NAME

bpp – bi-directional parallel port driver

SYNOPSIS

SUNW,bpp@*slot***,***offset***:bpp***n*

DESCRIPTION

The **bpp** driver provides a general-purpose bi-directional interface to parallel devices. It supports a variety of output (printer) and input (scanner) devices, using programmable timing relationships between the various handshake signals.

APPLICATION PROGRAMMING INTERFACE

The **bpp** driver is an *exclusive-use* device. If the device has already been opened, subsequent opens fail with **EBUSY.**

Default Operation

Each time the **bpp** device is opened, the default configuration is **BPP_ACK_BUSY_HS** for read handshake, **BPP_ACK_HS** for write handshake, 1 microsecond for all setup times and strobe widths, and 60 seconds for both timeouts. This configuration (in the write mode) drives many common personal computer parallel printers with Centronics-type interfaces. The application should use the **BPPIOC_SETPARMS** ioctl request to configure the **bpp** for the particular device which is attached, if necessary.

Write Operation

If a failure or error condition occurs during a **write**(2), the number of bytes successfully written is returned (short write). Note that **errno** will not be set. The contents of certain status bits will be captured at the time of the error, and can be retrieved by the application program, using the **BPPIOC_GETERR** ioctl request. Subsequent **write**(2) calls may fail with the system error **ENXIO** if the error condition is not rectified. The captured status information will be overwritten each time an attempted transfer or a **BPPIOC_TESTIO** ioctl request occurs.

Read Operations

If a failure or error condition occurs during a **read**(2), the number of bytes successfully read is returned (short read). Note that **errno** will not be set. The contents of certain status bits will be captured at the time of the error, and can be retrieved by the application, using the **BPPIOC_GETERR** ioctl request. Subsequent **read**(2) calls may fail with **ENXIO** if the error condition is not rectified. The captured register information will be overwritten each time an attempted transfer or a **BPPIOC_TESTIO** ioctl request.

If the **read_handshake** element of the **bpp_transfer_parms** structure (see below) is set to **BPP_CLEAR_MEM** or **BPP_SET_MEM**, zeroes or ones, respectively, are written into the user buffer.

Read/Write Operation

When the driver is opened for reading and writing, it is assumed that scanning will take place, as scanners are the only devices supported by this mode. Most scanners require that the **SLCT_IN** or **AFX** pin be set to tell the scanner the direction of the transfer. The **AFX** line is set when the **read_handshake** element of the **bpp_transfer_parms** structure is set to **BPP_HSCAN_HS**, otherwise the **SLCT_IN** pin is set. Normally, scanning starts by writing a command to the scanner, at which time the pin is set. When the scan data is read back, the pin is reset.

IOCTLS

The following ioctl requests are supported:

BPPIOC_SETPARMS
Set transfer parameters.
The argument is a pointer to a **bpp_transfer_parms** structure. See below for a description of the elements of this structure. If a parameter is out of range, **EINVAL** is returned.

BPPIOC_GETPARMS
Get current transfer parameters.
The argument is a pointer to a **bpp_transfer_parms** structure. See below for a description of the elements of this structure. If no parameters have been configured since the device was opened, the contents of the structure will be the default conditions of the parameters (see **Default Operation** above).

BPPIOC_SETOUTPINS
Set output pin values.
The argument is a pointer to a **bpp_pins** structure. See below for a description of the elements of this structure. If a parameter is out of range, **EINVAL** is returned.

BPPIOC_GETOUTPINS
Read output pin values.
The argument is a pointer to a **bpp_pins** structure. See below for a description of the elements of this structure.

BPPIOC_GETERR
Get last error status.
The argument is a pointer to a **bpp_error_status** structure. See below for a description of the elements of this structure. This structure indicates the status of all the appropriate status bits at the time of the most recent error condition during a **read**(2) or **write**(2) call, or the status of the bits at the most recent **BPPIOC_TESTIO** ioctl request. Note: The bits in the **pin_status** element indicate whether the associated pin is active, not the actual polarity. The application can check transfer readiness without attempting another transfer using the **BPPIOC_TESTIO** ioctl. Note: The **timeout_occurred** and **bus_error** fields will never be set by the **BPPIOC_TESTIO** ioctl, only by an actual failed transfer.

BPPIOC_TESTIO
Test transfer readiness.
This command checks to see if a read or write transfer would succeed based on pin status, opened mode, and handshake selected. If a handshake would succeed, **0** is returned. If a transfer would fail, **-1** is returned, and **errno** is set to **EIO,** and the error status information is captured. The captured status can be retrieved using the **BPPIOC_GETERR** ioctl call. Note that the **timeout_occurred** and **bus_error** fields will never be set by this ioctl.

