of 64 sectors. This system call performs writes, in one-sector (128 byte) units, from the starting sector in the Memory Card file using relative sector numbers.

### Syntax

swi 3

### Arguments

r0: Destination address

Destination address based on a relative sector number from the first sector of the file in Memory Card format (see notes below).

r1: Source data buffer address

### Return value

0: Write succeeded

1: Write failed

### Remarks

Before making this system call, the system clock frequency must be set to 4 MHz (no wait cycles).
The source data buffer must be outside the flash memory area.
The LSB of the source data buffer address must be 0.

When executing a PDA application, the MMU relocates the blocks assigned to the application to virtual addresses beginning at address 0x02000000. A "relative sector number" means a sector number relative to 0x02000000, where 0x02000000 is defined to be sector number 0.

## Set system clock frequency

### Syntax

swi 4

### Arguments

r0: System clock frequency specification

| | |
|---|---|
| 1: | 62kHz |
| 2: | 125kHz |
| 3: | 250kHz |
| 4: | 500kHz |
| 5: | 1MHz |
| 6: | 2MHz |
| 7: | 4MHz |
| 8: | 8MHz |

### Return value

The previous value of the system clock frequency

### Remarks

If the system clock frequency is set to 8 MHz, wait cycle insertion mode is enabled automatically. For all other clock frequencies, the wait cycle insertion mode is disabled.

Note:

Although in 1.4 and earlier versions of this document, it was possible to use the “0” argument to set the system clock frequency to 32KHz, problems with this system call functioning were detected with the “0” argument. Therefore, assigning “0” to an argument is now prohibited . To set the system clock frequency to 32KHz without using this system call, set the device register directly by making <PMFrequency><FREQ> “0”.

## Switch kernel mode

**Overview**

When the PDA is inserted in the PlayStation, it is put in a state where it can communicate with the PlayStation as a Memory Card. At this time the system clock frequency will be set to 4 MHz. Subsequently, if this system call is used to change the system clock frequency to 2 MHz or lower, it will not be possible to use the PDA as a Memory Card.

**Syntax**

```
swi  5
```

**Arguments**

None

**Return value**

None

## Get PDA status

**Syntax**

swi 6

**Arguments**

None

**Return value**

Address of status buffer

**Remarks**

Bit assignments for the status value (32 bits) are shown below.

**Table 5: PDA status**

| Bit | Contents | 0 | 1 |
|---|---|---|---|
| 0 | Write to flash memory | Enabled | Disabled |
| 1 | Speaker output | Enabled | Disabled |
| 2 | LED light | Enabled | Disabled |
| 3 | Transmit infrared | Enabled | Disabled |
| 8 | Insertion/removal from PlayStation | Inactive | Active |
| 9 | Communication with PlayStation | Not possible | Possible |
| 10 | File transfer status with PlayStation | Exited | In transfer |
| 11 | PDA application exited | No | Yes |

Bits 0 - 3 are initialized to the enabled state and are set to the disabled state when the PDA is inserted into the PlayStation. Subsequently, the PlayStation application may enable the devices as needed. The devices corresponding to bits 1 - 3 should be enabled by the PlayStation application provided current consumption is within the acceptable range. Thus, when PDA applications use these devices, they must check the PDA status and only use the devices that have been enabled.

The insertion/removal flag is cleared when the kernel mode switching system call (swi 5) is made.

The flag for communication with the PlayStation is controlled by the PlayStation communications control system call (swi 16).

The PDA application exited flag is set when a termination request is sent from the PlayStation application.

## Application block number control

**Overview**

Gets/sets the starting block number (the block number that contains the file header) of the PDA application resident in the Memory Card.

**Syntax**

swi 8

**Arguments**

r0: Block number control

0: Get block number

1: Set block number

r1: Block number (1-15)

r2: Application argument

* r1, r2 are set only when number setting

**Return value**

The get or set block number

**Remarks**

The application specified by this system call is started by the application terminate/suspend system call (swi 9).

Block number 0 indicates the default utility (clock feature, application selection feature).

## Terminate/suspend application

**Syntax**

swi 9

**Arguments**

r0: Method for restarting the active application

- 0: Terminate the active application and "coldstart" when the application is restarted.
- 1: Terminate the active application and "resume" when the application is restarted.

r1: Parameter passed to application to be started. (Used only when cold-starting the program).

