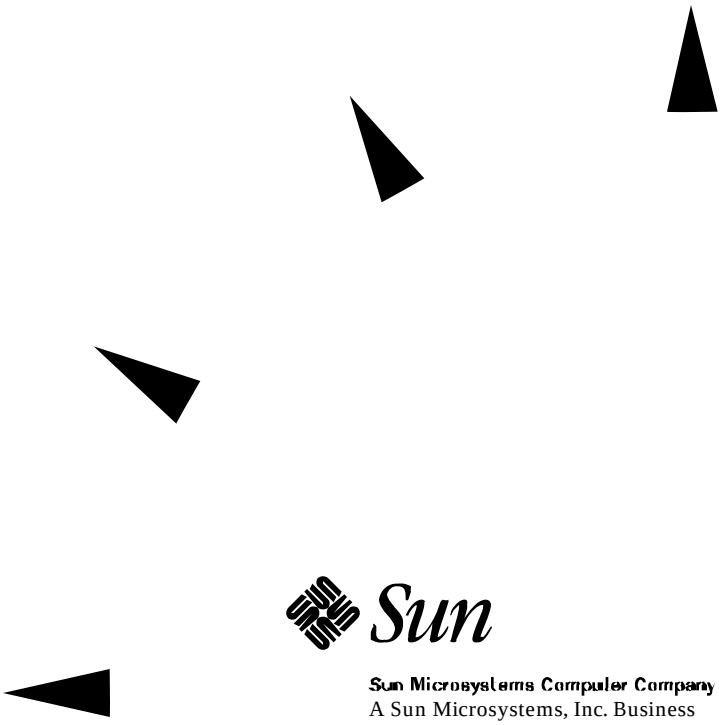
Solaris Reference Manual for SMCC-Specific Software



2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 801-7973-10
Revision A, November 1994

Sun Microsystems Computer Company
A Sun Microsystems, Inc. Business

Preface

This book, *Solaris Reference Manual for SMCC-Specific Software*, publishes man pages that are delivered on the SMCC Updates CD for Solaris 2.4. This book supplements the Solaris 2.4 Reference Manual set.

The man pages described in this book can be installed using the following packages present on the SMCC Updates CD:

- `SUNWpmman` — man pages related to Power Management software.
  - `cpr(7)`
  - `dtpower(1M)`
  - `pm(7)`
  - `pmconfig(1m)`
  - `power.conf(4)`
  - `powerd(1M)`
  - `speckeysd(1M)`
  - `sys-suspend(1M)`
- `SUNWtcxmn` — man pages related to TCX graphics.
  - `tcxconfig(1M)`
- `SUNWvygmn` — man pages related to SPARCstation Voyager software.
  - `mic(7)`
  - `pmc(7)`

For more information about installing these man page packages, see the *Solaris 2.4 SPARC Hardware Platform Guide for SMCC.*

How This Book Is Organized

Each chapter presents man pages that supplement one of the sections of the Solaris 2.4 Reference Manual set:

- **Chapter 1, "Maintenance Commands (1M)**"
- **Chapter 2, "File Formats (4)"**
- **Chapter 3, "Special Files (7)"**

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename.*
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide.* These are called *class* options. You *must* be root to do this.

Contents

Maintenance Commands (1M) 1

This chapter includes the following man pages to supplement section 1M of the Solaris 2.4 man pages:

- `dtpower(1M)`
- `pmconfig(1M)`
- `powerd(1M)`
- `speckeysd(1M)` [covered in the `sys-suspend(1M)` man page]
- `sys-suspend(1M)`
- `tcxconfig(1M)`

1

dtpower *(1M)*

Name

dtpower — Desk-top power manager, system and device power management tool.

Synopsis

```
dtpower [generic-tool-arguments] [-sampleTime n | -st n]
      [-warnTime1 n | -wt1 n] [-warnTime2 n | -wt2 n] [-nobell]
```

Availability

SUNWpmow

Description

dtpower provides a graphical user interface (GUI) to the power management system (see pm(7)). It allows the user to configure certain power manageable devices to shutdown after a specified period of inactivity. Different hardware platforms support different devices. Most platforms allow power management of display(s). Some platforms allow power management of disk drives. The set of power configurations for all devices is called a *power profile*.

dtpower also displays the current autoshutdown settings (see powerd(1M)). If dtpower is run as root, these settings may be changed. These settings are not included in a power profile.

If a battery is present, dtpower monitors the battery level. If the system is running from the battery, dtpower displays low power warnings when the battery charge is running low. dtpower maintains two device power profiles: One for use on AC and one for use with the battery. This enables you to customize your device power settings, depending on your power source. dtpower switches profiles automatically when the machine's power supply changes. There may be a small delay (about 30 seconds) before dtpower notices a change in power source.

