My Project

Generated by Doxygen 1.9.4

Chapter 1

libgpiod public API

This is the complete documentation of the public API made available to users of libgpiod.

The public header is logically split into two high-level parts: the simple API and the low-level API. The former allows users to easily interact with the GPIOs in the system without dealing with the low-level data structures and resource control. The latter gives the user much more fine-grained control over the GPIO interface.

The low-level API is further logically split into several parts such as: GPIO chip & line operators, iterators, GPIO events handling etc.

General note on error handling: all routines exported by libgpiod set errno to one of the error values defined in errno.h upon failure. The way of notifying the caller that an error occurred varies between functions, but in general a function that returns an int, returns -1 on error, while a function returning a pointer bails out on error condition by returning a NULL pointer.

Chapter 2

Deprecated List

Member gpiod_ctxless_event_loop (const char ∗device, unsigned int offset, bool active_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_↩ event_handle_cb event_cb, void ∗data) GPIOD_API GPIOD_DEPRECATED

This function suffers from an issue where HW may not allow setting up both rising and falling egde interrupts at the same time.

Member gpiod_ctxless_event_loop_multiple (const char ∗device, const unsigned int ∗offsets, unsigned int num_lines, bool active_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless↩ _event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data) GPIOD_API GPIOD_↩ DEPRECATED

This function suffers from an issue where HW may not allow setting up both rising and falling egde interrupts at the same time.

Member gpiod_line_needs_update (struct gpiod_line ∗line) GPIOD_API GPIOD_DEPRECATED

This mechanism no longer exists in the library and this function does nothing.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Module Documentation

6.1 Common helper macros

Macros

- #define **GPIOD_API** __attribute__((visibility("default")))

  Makes symbol visible.
- #define **GPIOD_UNUSED** __attribute__((unused))

  Marks a function argument or variable as potentially unused.
- #define GPIOD_BIT(nr) (1UL << (nr))

  Shift 1 by given offset.
- #define **GPIOD_DEPRECATED** __attribute__((deprecated))

  Marks a public function as deprecated.

6.1.1 Detailed Description

Commonly used utility macros.

6.1.2 Macro Definition Documentation

6.1.2.1 GPIOD_BIT

```
#define GPIOD_BIT(
                nr ) (1UL << (nr))
```

Shift 1 by given offset.

Parameters

nr	Bit position.

Returns

1 shifted by nr.

6.2 High-level API

Classes

- struct gpiod_ctxless_event_poll_fd

  Helper structure for the ctxless event loop poll callback.

Typedefs

- typedef void(∗ **gpiod_ctxless_set_value_cb**) (void ∗)

  Simple set value callback signature.
- typedef int(∗ gpiod_ctxless_event_handle_cb) (int, unsigned int, const struct timespec ∗, void ∗)

  Simple event callback signature.
- typedef int(∗ gpiod_ctxless_event_poll_cb) (unsigned int, struct gpiod_ctxless_event_poll_fd ∗, const struct timespec ∗, void ∗)

  Simple event poll callback signature.

Enumerations

- enum {
  GPIOD_CTXLESS_FLAG_OPEN_DRAIN = GPIOD_BIT(0) , GPIOD_CTXLESS_FLAG_OPEN_SOURCE = GPIOD_BIT(1) , GPIOD_CTXLESS_FLAG_BIAS_DISABLE = GPIOD_BIT(2) , GPIOD_CTXLESS_FLAG_BIAS_PULL_DOWN = GPIOD_BIT(3) ,
  GPIOD_CTXLESS_FLAG_BIAS_PULL_UP = GPIOD_BIT(4) }

  Miscellaneous GPIO flags.
- enum { GPIOD_CTXLESS_EVENT_RISING_EDGE = 1 , GPIOD_CTXLESS_EVENT_FALLING_EDGE , **GPIOD_CTXLESS_EVENT_BOTH_EDGES** }