**Return value**

None

## Read serial number

**Syntax**

swi 10

**Arguments**

None

**Return value**

Serial number

## Terminate file transfer with PlayStation

**Syntax**

swi 11

**Arguments**

None

**Return value**

None

**Remarks**

If the PlayStation is reset while a file is being transferred to the PDA, the termination of the transfer cannot be detected. The kernel must be notified of the termination of this transfer by invoking this system call.

When the PDA is removed from the PlayStation, the PDA is automatically restored to its initial state, so it is not necessary to make this system call.

## Write real-time clock

### Syntax

swi 12

### Arguments

r0: Year (high-order value), year (low-order value), month, day

r1: Day of the week, hours, minutes, seconds

### Return value

None

### Remarks

Parameters are expressed as two-digit BCD values. For the day of the week, the values 1 - 7 correspond to Sunday - Saturday.

For more information, please refer to the PDA specification.

## Read calendar

**Syntax**

swi 13

**Arguments**

None

**Return value**

Calendar value. BCD in the form: YYYYMMDD.

## Read clock

**Syntax**

swi 14

**Arguments**

None

**Return value**

Clock value. BCD in the form: DDHHMMSS.

For day (DD), values 1 - 7 correspond to Sunday - Saturday.

## Write serial number

**Syntax**

swi 15

**Arguments**

r0: Serial number (32 bits)

**Return value**

None

## Write flash memory (absolute number)

**Overview**

Each Memory Card block is 8 KBytes long and consists of 64 sectors. This system call uses absolute numbers to perform writes in one-sector (128 byte) units, where the sector starting at 0x08000000 corresponds to sector number 0.

**Syntax**

swi 16

**Arguments**

r0: Destination address

Destination address, expressed as an absolute sector number where the sector starting at 0x08000000 corresponds to sector number 0.

r1: Address of source data buffer

**Return value**

0: Write succeeded

1: Write failed

**Remarks**

Before making this system call, the system clock frequency must be set to 4 MHz (no wait cycles).

The source data buffer must be outside the flash memory area.
The LSB of the source data buffer address must be 0.

## Control PlayStation communication

**Syntax**

swi 17

**Arguments**

r0: State of communication with the PlayStation

- 0: Stopped
- 1: Started

**Return value**

None

**Remarks**

When communication between the PDA and PlayStation is interrupted, the PDA will no longer be recognized as a Memory Card. If the PDA is not mounted to the PlayStation, enabling the start of communication will not start communication.

## Get application status

### Overview

The starting block number (the block number that contains the file header) of the PDA application resident in the Memory Card is specified. The restart status of the application is retrieved.

### Syntax

swi 18

### Arguments

r0: Memory card block number (1-15)

### Return value

Application status

0: Coldstart

1: Resume

### Remarks

The presence of an application at the specified block number is not checked.

## Get user interface status

**Syntax**

swi 19

**Arguments**

None

**Return value**

Address of the status buffer of the user interface

**Remarks**

The bit assignments of the status value (64 bits) are as shown below.

**Table 6: User interface status**

| Bit | Contents | Notes |
|---|---|---|
| 47-32 | Font data starting address | Address relative to 0x4000000 |
| 23 | Clock setting flag | 0: Not set, 1: Set |
| 22-20 | Area code | 0: Japan, 1: North America, 2: Europe |
| 19-18 | Speaker volume | 0: high, 1: low, 2: off |
| 17 | Keylock ON/OFF | 0: OFF, 1: ON |
| 16 | Alarm ON/OFF | 0: OFF, 1: ON |
| 15-8 | Alarm: hour | 2 digits, BCD |
| 7-0 | Alarm: minutes | 2 digits, BCD |

Output is 1/4 if the speaker volume shown on the clock screen of the PDA is set to Low. If the setting is High, the output value is left unchanged.

Refer to the appendix for font data details.

## Get system call table

**Syntax**

swi 20

**Arguments**

None

**Return value**

Address at which the system call table address is stored.

**Remarks**

The system call table holds addresses in numeric order.

## Get application block number

**Syntax**

swi 22

**Arguments**

None

**Return value**

Application block number of active application

## Get PDA application flag

### Overview

The internal FAT flag that is set when a PDA application stored in the Memory Card is downloaded directly from the PlayStation is obtained. This flag is cleared when a PDA application has been copied using the PlayStation Memory Card Set-up screen.