You must be console owner or root to run dtpower.

Options

generic-tool-arguments

`dtpower` accepts the generic tool arguments described in `xview(7)`.

`-sampleTime` *n*
`-st` *n*

`dtpower` continually checks the battery capacity, if a battery is present. This option sets the period of this check. The default is 10 seconds.

`-warntime1` *n*
`-wt1` *n*
`-warntime2` *n*
`-wt2` *n*

`dtpower` displays two warnings of low battery power. These options set the time before battery exhaustion at which the warnings will occur. The default warning times are at 10 and 5 minutes. Note that `powerd(1M)` will shut the system down when the battery is exhausted.

`-nobell`

By default, whenever `dtpower` displays a warning dialog, it sounds a bell. This option disables the bell.

Usage

`dtpower` operates via a set of pull-down menus, slider(s), and buttons in a control panel. From the control panel you may access one other panel, the autoshutdown panel.

The Control Panel

Menu Bar

File

Exit

Exits the application. If you have pending changes, you will be prompted to apply or discard them before exiting.

Help

Help

Displays an overview of the dtpower application.

Information

With Battery

The charge level of the battery is displayed. If the battery is the connected power source, then the estimated battery life is displayed below the charge gauge.

You can select which device power profile to edit using the toggle buttons above the slider(s). Note that the active profile is determined by your power source, not the toggle buttons.

Without battery

The power profile displayed is for an AC power supply. There is no access to the battery power profile.

Slider(s)

Screen

This slider shows the amount of time the keyboard and mouse will be unused before the screen turns off. To change this time, move the slider and select apply. To turn the screen on, move the mouse or press a key.

Disk

This slider shows the amount of time the disk will be idle before spinning down. This is not available on all platforms. The disk will automatically spin up the next time it is needed.

Buttons

Apply

This applies any changes to you active power profile and saves all settings in to `$HOME/.pmrc` so they are remembered the next time the application is started.

Reset to Standard

This resets the active power profile to its default values and saves these into `$HOME/.pmrc`.

Set Autoshutdown...

This brings up the autoshutdown panel.

The Autoshutdown Panel

As root

This panel allows you to view and edit the parameters governing autoshutdown. The first box adjusts the amount of time the console keyboard and mouse must be unused before the system will autoshutdown.

The toggle buttons beneath determine the times when autoshutdown is in effect at all.

OK

This applies any changes made and saves them to `power.conf(4)` as the default settings.

Cancel

Dismisses the window and discards any changes.

Help

Displays a brief overview text.

As a regular user

This panel allows you to view the current settings. Changes are not permitted.

OK

Disabled.

Cancel

Dismisses window.

Help

Displays a brief overview text.

Files

`$HOME/.pmrc`

Per user customized power profile.

`/etc/power.conf`

System-wide power configuration profile.

`/usr/openwin/lib/app-defaults/Dtpower`

Text messages file

See Also

`cpr(7)`, `pm(7)`, `power.conf(4)`, `pmconfig(1M)`, `powerd(1M)`

Modified

23 Oct 1994

pmconfig *(1M)*

Name

pmconfig – Configure the power management system

Synopsis

```
/usr/sbin/pmconfig

/usr/sbin/pmconfig [-r]
```

Availability

SUNWpmu

Description

pmconfig enables the current system autoshutdown information to be viewed and/or the power management configuration modified. pmconfig reads in the configuration file power.conf(4) and issues commands to make this power configuration active. This may involve commands to the power management pseudo driver (pm(7) or a signal to the power daemon (powerd(1M)). If no daemon is present and autoshutdown information is present, a daemon will be started.

Errors

If the program cannot open either the pseudo driver or the configuration file it prints an error message to standard error. If the program encounters a syntax error in the configuration file, it prints an error message and the line number of the error in the configuration file. It then skips the rest of the information on that line and processes the next line. Any configuration information already processed on the line containing the error is *used*.

All error messages start with "pmconfig (*line n*):", and may be followed by:

Can't find device name :

The first field is not a device name.

`Can't find threshold value :`

The field following the device name is not an integer.

`Too many threshold values :`

More idle times than the device supports were given.

`Unrecognizable dependent name :`

The dependent field is not a device name.

A standard error message

Returned from the `pm` driver.