  Event types that the ctxless event monitor can wait for.
- enum { GPIOD_CTXLESS_EVENT_CB_TIMEOUT = 1 , GPIOD_CTXLESS_EVENT_CB_RISING_EDGE , GPIOD_CTXLESS_EVENT_CB_FALLING_EDGE }

  Event types that can be passed to the ctxless event callback.
- enum { GPIOD_CTXLESS_EVENT_CB_RET_ERR = -1 , GPIOD_CTXLESS_EVENT_CB_RET_OK = 0 , GPIOD_CTXLESS_EVENT_CB_RET_STOP = 1 }

  Return status values that the ctxless event callback can return.
- enum { GPIOD_CTXLESS_EVENT_POLL_RET_STOP = -2 , GPIOD_CTXLESS_EVENT_POLL_RET_ERR = -1 , GPIOD_CTXLESS_EVENT_POLL_RET_TIMEOUT = 0 }

  Return status values that the ctxless event poll callback can return.

Functions

- int gpiod_ctxless_get_value (const char ∗device, unsigned int offset, bool active_low, const char ∗consumer) GPIOD_API

  Read current value from a single GPIO line.
- int gpiod_ctxless_get_value_ext (const char ∗device, unsigned int offset, bool active_low, const char ∗consumer, int flags) GPIOD_API

  Read current value from a single GPIO line.
- int gpiod_ctxless_get_value_multiple (const char ∗device, const unsigned int ∗offsets, int ∗values, unsigned int num_lines, bool active_low, const char ∗consumer) GPIOD_API

  Read current values from a set of GPIO lines.
- int gpiod_ctxless_get_value_multiple_ext (const char ∗device, const unsigned int ∗offsets, int ∗values, unsigned int num_lines, bool active_low, const char ∗consumer, int flags) GPIOD_API

  Read current values from a set of GPIO lines.
- int gpiod_ctxless_set_value (const char ∗device, unsigned int offset, int value, bool active_low, const char ∗consumer, gpiod_ctxless_set_value_cb cb, void ∗data) GPIOD_API

  Set value of a single GPIO line.
- int gpiod_ctxless_set_value_ext (const char ∗device, unsigned int offset, int value, bool active_low, const char ∗consumer, gpiod_ctxless_set_value_cb cb, void ∗data, int flags) GPIOD_API

  Set value of a single GPIO line.
- int gpiod_ctxless_set_value_multiple (const char ∗device, const unsigned int ∗offsets, const int ∗values, unsigned int num_lines, bool active_low, const char ∗consumer, gpiod_ctxless_set_value_cb cb, void ∗data) GPIOD_API

  Set values of multiple GPIO lines.
- int gpiod_ctxless_set_value_multiple_ext (const char ∗device, const unsigned int ∗offsets, const int ∗values, unsigned int num_lines, bool active_low, const char ∗consumer, gpiod_ctxless_set_value_cb cb, void ∗data, int flags) GPIOD_API

  Set values of multiple GPIO lines.
- int gpiod_ctxless_event_loop (const char ∗device, unsigned int offset, bool active_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data) GPIOD_API GPIOD_DEPRECATED

  Wait for events on a single GPIO line.
- int gpiod_ctxless_event_loop_multiple (const char ∗device, const unsigned int ∗offsets, unsigned int num↩ _lines, bool active_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data) GPIOD_API GPIOD_DEPRECATED

  Wait for events on multiple GPIO lines.
- int gpiod_ctxless_event_monitor (const char ∗device, int event_type, unsigned int offset, bool active↩ _low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data) GPIOD_API

  Wait for events on a single GPIO line.
- int gpiod_ctxless_event_monitor_ext (const char ∗device, int event_type, unsigned int offset, bool active↩ _low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data, int flags) GPIOD_API

  Wait for events on a single GPIO line.
- int gpiod_ctxless_event_monitor_multiple (const char ∗device, int event_type, const unsigned int ∗offsets, unsigned int num_lines, bool active_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data) GPIOD_API

  Wait for events on multiple GPIO lines.
- int gpiod_ctxless_event_monitor_multiple_ext (const char ∗device, int event_type, const unsigned int ∗offsets, unsigned int num_lines, bool active_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data, int flags) GPIOD_API

  Wait for events on multiple GPIO lines.
- int gpiod_ctxless_find_line (const char ∗name, char ∗chipname, size_t chipname_size, unsigned int ∗offset) GPIOD_API

  Determine the chip name and line offset of a line with given name.