### Format

```
swi          24
```

### Arguments

r0: Memory Card block number (1-15)

### Return value

PDA application flag

0: File is not a PDA application, or is a PDA application that was not downloaded directly from the PlayStation

1: File is a PDA application that was downloaded directly from the PlayStation

### Remarks

Returns a 1-byte value from offset 126 of the application’s FAT.

# Appendix

## Font data used by the "Start-up Application"

The font data used by "Start-up Application" is arranged according to the Font Data Map shown below. This font data is available for use by PDA applications. The starting address for the font data can be obtained through a system call (swi 19).

Refer to the Font Data Image for displayed font data images.

For font data in the fontdata48 category, the high-order byte (the bottom two lines of the Font Data Image) is reserved for other data and must be masked when used as font data.

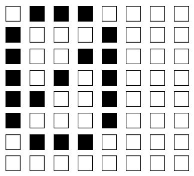
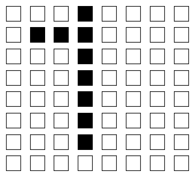
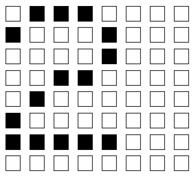
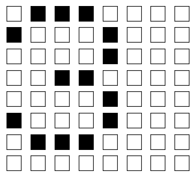
### Font Data Map

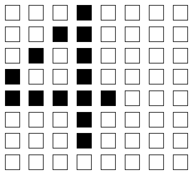
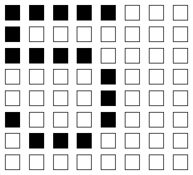
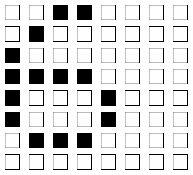
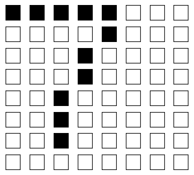
```
fontdata88
    DCD 0x1519110E, 0x000E1113      ;'0'
    DCD 0x08080E08, 0x00080808      ;'1'
    DCD 0x0C10110E, 0x001F0102      ;'2'
    DCD 0x0C10110E, 0x000E1110      ;'3'
    DCD 0x090A0C08, 0x0008081F      ;'4'
    DCD 0x100F011F, 0x000E1110      ;'5'
    DCD 0x0F01020C, 0x000E1111      ;'6'
    DCD 0x0808101F, 0x00040404      ;'7'
    DCD 0x0E11110E, 0x000E1111      ;'8'
    DCD 0x1E11110E, 0x00060810      ;'9'
    DCD 0x00000000, 0x00000000      ;' '

    DCD 0x00000300, 0x00000300      ;null
    DCD 0x7E242418, 0x007E425A      ;lock
    DCD 0x95175418, 0x00185417      ;speaker
    DCD 0x02000001, 0x00010000      ;loud
    DCD 0x09172458, 0x00191215      ;spkoff
    DCD 0x93a3de80, 0x000204fa      ;batlow
    DCD 0x2a122c48, 0x00097b26      ;alloff
    DCD 0x222a1c08, 0x00087f22      ;allarm
    DCD 0x2121213f, 0x001e212d      ;mcard

    DCD 0x1f111111, 0x00111111      ;H
    DCD 0x110e0000, 0x001e011f      ;e
    DCD 0x04040406, 0x000e0404      ;l
    DCD 0x110e0000, 0x000e1111      ;o
```

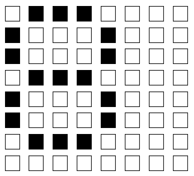
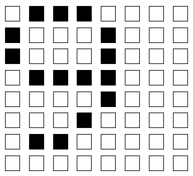
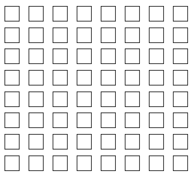
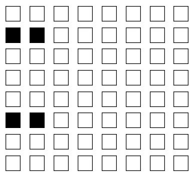
```
        DCD 0xfdf9621c, 0xfcfafdfd          ;hart01
        DCD 0x7e7c231c, 0x1f3f7f7f          ;hart02
        DCD 0x081c3e7f, 0x00000000          ;hart03
        DCD 0xfd6db10e, 0xf0f8fefd          ;hart11
        DCD 0x1f17100f, 0x01030f1f          ;hart12

        DCD 0x1f1f0a00, 0x0000040e          ;s_hart2
        DCD 0x05000000, 0x00000207          ;s_harSL
        DCD 0x14000000, 0x0000081c          ;s_harSS

        DCD 0x0e12120f, 0x000f1212          ;B
        DCD 0x11110000, 0x00161911          ;u
        DCD 0x011e0000, 0x000f100e          ;s
        DCD 0x11110000, 0x000e101e          ;y

    fontdata48
        DCD 0x00075557      ;0
        DCD 0x04022223      ;1
        DCD 0x05071747      ;2
        DCD 0x06074747      ;3
        DCD 0x07047564      ;4
        DCD 0x08074717      ;5
        DCD 0x08075717      ;6
        DCD 0x00022247      ;7
        DCD 0x04075757      ;8
        DCD 0x05074757      ;9
        DCD 0x06000000      ;' '
        DCD 0x07002020      ;:
        DCD 0x08012244      ;/
        DCD 0x08011122      ;/
        DCD 0x08024655      ;y
        DCD 0x01000000      ;' '
        DCD 0x020f5642      ;a
        DCD 0x01075744      ;d
        DCD 0x03061752      ;e

        DCD 0x2D022e2e      ;F
        DCD 0x62055711      ;h
        DCD 0x65088ad8      ;M
        DCD 0x64055700      ;n
        DCD 0x3D075700      ;o
```

