NAME

ppi - user-space interface to ppbus parallel 'geek' port

SYNOPSIS

device ppi

Minor numbering: unit numbers correspond directly to ppbus numbers.

#include <dev/ppbus/ppi.h>
#include <dev/ppbus/ppbconf.h>

DESCRIPTION

The **ppi** driver provides a convenient means for user applications to manipulate the state of the parallel port, enabling easy low-speed I/O operations without the security problems inherent with the use of the */dev/io* interface.

PROGRAMMING INTERFACE

All I/O on the **ppi** interface is performed using **ioctl**() calls. Each command takes a single *uint8_t* argument, transferring one byte of data. The following commands are available:

PPIGDATA, PPISDATA
: Get and set the contents of the data register.

PPIGSTATUS, PPISSTATUS
: Get and set the contents of the status register.

PPIGCTRL, PPISCTRL
: Get and set the contents of the control register. The following defines correspond to bits in this register. Setting a bit in the control register drives the corresponding output low.
STROBE
AUTOFEED
nINIT
SELECTIN
PCD

PPIGEPP, PPISEPP
: Get and set the contents of the EPP control register.

PPIGECR, PPISECR
: Get and set the contents of the ECP control register.

PPIGFIFO, PPISFIFO
Read and write the ECP FIFO (8-bit operations only).

EXAMPLES

To present the value 0x5a to the data port, drive STROBE low and then high again, the following code fragment can be used:

```
int             fd;
uint8_t         val;

val = 0x5a;
ioctl(fd, PPISDATA, &val);
ioctl(fd, PPIGCTRL, &val);
val |= STROBE;
ioctl(fd, PPISCTRL, &val);
val &= ~STROBE;
ioctl(fd, PPISCTRL, &val);
```

BUGS

The inverse sense of signals is confusing.

The **ioctl**() interface is slow, and there is no way (yet) to chain multiple operations together.

The headers required for user applications are not installed as part of the standard system.