Options

`-r`

Reset all power managed devices to unconfigured

Files

`/etc/power.conf`

System power management configuration file

See Also

`pm(7)`, `power.conf(4)`, `powerd(1M)`

Modified

5 Jul 1994

powerd *(1M)*

Name

powerd — Power manager daemon

Synopsis

```
/usr/lib/power/powerd [-n]
```

Availability

SUNWpmu

Description

This daemon manages two types of system shutdown. The two types of shutdown are automatic shutdown, set on a daily basis, and low power shutdown on systems which support battery operation. If the system suspend module, cpr(7), is present, it will be used to shut the system down, otherwise the poweroff(1M) utility will be used. The autoshutdown information is read from the file /etc/power.conf by the daemon. It is reread whenever the daemon receives a hangup signal, SIGHUP.

Automatic shutdown can occur when two conditions are met. The current time is between the start and finish times, and the system has been idle for at least the set time period. System idleness is determined by inactivity on the console keyboard and mouse.

The start and finish times are specified in the file in 24-hour time notation, measured since the start of the day (12:00 am). If the finish time is less than or equal to the start time, the active period of the daemon will span from midnight to the finish time and from the start time to the following midnight. Thus to specify continuous operation, the finish time may be set equal to the start time. Specifying a negative idle time disables automatic shutdowns from occurring.

Low power shutdown will occur if the system is running from battery and the daemon monitors that the charge in the battery is too low to reliably continue operation.

Immediately prior to system shutdown, the daemon notifies `syslogd(1M)` of the shutdown, which broadcasts the notification.

Options

`-n`

No broadcast mode. The daemon will shutdown the system silently without notifying `syslogd(1M)`.

Files

`/etc/power.conf`

Used to obtain the current daemon autoshutdown settings.

Notes

The daemon uses shared memory IPC, which may increase the system image size if the shared memory module has not already been loaded.

The daemon ensures that only one daemon is running. If another daemon is running, then the new daemon will exit with and error. if the daemon dies unexpectedly (non-maskable signal) then residual shared memory state will remain. Starting a new daemon will remove this residual state.

See Also

crp(7), pm(7), pmconfig(1M), power.conf(4), poweroff(1M0, syslogd(1M)

Modified

8 Jul 1994

sys-suspend *(1M)*

Name

`sys-suspend` — Suspend the system and power off

`speckeysd` — Detects special keys on Type 4 or Compact 1 keyboard.

Synopsis

```
/usr/openwin/bin/sys-suspend [-fnx]

/usr/openwin/bin/speckeysd
```

Availability

`SUNWpmow`

Description

sys-suspend(1M) invokes the uadmin(1M) system call with the right options to syspend the whole system. A system can be suspended to conserve power or to prepare the system for transport. It should not be used in place of a shutdown when performing any hardware reconfiguration or replacement.

The current system state will be preserved until a resume operation is performed (the next power on).

On a resume from a manually initiated suspend in the windows environment, the system brings up xlock(1) to make certain that only the same person who suspended the system can have access to the system. In a non-windows environment, the user will be prompted for a password. If the suspend was initiated by the powerd(1M)—also known as AutoShutdown—mechanism, no additional security measure is initiated. It is the user's responsibility to secure his/her work session before AutoShutdown takes place.

It is possible that when devices or processes are performing critical or time sensitive operations (such as real time operations), the system may fail to suspend. When this occurs, the system will remain in tis current running state.

Messages reporting the failure will be displayed on the console. Once the system is successfully suspended, the resume operation will always succeed barring external influences such as hardware reconfiguration or the like.

speckeysd(1M) is a daemon that is started at OpenWIndows start time to pick up the power and contrast key strokes from Type 5 and COmpact 1 keyboards.

The *Power* key is the initiator of a manual suspend in sys-suspend fashion. There are several different ways of using the *Power* key.

Power key

Displays a confirmation popup, prompting the user to confirm the suspend request.

Shift+Power keys

Initiates an immediate suspend.

The contrast keys (which are next to the Power key) control the screen contrast on some systems. To change screen contrast, use the Shift key and the appropriate contrast key: The *Contrast+* for increasing the screen contrast and *Contrast-* for decreasing the screen contrast.

Options

`-f`

Force suspend. This should be used with care. Using this options causes the system to force stops all processes that does not through the default mechanism. This option should be used only during unattended operations.

`-n`

Disable confirmation. This flag disables the confirmation popup dialog at suspend time.

`-x`

Disable lockscreen. This flag disables the execution of lockscreen at resume time.