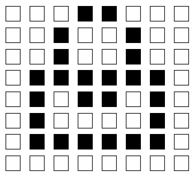
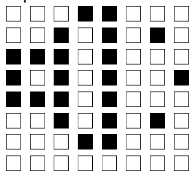
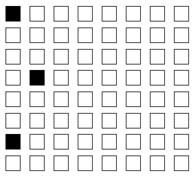
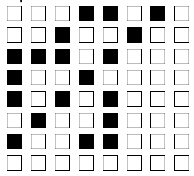
```
DCD 0x70011350        ;r
DCD 0x65068e2c        ;S
DCD 0x6E04444e        ;T
DCD 0x2B062720        ;t
DCD 0x6D075500        ;u
DCD 0x6105aaaa        ;W
DCD 0x64022202        ;i
DCD 0x2E001111        ;!

DCD 0x16005752        ;A(AM)
DCD 0x11001757        ;P(PM)
DCD 0x0D00aa40        ;^(M)
DCD 0x09002223        ;1(12h)
DCD 0x02003123        ;2(12h)
DCD 0x00004755        ;4(24h)
DCD 0x00005531        ;h
```

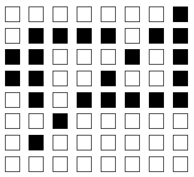
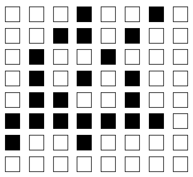
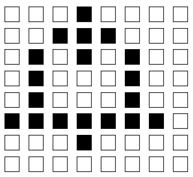
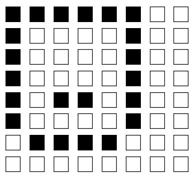
## Font Data Image

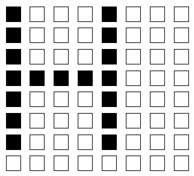
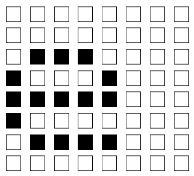
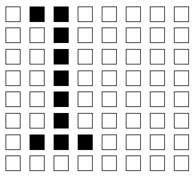
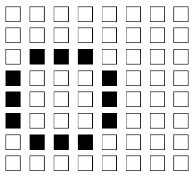
0

1

2

3


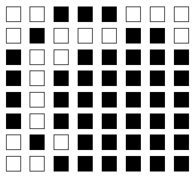
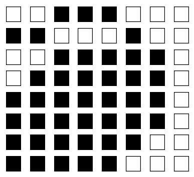
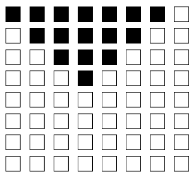
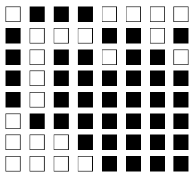
4

5

6

7


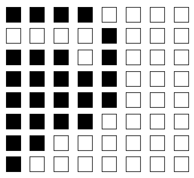
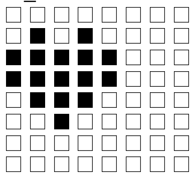
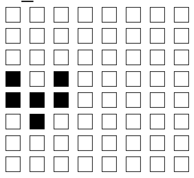
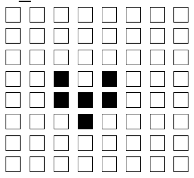
8