6.2.1 Detailed Description

Simple high-level routines for straightforward GPIO manipulation without the need to use the gpiod_∗ structures or to keep track of resources.

6.2.2 Typedef Documentation

6.2.2.1 gpiod_ctxless_event_handle_cb

```
typedef int(∗ gpiod_ctxless_event_handle_cb) (int, unsigned int, const struct timespec ∗, void
∗)
```

Simple event callback signature.

The callback function takes the following arguments: event type (int), GPIO line offset (unsigned int), event timestamp (const struct timespec ∗) and a pointer to user data (void ∗).

This callback is called by the ctxless event loop functions for each GPIO event. If the callback returns GPIOD_CTXLESS_EVENT_CB_RET_ERR, it should also set errno.

6.2.2.2 gpiod_ctxless_event_poll_cb

```
typedef int(∗ gpiod_ctxless_event_poll_cb) (unsigned int, struct gpiod_ctxless_event_poll_fd ∗,
const struct timespec ∗, void ∗)
```

Simple event poll callback signature.

The poll callback function takes the following arguments: number of lines (unsigned int), an array of file descriptors on which input events should be monitored (struct gpiod_ctxless_event_poll_fd ∗), poll timeout (const struct timespec ∗) and a pointer to user data (void ∗).

The callback should poll for input events on the set of descriptors and return an appropriate value that can be interpreted by the event loop routine.

6.2.3 Enumeration Type Documentation

6.2.3.1 anonymous enum

```
anonymous enum
```

Miscellaneous GPIO flags.

Enumerator

GPIOD_CTXLESS_FLAG_OPEN_DRAIN	The line is an open-drain port.
GPIOD_CTXLESS_FLAG_OPEN_SOURCE	The line is an open-source port.
GPIOD_CTXLESS_FLAG_BIAS_DISABLE	The line has neither either pull-up nor pull-down resistor
GPIOD_CTXLESS_FLAG_BIAS_PULL_DOWN	The line has pull-down resistor enabled
GPIOD_CTXLESS_FLAG_BIAS_PULL_UP	The line has pull-up resistor enabled

6.2.3.2 anonymous enum

```
anonymous enum
```

Event types that the ctxless event monitor can wait for.

Enumerator

GPIOD_CTXLESS_EVENT_RISING_EDGE	Wait for rising edge events only. Wait for falling edge events only.
GPIOD_CTXLESS_EVENT_FALLING_EDGE	Wait for both types of events.

6.2.3.3 anonymous enum

```
anonymous enum
```

Event types that can be passed to the ctxless event callback.

Enumerator

GPIOD_CTXLESS_EVENT_CB_TIMEOUT	Waiting for events timed out.
GPIOD_CTXLESS_EVENT_CB_RISING_EDGE	Rising edge event occured.
GPIOD_CTXLESS_EVENT_CB_FALLING_EDGE	Falling edge event occured.

6.2.3.4 anonymous enum

```
anonymous enum
```

Return status values that the ctxless event callback can return.

Enumerator

GPIOD_CTXLESS_EVENT_CB_RET_ERR	Stop processing events and indicate an error.
GPIOD_CTXLESS_EVENT_CB_RET_OK	Continue processing events.
GPIOD_CTXLESS_EVENT_CB_RET_STOP	Stop processing events.

6.2.3.5 anonymous enum

```
anonymous enum
```

Return status values that the ctxless event poll callback can return.

Positive value returned from the polling callback indicates the number of events that occurred on the set of monitored lines.