Files

`/kernel/misc/cpr`

Loadable module for cpr

`/cprboot`

Special bootstrapper for cpr

`/.CPR`

System state file

`/.cpr_generic_info`

sys-suspend control file

`/.cpr_defaultboot_info`

sys-suspend control file

Notes

`xlock(1)` on resume can be disabled by default. The following line needs to be added to the user's `.Xdefaults` or `.OWdefaults` file:

```
Speckeysd*xlock:      False
```

The xlock mode defaults to *life*. This can be changed by adding the following line to the user's .Xdefaults or .OWdefaults file:

```
Speckeysd*xlockmode:<xlockmode>
```

See Also

`uadmin(2)`, `cpr(7)`

Modified

8 Aug 1994

tcxconfig *(1M)*

Name

tcxconfig — Configure the default linearity of the 24-bit TrueColor Visual for OpenWindows on TCX.

Synopsis

```
/usr/sbin/tcxconfig [linear | nonlinear]
```

Availability

SUNWtcxow

Description

The tcxconfig script changes the default linearity of a 24-bit TrueColor Visual for OpenWindows on TCX. When the TCX driver for OpenWindows is installed, the default 24-bit TrueColor Visual is nonlinear. You can run tcxconfig with an argument that specifies the setting you want.

OpenWindows should not be running when you run the tcxconfig script. Start OpenWindows after tcxconfig has set the linearity you desire.

Options

Entering tcxconfig without options displays the current default setting

linear

Set linear visual to be the default 24-bit TrueColor Visual. This means color will be gamma-corrected.

nonlinear

Set nonlinear visual to be the default 24-bit TrueColor Visual.

Modified

23 Aug 1994

1

File Formats (4) 2

This chapter includes the following man pages to supplement section 4 of the Solaris 2.4 man pages:

- `power.conf(4)`

power.conf *(4)*

Name

power.conf — Power management configuration information file.

Synopsis

```
/etc/power.conf
```

Availability

SUNWpmr

Description

The power.conf file is used by the power management configuration program, pmconfig(1M), to initialize the settings for power management of the system. Devices not appearing in this file will not be power managed without explicit configuration using the power management pseudo driver. It is recommended that you fully understand the power management framework before modifying this file. Although inappropriate settings will not cause system damage, severe performance reduction may result.

The power.conf file contains a line by line listing of the devices to be configured. A line may be more than 80 characters. A new line character will be treated as white space, if it is preceded by a backslash (“\”). Comment lines must begin with a hash character (“#”). Each line must contain a device name field and a threshold field; it may also contain a dependents field. The fields must be in that order (name, threshold, dependents). Fields and subfields are separated by white space (tabs or spaces).

The device name field specifies the device to be configured. The name is a pathname either leading to a special file or specifying the device as in the /etc/path_to_inst file. When using the latter format, instead of using the full pathname, it is possible to omit the portion of the pathname specifying the device’s parent(s) (parent devices). This includes the leading “/”. Using this “relative” pathname format, the first device whose full pathname contains the given pathname as its tail is matched.

For example, SCSI disk target 1 with the following full path name:

```
/iommu@f,e000/sbus@f,e001/espdma@f,4000/esp@f,8000/sd@1,0
```

May also be specified as:

```
sbus@f,e000/espdma@f,4000/esp@f,8000/sd@1,0
esp@f,8000/sd@1,0
sd@1,0
```

The threshold field is used to configure the power manageable components of a device. These components represent the separately power manageable entities within a device. This field may contain as many integer values as the device has components. Each threshold time specifies the idle time in seconds before the respective component may be powered down. if there are fewer component threshold times than device components, the remaining components are not power managed. To explicitly set a component to not turn off, use a value of –1. At least one component threshold must be specified per device (in the file).

The dependent field may contain a list of logical dependents of this device. A logical dependent is a selected device that is not physically connected to the power managed device (e.g., display and the keyboard). A dependent device is one which must be idle and powered down before the managed device may be powered down (over and above any threshold times). The names use the same formats allowed in the first field and a re separated by white space. A device must be previously configured before it may be used as a dependent.

The device name field may also contain the special device name “autoshutdown”, which indicates that the line is to initiate the `powerd(1M)`. In this case, the threshold value specifies the system idle time (measured by console keyboard and mouse) before the system may be shutdown. The line must also contain start and finish times which also determine when the system may be automatically suspended.

Examples

The following is a sample power.conf file:

```
# This is a sample power management configuration file
# Fields must be separated by white space.
#
```

```
# Name            Threshold(s)     Logical Dependent(s)
/dev/conskbd      30
/dev/consms       30
/dev/fb           0                 /dev/conskbd  /dev/consms
/dev/fbs/cgsix1   0                 /dev/conskbd   #second display
/dev/dsk/c0t3d0s0 20
esp@4,8800000     0
le@4,8c00000      0
zs@0,0            0 0 0
zs@0,100000       0 0 0

# Auto-Shutdown      Idle Time (min)   Start/Finish Times (hh:mm)
autoshutdown         60                 17:00 8:00
```

Notes

Remember that *physical* dependents are automatically included by the power manager and need not be specified.

