NAME

setpci – configure PCI devices

SYNOPSIS

setpci [**options**] **devices operations**...

DESCRIPTION

setpci is a utility for querying and configuring PCI devices.

All numbers are entered in hexadecimal notation.

Root privileges are necessary for almost all operations, excluding reads of the standard header of the configuration space on some operating systems. Please see **lspci(8)** for details on access rights.

OPTIONS

General options

-v Tells *setpci* to be verbose and display detailed information about configuration space accesses.

-f Tells *setpci* not to complain when there's nothing to do (when no devices are selected). This option is intended for use in widely-distributed configuration scripts where it's uncertain whether the device in question is present in the machine or not.

-D 'Demo mode' -- don't write anything to the configuration registers. It's useful to try **setpci -vD** to verify that your complex sequence of **setpci** operations does what you think it should do.

--version
Show *setpci* version. This option should be used stand-alone.

--help Show detailed help on available options. This option should be used stand-alone.

--dumpregs
Show a list of all known PCI registers and capabilities. This option should be used stand-alone.

PCI access options

The PCI utilities use the PCI library to talk to PCI devices (see **pcilib**(7) for details). You can use the following options to influence its behavior:

-A <method>
The library supports a variety of methods to access the PCI hardware. By default, it uses the first access method available, but you can use this option to override this decision. See **-A help** for a list of available methods and their descriptions.

-O <param>=<value>
The behavior of the library is controlled by several named parameters. This option allows one to set the value of any of the parameters. Use **-O help** for a list of known parameters and their default values.

-H1 Use direct hardware access via Intel configuration mechanism 1. (This is a shorthand for **-A intel-conf1**.)

-H2 Use direct hardware access via Intel configuration mechanism 2. (This is a shorthand for **-A intel-conf2**.)

-G Increase debug level of the library.

DEVICE SELECTION

Before each sequence of operations you need to select which devices you wish that operation to affect.

-s [[[[<domain>]:]<bus>]:][<slot>][.[<func>]]
Consider only devices in the specified domain (in case your machine has several host bridges, they can either share a common bus number space or each of them can address a PCI domain of its own; domains are numbered from 0 to ffff), bus (0 to ff), slot (0 to 1f) and function (0 to 7). Each component of the device address can be omitted or set to "*", both meaning "any value". All numbers are hexadecimal. E.g., "0:" means all devices on bus 0, "0" means all functions of device 0 on any bus, "0.3" selects third function of device 0 on all buses and ".4" matches only the fourth function of each device.

-d [<vendor>]:[<device>]
Select devices with specified vendor and device ID. Both ID's are given in hexadecimal and may be omitted or given as "*", both meaning "any value".

When **-s** and **-d** are combined, only devices that match both criteria are selected. When multiple options of the same kind are specified, the rightmost one overrides the others.

OPERATIONS

There are two kinds of operations: reads and writes. To read a register, just specify its name. Writes have the form *name*=*value*,*value*... where each *value* is either a hexadecimal number or an expression of type *data*:*mask* where both *data* and *mask* are hexadecimal numbers. In the latter case, only the bits corresponding to binary ones in the *mask* are changed (technically, this is a read-modify-write operation).

There are several ways how to identity a register:

- Tell its address in hexadecimal.
- Spell its name. Setpci knows the names of all registers in the standard configuration headers. Use '**setpci --dumpregs**' to get the complete list. See PCI bus specifications for the precise meaning of these registers or consult **header.h** or **/usr/include/pci/pci.h** for a brief sketch.
- If the register is a part of a PCI capability, you can specify the name of the capability to get the address of its first register. See the names starting with 'CAP_' or 'ECAP_' in the **--dumpregs** output.
- If the name of the capability is not known to **setpci**, you can refer to it by its number in the form CAP**id** or ECAP**id**, where **id** is the numeric identifier of the capability in hexadecimal.
- Each of the previous formats can be followed by **+offset** to add an offset (a hex number) to the address. This feature can be useful for addressing of registers living within a capability, or to modify parts of standard registers.
- Finally, you should append a width specifier **.B**, **.W**, or **.L** to choose how many bytes (1, 2, or 4) should be transferred. The width can be omitted if you are referring to a register by its name and the width of the register is well known.

All names of registers and width specifiers are case-insensitive.

EXAMPLES

COMMAND
asks for the word-sized command register.

4.w is a numeric address of the same register.

COMMAND.l
asks for a 32-bit word starting at the location of the command register, i.e., the command and status registers together.

VENDOR_ID+1.b
: specifies the upper byte of the vendor ID register (remember, PCI is little-endian).

CAP_PM+2.w
: corresponds to the second word of the power management capability.

ECAP108.l
: asks for the first 32-bit word of the extended capability with ID 0x108.

SEE ALSO

lspci(8), **pcilib**(7)

AUTHOR

The PCI Utilities are maintained by Martin Mares <mj@ucw.cz>.