Enumerator

GPIOD_CTXLESS_EVENT_POLL_RET_STOP	The event loop should stop processing events.
GPIOD_CTXLESS_EVENT_POLL_RET_ERR	Polling error occurred (the polling function should set errno).
GPIOD_CTXLESS_EVENT_POLL_RET_TIMEOUT	Poll timed out.

6.2.4 Function Documentation

6.2.4.1 gpiod_ctxless_event_loop()

```
int gpiod_ctxless_event_loop (
            const char * device,
            unsigned int offset,
            bool active_low,
            const char * consumer,
            const struct timespec * timeout,
            gpiod_ctxless_event_poll_cb poll_cb,
            gpiod_ctxless_event_handle_cb event_cb,
            void * data )
```

Wait for events on a single GPIO line.

Parameters

device	Name, path, number or label of the gpiochip.
offset	GPIO line offset to monitor.
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
timeout	Maximum wait time for each iteration.
poll_cb	Callback function to call when waiting for events.
event_cb	Callback function to call for each line event.
data	User data passed to the callback.

Returns

0 if no errors were encountered, -1 if an error occurred.

Note

The way the ctxless event loop works is described in detail in gpiod_ctxless_event_loop_multiple - this is just a wrapper aound this routine which calls it for a single GPIO line.

Deprecated This function suffers from an issue where HW may not allow setting up both rising and falling egde interrupts at the same time.

6.2.4.2 gpiod_ctxless_event_loop_multiple()

```
int gpiod_ctxless_event_loop_multiple (
            const char * device,
            const unsigned int * offsets,
            unsigned int num_lines,
            bool active_low,
            const char * consumer,
            const struct timespec * timeout,
            gpiod_ctxless_event_poll_cb poll_cb,
            gpiod_ctxless_event_handle_cb event_cb,
            void * data )
```

Wait for events on multiple GPIO lines.

Parameters

device	Name, path, number or label of the gpiochip.
offsets	Array of GPIO line offsets to monitor.
num_lines	Number of lines to monitor.
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
timeout	Maximum wait time for each iteration.
poll_cb	Callback function to call when waiting for events. Can be NULL.
event_cb	Callback function to call on event occurrence.
data	User data passed to the callback.

Returns

0 no errors were encountered, -1 if an error occurred.

Note

The poll callback can be NULL in which case the routine will fall back to a basic, ppoll() based callback.

Deprecated This function suffers from an issue where HW may not allow setting up both rising and falling egde interrupts at the same time.

Internally this routine opens the GPIO chip, requests the set of lines for both-edges events and calls the polling callback in a loop. The role of the polling callback is to detect input events on a set of file descriptors and notify the caller about the fds ready for reading.

The ctxless event loop then reads each queued event from marked descriptors and calls the event callback. Both callbacks can stop the loop at any point.

The poll_cb argument can be NULL in which case the function falls back to a default, ppoll() based callback.

6.2.4.3 gpiod_ctxless_event_monitor()

```
int gpiod_ctxless_event_monitor (
            const char * device,
            int event_type,
            unsigned int offset,
            bool active_low,
            const char * consumer,
            const struct timespec * timeout,
            gpiod_ctxless_event_poll_cb poll_cb,
            gpiod_ctxless_event_handle_cb event_cb,
            void * data )
```

Wait for events on a single GPIO line.

Parameters

device	Name, path, number or label of the gpiochip.
event_type	Type of events to listen for.
offset	GPIO line offset to monitor.
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
timeout	Maximum wait time for each iteration.
poll_cb	Callback function to call when waiting for events.
event_cb	Callback function to call for each line event.
data	User data passed to the callback.

Returns

0 if no errors were encountered, -1 if an error occurred.

Note

The way the ctxless event loop works is described in detail in gpiod_ctxless_event_monitor_multiple - this is just a wrapper aound this routine which calls it for a single GPIO line.