The default `power.conf` file suports the standard hardware configuration. for each additional power manageable device (e.g., a second display), a new entry must be manually added to the `power.conf` file and `pmconfig(1M)` executed to activate the new change.

Powering devices up and down frequently may reduce device reliability, especially for devices not designed for power management. So do not put additional devices under power management unless the hardware documentation permits it. At this time most SCSI hard disks are not power manageable.

See Also

`pm(7)`, `pmconfig(1M)`, `powerd(1M)`

Special Files (7) 3

This chapter includes the following man pages to supplement section 7 of the Solaris 2.4 man pages:

- `cpr(7)`
- `mic(7)`
- `pm(7)`
- `pmc(7)`

cpr *(7)*

Name

cpr — Suspend and resume module.

Synopsis

```
/kernel/misc/cpr
```

Availability

```
SUNWcpr
```

Description

cpr is a loadable module which is used to suspend and resume the whole system. You may wish to suspend a system to save power, or to temporarily power off for transport. It should not be used in place of a normal shutdown when performing any hardware reconfiguration or replacement. In order for resume to succeed, it is important that the hardware configuration remain the same. When the system is suspended, the entire system state is preserved in nonvolatile storage until a resume operation is conducted.

The principle way to suspend the system using this module is through the sys-suspend(1M) command. There are other utilities which may be installed on your system which will also access this module (such as uadmin(1M), uadmin(2), or the *Power* key and the *Shift+Power* key on a type 5 keyboard).

The module performs the following actions when suspending the system. The signal SIGFREEZE is first sent to all user threads and then the threads are stopped. The system is brought down to a uniprocessor mode for multiprocessor systems. Next, dirty user pages are swapped out to their backing storage device and all file systems are synchronized. All devices are made quiescent and system interrupts are disabled. To complete the system suspend, the kernel memory pages and remaining user pages are written to the root file system in a compressed form.

When the system is powered on again, essentially the reverse of the suspend procedure occurs. The kernel image is restored from the root file system by the bootstrapper `/cprboot`, interrupts and devices are restored to their previous state. Finally the user threads are rescheduled and SIGTHAW is broadcast to notify any interested processes of system resumption. Additional processors, if available, are restored and brought online. The system is now back to exactly the state prior to suspension.

In some cases the `cpr` module may be unable to perform the suspend operation. If a system contains additional devices outside the standard shipped configuration, it is possible that these additional devices may not support `cpr`. In this case, the suspend will fail and an error message will be displayed to that effect. These devices must be removed or its device drivers unloaded for suspend to work. Contact the device manufacturer to obtain a new version of device driver that supports `cpr`. A suspend may also fail when devices or processes are performing critical or time sensitive operations (e.g., real time operations). In this case the system will remain in its current running state. Messages reporting the failure will be displayed on the console and status returned to the caller. Once the system is successfully suspended the resume operation will always succeed barring external influences such as hardware reconfiguration or the like.

Some network based applications may fail across a suspend and resume cycle. This largely depends on the underlying network protocol and the applications involved. In general, applications that retry and automatically reestablish connections will continue to operate transparently on resume, those applications that do not, will likely fail.

The speed of suspend and resume can range from 15 seconds to around a minute, depending on the system speed, memory size, and load.

Files

`/cprboot`

Special bootstrapper for `cpr`.

`/.CPR`

System state file.

`/.cpr_generic_info`

sys-suspend control file.

`/.cpr_defaultboot_info`

sys-suspend control file.

Bugs

The signals SIGFREEZE and SIGTHAW are not properly implemented for the Solaris 2.4 release; these will be available in a later release. This should only be a concern for specially customized applications that need to perform additional tasks at suspend or resume time, of which none exist at the present time.

In extremely rare occasion the system may fail during the early stages of a resume. In this small window, it is theoretically possible to be stuck in a loop that the system does not resume and it does not boot normally. If you are in such a loop, get to the prom `ok` prompt via *L1+A* and enter the following command:

```
set-default boot-file
```

This will reset the system and allow it to boot normally.

Notes

For suspend/resume to work on multiprocessor platforms, it must be able to control all CPUs. It is recommended that no MP tests (such as sundiag CPU tests) are running when suspend is initiated because the suspend may be rejected, if it cannot shut off all CPUs.

Certain device operations such as tape or floppy disk activities are not resumable due to the nature of removable media. These activities are detected at suspend time, and must be stopped before suspend will complete successfully.

See Also

`sys-suspend(1M)`, `uadmin(1M)`, `uadmin(2)`

Modified

5 Jul 1994

mic *(7)*

Name

mic — Multi-interface Chip driver.