Transfer Parameters Structure

This structure is defined in **<sys/bpp_io.h>**.

```
struct bpp_transfer_parms {
      enum  handshake_t
            read_handshake;       /* parallel port read handshake mode */
      int   read_setup_time;      /* DSS register - in nanoseconds */
      int   read_strobe_width;    /* DSW register - in nanoseconds */
      int   read_timeout;         /*
                                   * wait this many seconds
                                   * before aborting a transfer
                                   */
      enum  handshake_t
            write_handshake;      /* parallel port write handshake mode */
      int   write_setup_time;     /* DSS register - in nanoseconds */
      int   write_strobe_width;   /* DSW register - in nanoseconds */
      int   write_timeout;        /*
                                   * wait this many seconds
                                   * before aborting a transfer
                                   */

};

/* Values for read_handshake and write_handshake fields */
enum        handshake_t {
      BPP_NO_HS,                  /* no handshake pins */
      BPP_ACK_HS,                 /* handshake controlled by ACK line */
      BPP_BUSY_HS,                /* handshake controlled by BSY line */
      BPP_ACK_BUSY_HS,            /*
                                   * handshake controlled by ACK and BSY lines
                                   * read_handshake only!
                                   */
      BPP_XSCAN_HS,               /* xerox scanner mode,
                                   * read_handshake only!
                                   */
      BPP_HSCAN_HS,               /*
                                   * HP scanjet scanner mode
                                   * read_handshake only!
                                   */
      BPP_CLEAR_MEM,              /* write 0's to memory,
                                   * read_handshake only!
                                   */
      BPP_SET_MEM,                /* write 1's to memory,
                                   * read_handshake only!
                                   */
      /* The following handshakes are RESERVED. Do not use. */
```

```
    BPP_VPRINT_HS,             /* valid only in read/write mode */
    BPP_VPLOT_HS               /* valid only in read/write mode */
};
```

The **read_setup_time** field controls the time between dstrb falling edge to bsy rising edge if the **read_handshake** field is set to **BPP_NO_HS** or **BPP_ACK_HS**. It controls the time between dstrb falling edge to ack rising edge if the **read_handshake** field is set to **BPP_ACK_HS** or **BPP_ACK_BUSY_HS**. It controls the time between ack falling edge to dstrb rising edge if the **read_handshake** field is set to **BPP_XSCAN_HS**.

The **read_strobe_width** field controls the time between ack rising edge and ack falling edge if the **read_handshake** field is set to **BPP_NO_HS** or **BPP_ACK_BUSY_HS**. It controls the time between dstrb rising edge to dstrb falling edge if the **read_handshake** field is set to **BPP_XSCAN_HS**.

The values allowed for the **write_handshake** field are duplicates of the definitions for the **read_handshake** field. Note that some of these handshake definitions are only valid in one mode or the other.

The **write_setup_time** field controls the time between data valid to dstrb rising edge for all values of the **write_handshake** field.

The **write_strobe_width** field controls the time between dstrb rising edge and dstrb falling edge if the **write_handshake** field is not set to **BPP_VPRINT_HS** or **BPP_VPLOT_HS**. It controls the minimum time between dstrb rising edge to dstrb falling edge if the **write_handshake** field is set to **BPP_VPRINT_HS** or **BPP_VPLOT_HS**.

Transfer Pins Structure

This structure is defined in **<sys/bpp_io.h>**.

```
struct  bpp_pins {
    u_char   output_reg_pins;      /* pins in P_OR register */
    u_char   input_reg_pins;       /* pins in P_IR register */
};

/* Values for output_reg_pins field */
#define BPP_SLCTIN_PIN  0x01       /* Select in pin */
#define BPP_AFX_PIN     0x02       /* Auto feed pin */
#define BPP_INIT_PIN    0x04       /* Initialize pin */
#define BPP_V1_PIN      0x08       /* reserved pin 1 */
#define BPP_V2_PIN      0x10       /* reserved pin 2 */
#define BPP_V3_PIN      0x20       /* reserved pin 3 */

#define BPP_ERR_PIN     0x01       /* Error pin */
#define BPP_SLCT_PIN    0x02       /* Select pin */
#define BPP_PE_PIN      0x04       /* Paper empty pin */
```

Error Pins Structure

This structure is defined in the include file **<sys/bpp_io.h>**.

```
struct bpp_error_status {
```

```
        char      timeout_occurred;    /* 1 if a timeout occurred */
        char      bus_error;           /* 1 if an SBus bus error */
        u_char    pin_status;          /*
                                        * status of pins which could
                                        * cause an error
                                        */
};

/* Values for pin_status field */
#define BPP_ERR_ERR       0x01       /* Error pin active */
#define BPP_SLCT_ERR      0x02       /* Select pin active */
#define BPP_PE_ERR        0x04       /* Paper empty pin active */
#define BPP_SLCTIN_ERR  0x10         /* Select in pin active*/
#define BPP_BUSY_ERR      0x40       /* Busy pin active */
```

ERRORS

EBADF The device is opened for write-only access and a read is attempted, or the device is opened for read-only access and a write is attempted.

EBUSY The device has been opened and another open is attempted.
An attempt has been made to unload the driver while one of the units is open.

EINVAL A **BPPIOC_SETPARMS ioctl** is attempted with an out of range value in the **bpp_transfer_parms** structure.
A **BPPIOC_SETOUTPINS ioctl** is attempted with an invalid value in the pins structure.
An **ioctl** is attempted with an invalid value in the command argument.
An invalid command argument is received during **modload**(1M) or **modunload**(1M).

EIO The driver encountered an SBus bus error when attempting an access.

A read or write does not complete properly, due to a peripheral error or a transfer timeout.

A **BPPIOC_TESTIO ioctl** call is attempted while a condition exists which would prevent a transfer (such as a peripheral error).

ENXIO The driver has received an open request for a unit for which the attach failed.
The driver has received a read or write request for a unit number greater than the number of units available.
The driver has received a write request for a unit which has an active peripheral error.

FILES

/dev/bpp*n* bi-directional parallel port devices

SEE ALSO

ioctl(2), **read**(2), **write**(2), **sbus**(4)