6.2.4.4 gpiod_ctxless_event_monitor_ext()

```
int gpiod_ctxless_event_monitor_ext (
            const char * device,
            int event_type,
            unsigned int offset,
            bool active_low,
            const char * consumer,
            const struct timespec * timeout,
            gpiod_ctxless_event_poll_cb poll_cb,
            gpiod_ctxless_event_handle_cb event_cb,
            void * data,
            int flags )
```

Wait for events on a single GPIO line.

Parameters

device	Name, path, number or label of the gpiochip.
event_type	Type of events to listen for.
offset	GPIO line offset to monitor.
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
timeout	Maximum wait time for each iteration.
poll_cb	Callback function to call when waiting for events.
event_cb	Callback function to call for each line event.
data	User data passed to the callback.
flags	The flags for the line.

Returns

0 if no errors were encountered, -1 if an error occurred.

Note

The way the ctxless event loop works is described in detail in gpiod_ctxless_event_monitor_multiple - this is just a wrapper aound this routine which calls it for a single GPIO line.

6.2.4.5 gpiod_ctxless_event_monitor_multiple()

```
int gpiod_ctxless_event_monitor_multiple (
            const char * device,
            int event_type,
            const unsigned int * offsets,
            unsigned int num_lines,
            bool active_low,
            const char * consumer,
            const struct timespec * timeout,
            gpiod_ctxless_event_poll_cb poll_cb,
            gpiod_ctxless_event_handle_cb event_cb,
            void * data )
```

Wait for events on multiple GPIO lines.

Parameters

device	Name, path, number or label of the gpiochip.
event_type	Type of events to listen for.
offsets	Array of GPIO line offsets to monitor.
num_lines	Number of lines to monitor.
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
timeout	Maximum wait time for each iteration.
poll_cb	Callback function to call when waiting for events. Can be NULL.
event_cb	Callback function to call on event occurrence.
data	User data passed to the callback.

Returns

0 no errors were encountered, -1 if an error occurred.

Note

The poll callback can be NULL in which case the routine will fall back to a basic, ppoll() based callback.

Internally this routine opens the GPIO chip, requests the set of lines for the type of events specified in the event_type parameter and calls the polling callback in a loop. The role of the polling callback is to detect input events on a set of file descriptors and notify the caller about the fds ready for reading.

The ctxless event loop then reads each queued event from marked descriptors and calls the event callback. Both callbacks can stop the loop at any point.

The poll_cb argument can be NULL in which case the function falls back to a default, ppoll() based callback.

6.2.4.6 gpiod_ctxless_event_monitor_multiple_ext()

```
int gpiod_ctxless_event_monitor_multiple_ext (
            const char * device,
            int event_type,
            const unsigned int * offsets,
            unsigned int num_lines,
            bool active_low,
            const char * consumer,
            const struct timespec * timeout,
            gpiod_ctxless_event_poll_cb poll_cb,
            gpiod_ctxless_event_handle_cb event_cb,
            void * data,
            int flags )
```

Wait for events on multiple GPIO lines.

Parameters

device	Name, path, number or label of the gpiochip.
event_type	Type of events to listen for.
offsets	Array of GPIO line offsets to monitor.

Parameters

num_lines	Number of lines to monitor.
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
timeout	Maximum wait time for each iteration.
poll_cb	Callback function to call when waiting for events. Can be NULL.
event_cb	Callback function to call on event occurrence.
data	User data passed to the callback.
flags	The flags for the lines.

Returns

0 no errors were encountered, -1 if an error occurred.

Note

The poll callback can be NULL in which case the routine will fall back to a basic, ppoll() based callback.

Internally this routine opens the GPIO chip, requests the set of lines for the type of events specified in the event_type parameter and calls the polling callback in a loop. The role of the polling callback is to detect input events on a set of file descriptors and notify the caller about the fds ready for reading.

The ctxless event loop then reads each queued event from marked descriptors and calls the event callback. Both callbacks can stop the loop at any point.

The poll_cb argument can be NULL in which case the function falls back to a default, ppoll() based callback.