Synopsis

```
#include <fcntl.h>
#include <sys/termios.h>
#include <sys/micro.h>

open("/dev/term/mic/a", mode);
open("/dev/term/mic/b", mode);
open("/dev/term/mic/ir", mode);
```

Availability

SUNWmic

Platform

SPARCstation Voyager

Description

The Multi-interface Chip (MIC) provides two asynchronous serial input/output channels. These channels provide high speed buffered serial I/O, with optional hardware flow control support. Baud rates from 110 to 115200 are supported.

The first channel can either be routed through an infra-red port of the "a" serial port. If the device is opened using the "ir" device, then the driver routes the first channel through the infra-red port. If the device is opened using the "a" device, the first channel is routed through the "a" serial port. You cannot use both the "a" port and the "ir" port simultaneously. the second channel (the "b" serial port) has no infra-red capability and may be used independently of the first channel.

The `mic` module is a loadable STREAMS driver that provides basic support for the MIC hardware, together with basic asynchronous communication support. The driver supports those `termio(7)` device control functions specified by flags in the `c_cflag` word of the `termios` structure, excluding HUPCL, CLOCAL, CIBAUD, CRTSCTS, and PAREXT. The driver does not support device control functions specified by flags in the `c_iflag` word of the `termios` structure. Specifically, the driver assumes that IGNBRK and IGNPAR are always set. All other `termio(7)` functions must be performed by STREAMS modules pushed atop the driver. When a device is opened, the `ldterm(7)` and `ttcompat(7)` STREAMS modules are automatically pushed on top of the stream, providing the standard `termio(7)` interface.

The infra-red port provides access to two different modes of modulation. The default mode is called pulse mode and is compatible with the Infra-red Data Association (IrDA) modulation and the Hewlett-Packard Serial Infra-red (SIR) modulation. The second modulation is called high frequency mode and is compatible with the Sharp Amplitude Shift Keying (ASK) modulation. The default modulation when using high frequency mode is 500 KHz.

The character-special devices `/dev/term/mic/a` and `/dev/term/mic/b` are used to access the two serial ports on the MIC chip.

The character-special device /dev/term/mic/ir is used to access the infra-red port of the chip.

IOCTLS

The standard set of `termio` `ioctl()` calls are supported by the `mic` driver.

Breaks can be generated by the TCSBRK, TIOCSBRK, and TIOCCBRK `ioctl()` calls.

The input and output line speeds may be set to any of the speeds supported by termio. The speeds cannot be set independently; when the output speed is set, the input speed is set to the same speed. To support higher speeds than defined in termio the two lowest speeds, B50 and B75, have been remapped to 96000 and 115200 baud respectively.

There are six ioctl() calls which are specific to the infra-red port and can only be used when the device has been opened in infra-red mode:

MIOCGETM_IR

Returns the current IR mode defined in micio.h.

MIOCSETM_IR

Takes an additional argument of the desired IR mode (defined in micio.h) and sets the port to this mode.

MIOCGETD_IR

Returns the current IR carrier divisor. The carrier frequency can be calculated from the divisor and the formula:

carrier frequency = 19660 / (4 (*divisor* + 1)) KHz

MIOCSETD_IR

Sets the current IR carrier divisor. The desired frequency can be set by using a divisor calculated by the following formula, where the frequency is specified in KHz:

divisor = 19660 / *frequency* / 4 – 1

MIOCSLPBK_IR

Set IR loopback mode. This enables the receiver during transmit, so that sent messages are also received through the IR port.

MIOCCLPBK_IR

Clears IR loopback mode.

There are two `mic`-specific `ioctl()` calls:

MIOCSLPBK

Set SCC loopback mode. This internally loops back transmitted messages within the channel.

MIOCCLPBK

Clear SCC loopback mode.

Errors

An open() will fail if:

ENXIO

The unit being opened does not exist.

EBUSY

The channel is in use by another serial protocol. Remember that both the “a” and “ir” ports use the same channel.

Files

`/dev/term/mic/a`

Asynchronous serial line using port a

`/dev/term/mic/b`

Asynchronous serial line using port b

`/dev/term/mic/ir`

Asynchronous serial infra-red line using the infra-red port.

Diagnostics

`mic: Rx FIFO overflow`

The mic’s internal 64 character buffer overflowed before it could be serviced.

`mic: Rx buffer full - draining`

The driver’s character input buffer overflowed before it could be serviced.

Notes

Currently hardware flow control is not implemented. The state of DCD, CTS, RTS, and DTR interface signals cannot be queried, nor can hardware flow control be enabled using the CRTSCTS flag in the `c_cflag` word of the `termios` structure.

