NAME

gpio_open, **gpio_close** - library to handle GPIO pins

LIBRARY

General-Purpose Input Output (GPIO) library (libgpio, -lgpio)

SYNOPSIS

#include <sys/types.h>
#include <libgpio.h>

gpio_handle_t
gpio_open(*unsigned int unit*);

gpio_handle_t
gpio_open_device(*const char *device*);

void
gpio_close(*gpio_handle_t handle*);

int
gpio_pin_list(*gpio_handle_t handle*, *gpio_config_t **pcfgs*);

int
gpio_pin_config(*gpio_handle_t handle*, *gpio_config_t *cfg*);

int
gpio_pin_set_name(*gpio_handle_t handle*, *gpio_pin_t pin*, *char *name*);

int
gpio_pin_set_flags(*gpio_handle_t handle*, *gpio_config_t *cfg*);

gpio_value_t
gpio_pin_get(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_set(*gpio_handle_t handle*, *gpio_pin_t pin*, *gpio_value_t value*);

int
gpio_pin_toggle(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_low(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_high(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_input(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_output(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_opendrain(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_pushpull(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_tristate(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_pullup(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_pulldown(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_invin(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_invout(*gpio_handle_t handle*, *gpio_pin_t pin*);

int
gpio_pin_pulsate(*gpio_handle_t handle*, *gpio_pin_t pin*);

DESCRIPTION

The **libgpio** library provides an interface to configure GPIO pins. The library operates with a *gpio_handle_t* opaque type which can be created with **gpio_open**() or **gpio_open_device**(). When no more GPIO operations are needed, this handle can be destroyed with **gpio_close**().

To get a list of all available pins, one can call **gpio_pin_list**(). This function takes a pointer to a *gpio_config_t* which is dynamically allocated. This pointer should be freed with free(3) when it is no longer necessary.

The function **gpio_pin_config**() retrieves the current configuration of a pin. The pin number should be passed in via the *g_pin* variable which is part of the *gpio_config_t* structure.

The function **gpio_pin_set_name**() sets the name used to describe a pin.

The function **gpio_pin_set_flags**() configures a pin with the flags passed in by the *gpio_config_t* structure. The pin number should also be passed in through the *g_pin* variable. All other structure members will be ignored by this function. The list of flags can be found in */usr/include/sys/gpio.h*.

The get or set the state of a GPIO pin, the functions **gpio_pin_get**() and **gpio_pin_set**() are available, respectively. To toggle the state, use **gpio_pin_toggle**().

The functions **gpio_pin_low**() and **gpio_pin_high**() are wrappers around **gpio_pin_set**().

The functions **gpio_pin_input**(), **gpio_pin_output**(), **gpio_pin_opendrain**(), **gpio_pin_pushpull**(), **gpio_pin_tristate**(), **gpio_pin_pullup**(), **gpio_pin_pulldown**(), **gpio_pin_invin**(), **gpio_pin_invout**() and **gpio_pin_pulsate**() are wrappers around **gpio_pin_set_flags**().

EXAMPLES

The following example shows how to configure pin 16 as output and then drive it high:

```
#include <sys/types.h>
#include <err.h>
#include <libgpio.h>

gpio_handle_t handle;

handle = gpio_open(0);
if (handle == GPIO_INVALID_HANDLE)
        err(1, "gpio_open failed");
gpio_pin_output(handle, 16);
gpio_pin_high(handle, 16);
gpio_close(handle);
```

The following example shows how to get a configuration of a pin:

```
gpio_config_t cfg;

cfg.g_pin = 32;
gpio_pin_config(handle, &cfg);
```

The structure will contain the name of the pin and its flags.

SEE ALSO

gpiobus(4), gpioctl(8)

HISTORY

The **libgpio** library first appeared in FreeBSD 11.0.

AUTHORS

The **libgpio** library was implemented by Rui Paulo <*rpaulo@FreeBSD.org*>.