6.2.4.7 gpiod_ctxless_find_line()

```
int gpiod_ctxless_find_line (
            const char * name,
            char * chipname,
            size_t chipname_size,
            unsigned int * offset )
```

Determine the chip name and line offset of a line with given name.

Parameters

name	The name of the GPIO line to lookup.
chipname	Buffer in which the name of the GPIO chip will be stored.
chipname_size	Size of the chip name buffer.
offset	Pointer to an integer in which the line offset will be stored.

Returns

-1 on error, 0 if the line with given name doesn't exist and 1 if the line was found. In the first two cases the contents of chipname and offset remain unchanged.

Note

The chip name is truncated if the buffer can't hold its entire size.

Attention

GPIO line names are not unique in the linux kernel, neither globally nor within a single chip. This function finds the first line with given name.

6.2.4.8 gpiod_ctxless_get_value()

```
int gpiod_ctxless_get_value (
             const char * device,
             unsigned int offset,
             bool active_low,
             const char * consumer )
```

Read current value from a single GPIO line.

Parameters

device	Name, path, number or label of the gpiochip.
offset	Offset of the GPIO line.
active_low	The active state of this line - true if low.
consumer	Name of the consumer.

Returns

0 or 1 (GPIO value) if the operation succeeds, -1 on error.

6.2.4.9 gpiod_ctxless_get_value_ext()

```
int gpiod_ctxless_get_value_ext (
             const char * device,
             unsigned int offset,
             bool active_low,
             const char * consumer,
             int flags )
```

Read current value from a single GPIO line.

Parameters

device	Name, path, number or label of the gpiochip.
offset	Offset of the GPIO line.
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
flags	The flags for the line.

Returns

0 or 1 (GPIO value) if the operation succeeds, -1 on error.

6.2.4.10 gpiod_ctxless_get_value_multiple()

```
int gpiod_ctxless_get_value_multiple (
            const char * device,
            const unsigned int * offsets,
            int * values,
            unsigned int num_lines,
            bool active_low,
            const char * consumer )
```

Read current values from a set of GPIO lines.

Parameters

device	Name, path, number or label of the gpiochip.
offsets	Array of offsets of lines whose values should be read.
values	Buffer in which the values will be stored.
num_lines	Number of lines, must be $>$ 0.
active_low	The active state of the lines - true if low.
consumer	Name of the consumer.

Returns

0 if the operation succeeds, -1 on error.

6.2.4.11 gpiod_ctxless_get_value_multiple_ext()

```
int gpiod_ctxless_get_value_multiple_ext (
            const char * device,
            const unsigned int * offsets,
            int * values,
            unsigned int num_lines,
            bool active_low,
            const char * consumer,
            int flags )
```

Read current values from a set of GPIO lines.

Parameters

device	Name, path, number or label of the gpiochip.
offsets	Array of offsets of lines whose values should be read.
values	Buffer in which the values will be stored.
num_lines	Number of lines, must be $>$ 0.
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
flags	The flags for the lines.

Returns

0 if the operation succeeds, -1 on error.

6.2.4.12 gpiod_ctxless_set_value()

```
int gpiod_ctxless_set_value (
             const char * device,
             unsigned int offset,
             int value,
             bool active_low,
             const char * consumer,
             gpiod_ctxless_set_value_cb cb,
             void * data )
```

Set value of a single GPIO line.

Parameters

device	Name, path, number or label of the gpiochip.
offset	The offset of the GPIO line.
value	New value (0 or 1).
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
cb	Optional callback function that will be called right after setting the value. Users can use this, for example, to pause the execution after toggling a GPIO.
data	Optional user data that will be passed to the callback function.

Returns

0 if the operation succeeds, -1 on error.

6.2.4.13 gpiod_ctxless_set_value_ext()

```
int gpiod_ctxless_set_value_ext (
             const char * device,
             unsigned int offset,
             int value,
             bool active_low,
             const char * consumer,
             gpiod_ctxless_set_value_cb cb,
             void * data,
             int flags )
```

Set value of a single GPIO line.