See Also

`tip(1)`, `ports(1M)`, `ioctl(2)`, `open(2)`, `ldterm(7)`, `termio(7)`, `ttcompat(7)`

Modified

6 Sep 1994

pm *(7)*

Name

pm — Power Management driver

Synopsis

```
#include <sys/pm.h>
int ioctl(int fildes, int command, int arg);
```

Availability

SUNWpmu

Description

The Power Management driver provides an interface for applications to configure the devices within the system for power management. The interface is provided through `ioctl(2)` commands. The `pm` driver may be accessed using `/dev/pm`.

fildes is an open file descriptor that refers to the `pm` driver.

command determines the control function to be performed as described below.

arg represents additional information that is needed by this command. The type of *arg* depends upon the command, but it is generally an integer or a pointer to a command-specific data structure.

Command Functions

Unless configured by using the commands below, `pm` does not power manage devices by default. Note, however, that the `pmconfig(1M)` program is typically run at boot time, and by reading the `power.conf(4)` file will use the commands below to configure `pm`. Any devices configured for power management by `pm` will have their drivers loaded (if not already) and locked into memory until that device is unmanaged. Some devices may be able to fully operate at non-full power levels. Using the command `PM_SET_POWER` on

such a device allows this low power mode to become the normal (on) power level for that device. This mode of operation is distinct from the power managed mode of operation.

pm periodically searches the system for devices which it can power manage. A device will only be power managed when it is not in use (explained further below). When a power managed device is subsequently used, it will be automatically returned to normal power.

The pm model of power management is to view the system as a collection of devices. Each device is a collection of components; a component is the smallest power manageable unit. The devices, and the components within those devices that are power manageable, are dependent upon the implementation of their respective device drivers. A power manageable component has three states. It may be busy (in use), it may be idle (not in use but using normal power), or it may be power managed (not in use and not using normal power). The pm driver manages the component transition from the second to the third state. pm uses two factors to determine this transition: the component must have been idle for at least the threshold time; and the device to which the component belongs must satisfy any dependencies requirements.

A dependency is when a device requires another device to be power managed before it can be power managed. A device is considered to be power managed when all of its components are power managed. Note that dependencies occur on a per device basis; when a dependency exists, no components of a device may be managed unless all the components it depends upon are first managed.

For more information, see the *Guide to Writing Device Drivers* manual, `attach(9e)`, `detach(9e)`, and `power(9e)`.

Thus the configuration of a device for power management is the setting of the threshold for any component that is to be managed and defining any dependencies for that device.

For all commands excluding `PM_SCHEDULE`, *arg* points to a structure of type `pm_request` defined in `sys/pm.h`:

```
typedef struct {
          char      *who;          /*device to configure*/
          int       select;        /* selects the component or
                                        dependent of the device*/
          int       level;         /*power or threshold level */
```

```
        char     *dependent;     /*hold name or dependent */
        int       size;           /size of dependent buffer 8?
} pm_request
```

The fields should contain the following data:

who is a pointer to the name of the device to be configured. The name must be in the format described in `power.conf(4)`.

select is a non-negative integer specifying the component or dependent being configured. the numbering starts at zero.

level is a non-negative integer giving the threshold level in seconds or the desired power level.

dependent is a pointer to a buffer which contains or receives the name of a device on which this device has a dependency. It uses the same format as the first field.

size is the size of the dependent buffer.

Not all fields are used in each command. Upon error the commands will return –1, and set *errno* to the error condition specified below. The following error codes are common to all commands:

`EFAULT:`

Bad address passed in as argument

`ENODEV:`

Device is not power manageable, or device is not configured (use `PM_SET_THRESHOLD` command first).

`ENXIO:`

Invalid instance number (device not attached).

`EPERM:`

Permission denied. You must be root or console owner.

Error codes specific to a command will follow that command's description.

PM_SCHEDULE:

arg sets the period in seconds of pm device scans. A value of zero inhibits scans which stops any further components from being managed. A negative value is ignored. The ioctl returns the new (or current) period.

PM_GET_IDLE_TIME:

Using the fields *who* and *select*, this command returns the time in seconds since the component was last busy.

Error codes:

EINVAL:

Device component out of range.

PM_GET_NUM_CMPTS:

Using the field *who*, this command returns the number of components defined for this device.

PM_GET_THRESHOLD:

Using the fields *who* and *select*, this command returns the threshold level of the component.

Error codes:

EINVAL:

Device component out of range.

PM_SET_THRESHOLD:

Using the fields *who*, *select*, and *level*, this command sets the threshold level of the component. It returns zero on success.