9

“ “

null


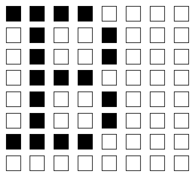
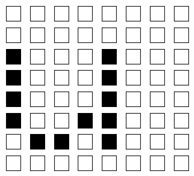
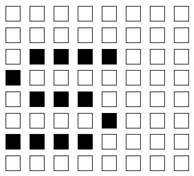
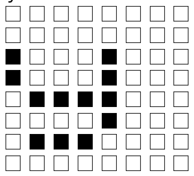
lock

speaker

loud

spkoff


batlow

alloff

alarm

mcard


H

e

l

o


hart01

hart02

hart03

hart11


hart12

s_hart2

s_harSL

s_harSS


B

u

s

y


| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| ■■■□ | ■■□□ | ■■■□ | ■■■□ | □□■□ | ■■■□ | ■■■□ | ■■■□ |
| ■□■□ | □■□□ | □□■□ | □□■□ | □■■□ | ■□□□ | ■□□□ | □□■□ |
| ■□■□ | □■□□ | ■■■□ | ■■■□ | ■□■□ | ■■■□ | ■■■□ | □■□□ |
| ■□■□ | □■□□ | ■□□□ | □□■□ | ■■■□ | □□■□ | ■□■□ | □■□□ |
| ■■■□ | □■□□ | ■■■□ | ■■■□ | □□■□ | ■■■□ | ■■■□ | □■□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |

| 8 | 9 | “ ” | : | / | “ “ | “ “ | “ “ |
|---|---|---|---|---|---|---|---|
| ■■■□ | ■■■□ | □□□□ | □□□□ | □□■□ | □■□□ | ■□■□ | □□□□ |
| ■□■□ | ■□■□ | □□□□ | □■□□ | □□■□ | □■□□ | ■□■□ | □□□□ |
| ■■■□ | ■■■□ | □□□□ | □□□□ | □■□□ | ■□□□ | □■■□ | □□□□ |
| ■□■□ | □□■□ | □□□□ | □■□□ | □■□□ | ■□□□ | □□■□ | □□□□ |
| ■■■□ | ■■■□ | □□□□ | □□□□ | ■□□□ | ■□□□ | □■□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |

| a | d | e | F | h | M | n | o |
|---|---|---|---|---|---|---|---|
| □■□□ | □□■□ | □■□□ | □■■■ | ■□□□ | □□□■ | □□□□ | □□□□ |
| □□■□ | □□■□ | ■□■□ | □■□□ | ■□□□ | ■□■■ | □□□□ | □□□□ |
| □■■□ | ■■■□ | ■■■□ | □■■■ | ■■■□ | □■□■ | ■■■□ | ■■■□ |
| ■□■□ | ■□■□ | ■□□□ | □■□□ | ■□■□ | □□□■ | ■□■□ | ■□■□ |
| ■■■■ | ■■■□ | □■■□ | □■□□ | ■□■□ | □□□■ | ■□■□ | ■■■□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |

| r | S | T | t | u | W | i | ! |
|---|---|---|---|---|---|---|---|
| □□□□ | □□■■ | □■■■ | □□□□ | □□□□ | □■□■ | □■□□ | ■□□□ |
| ■□■□ | □■□□ | □□■□ | □■□□ | □□□□ | □■□■ | □□□□ | ■□□□ |
| ■■□□ | □■■■ | □□■□ | ■■■□ | ■□■□ | □■□■ | □■□□ | ■□□□ |
| ■□□□ | □□□■ | □□■□ | □■□□ | ■□■□ | □■□■ | □■□□ | ■□□□ |
| ■□□□ | □■■□ | □□■□ | □■■□ | ■■■□ | ■□■□ | □■□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |

| A(AM) | P(PM) | ^(M) | 1(12h) | 2(12h) | 4(24h) | h |
|---|---|---|---|---|---|---|
| □■□□ | ■■■□ | □□□□ | ■■□□ | ■■□□ | ■□■□ | ■□□□ |
| ■□■□ | ■□■□ | □□■□ | □■□□ | □■□□ | ■□■□ | ■■□□ |
| ■■■□ | ■■■□ | □■□■ | □■□□ | ■□□□ | ■■■□ | ■□■□ |
| ■□■□ | ■□□□ | □■□■ | □■□□ | ■■□□ | □□■□ | ■□■□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |
| □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ | □□□□ |