Error codes:

EINVAL:

Device component out of range, or threshold value < 0.

PM_GET_POWER

Using the fields *who* and *select*, this command returns the current normal power level of the component.

Error codes:

`EINVAL:`

Device component out of range.

`EIO:`

Non-power manageable device (or properties are removed).

`PM_SET_POWER:`

Using the fields *who*, *select*, and *level*, this command sets the current normal power level of the component to the given power level.

Error codes:

`EINVAL:`

Device component out of range, or power level <= 0.

`EIO:`

Failed to power device or its parent or its dependents.

`PM_GET_CUR_PWR:`

Using the fields *who* and *select*, this command returns the current power level of the component.

Error codes:

`EINVAL:`

Device component out of range.

`PM_GET_NUM_DEPS:`

Using the field who, this command returns the number of dependents configured for this device.

`PM_GET_DEP:`

Using the fields *who*, *select*, *level*, and *dependent*, this command writes the name of the dependent into the buffer supplied by the dependent field.

Error codes:

`EINVAL:`

Dependent component out of range, or user buffer is too small for dependent name.

`EFAULT:`

Bad buffer address was given.

`PM_ADD_DEP:`

Using the fields *who* and *dependent*, this command adds the dependent to the device.

Error codes

`ENODEV:`

Device is non-power manageable or is not configured.

`PM_REM_DEP:`

Using the fields *who* and *dependent*, this command removes the dependent from the device.

Error codes:

`ENODEV:`

Device is non-power manageable or is not configured, or the device has no dependents.

`PM_REM_DEVICE:`

Using the field *who*, this command unmanages the device and returns the device to normal power, if it is not already.

`PM_REM_DEVICES:`

This command unmanages all devices and returns them to normal power.

Notes

To unload a power managed driver, the driver must first be unmanaged using `PM_REM_DEVICE` or `PM_REM_DEVICES`.

Currently it is not an error to remove a nonexistent dependent or add a repeated dependent. The pseudo driver will silently ignore the redundant command.

See Also

`intro(2)`, `ioctl(2)`, `pmconfig(1M)`, `power.conf(4)`, `attach(9e)`, `detach(9e)`, `power(9e)`

Modified

5 Jul 1994

pmc *(7)*

Name

pmc — Platform Management Chip driver.

Synopsis

```
#include <sys/pmcio.h>
int ioctl(int fildes, int command, int arg);
```

Availability

SUNWpmc

Platform

SPARCstation Voyager

Description

The Platform Management Chip driver provides a number of miscellaneous platform specific functions. Principally these are to provide power control for devices which cannot manage their own power control (see `ddi_power(9f)`) and to provide information about the connection status of the machine. Not all functions are supported on all platforms.

The user interface is provided through `ioctl(2)` commands. The `pmc` driver may be accessed using `/dev/pmc`. The system interface (to power manage devices) is provided by registering its power function (using the "platform-pm" property of the root node).

fildes is an open file descriptor that refers to the pmc driver.

command determines the control function to be performed as described below.

arg is not used and may be any value.

Command Functions

These functions fall into three categories: connection status, power control, and miscellaneous. Connection status can be used to find out whether the following devices are plugged in: keyboard, ethernet, and ISDN.

The power control function fontrols the removal of the platform power. Miscellaneous functions enable the reading of the digital to analog converter.

`PMC_GET_KBD:`

This command returns the connection status of the keyboard. When the keyboard is connected it will return `PMC_KB_STAT`, and zero when it is not connected.

`PMC_GET_ENET:`

This command returns the connection status of the ethernet. When the ethernet is connected it will return `PMC_ENET_STAT`, and zero when it is not connected.

`PMC_GET_ISDN:`

This command returns the connection status of the ISDN channels. The return value is a bit map of the connected channels: `PMC_ISDN_ST0` for `NT` and `PMC_ISDN_ST1` for `TE`.

`PCM_GET_A2D:`

This command returns the result of an eight bit analog to digital conversion. The meaning of the reading is platform specific.

`PMC_POWER_OFF:`

This command is only available to the super-user. It turns off all power to the system. Note that critical data may be lost if proper preparation prior to power removal is not performed.

The `poll(2)` interface is supported. It may be used to poll for connection status changes. A process wishing to detect such connection changes should use the `POLLIN` event flag. When ANY connection status changes, the `poll(2)` mechanism will be notified. It is up to the user to verify whether the connection status change is of interest.

Errors

`EPERM`

Must be privileged user to use `PMC_POWER_OFF`.

See Also

`ddi_power(9f)`, `intro(2)`, `ioctl(2)`, `open(2)`, `pm(7)`, `poll(2)`

Modified

5 Oct 1994