Parameters

device	Name, path, number or label of the gpiochip.
offset	The offset of the GPIO line.
value	New value (0 or 1).
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
cb	Optional callback function that will be called right after setting the value. Users can use this, for example, to pause the execution after toggling a GPIO.
data	Optional user data that will be passed to the callback function.
flags	The flags for the line.

Returns

0 if the operation succeeds, -1 on error.

6.2.4.14 gpiod_ctxless_set_value_multiple()

```
int gpiod_ctxless_set_value_multiple (
             const char * device,
             const unsigned int * offsets,
             const int * values,
             unsigned int num_lines,
             bool active_low,
             const char * consumer,
             gpiod_ctxless_set_value_cb cb,
             void * data )
```

Set values of multiple GPIO lines.

Parameters

device	Name, path, number or label of the gpiochip.
offsets	Array of offsets of lines the values of which should be set.
values	Array of integers containing new values.
num_lines	Number of lines, must be $>$ 0.
active_low	The active state of the lines - true if low.
consumer	Name of the consumer.
cb	Optional callback function that will be called right after setting all values. Works the same as in gpiod_ctxless_set_value.
data	Optional user data that will be passed to the callback function.

Returns

0 if the operation succeeds, -1 on error.

6.2.4.15 gpiod_ctxless_set_value_multiple_ext()

```
int gpiod_ctxless_set_value_multiple_ext (
            const char * device,
            const unsigned int * offsets,
            const int * values,
            unsigned int num_lines,
            bool active_low,
            const char * consumer,
            gpiod_ctxless_set_value_cb cb,
            void * data,
            int flags )
```

Set values of multiple GPIO lines.

Parameters

device	Name, path, number or label of the gpiochip.
offsets	Array of offsets of lines the values of which should be set.
values	Array of integers containing new values.
num_lines	Number of lines, must be $>$ 0.
active_low	The active state of this line - true if low.
consumer	Name of the consumer.
cb	Optional callback function that will be called right after setting all values. Works the same as in gpiod_ctxless_set_value.
data	Optional user data that will be passed to the callback function.
flags	The flags for the lines.

Returns

0 if the operation succeeds, -1 on error.

6.3 GPIO chip operations

Functions

- struct gpiod_chip ∗ gpiod_chip_open (const char ∗path) GPIOD_API

  Open a gpiochip by path.
- struct gpiod_chip ∗ gpiod_chip_open_by_name (const char ∗name) GPIOD_API

  Open a gpiochip by name.
- struct gpiod_chip ∗ gpiod_chip_open_by_number (unsigned int num) GPIOD_API

  Open a gpiochip by number.
- struct gpiod_chip ∗ gpiod_chip_open_by_label (const char ∗label) GPIOD_API

  Open a gpiochip by label.
- struct gpiod_chip ∗ gpiod_chip_open_lookup (const char ∗descr) GPIOD_API

  Open a gpiochip based on the best guess what the path is.
- void gpiod_chip_close (struct gpiod_chip ∗chip) GPIOD_API

  Close a GPIO chip handle and release all allocated resources.
- const char ∗ gpiod_chip_name (struct gpiod_chip ∗chip) GPIOD_API

  Get the GPIO chip name as represented in the kernel.

- const char ∗ gpiod_chip_label (struct gpiod_chip ∗chip) GPIOD_API

  Get the GPIO chip label as represented in the kernel.
- unsigned int gpiod_chip_num_lines (struct gpiod_chip ∗chip) GPIOD_API

  Get the number of GPIO lines exposed by this chip.
- struct gpiod_line ∗ gpiod_chip_get_line (struct gpiod_chip ∗chip, unsigned int offset) GPIOD_API

  Get the handle to the GPIO line at given offset.
- int gpiod_chip_get_lines (struct gpiod_chip ∗chip, unsigned int ∗offsets, unsigned int num_offsets, struct gpiod_line_bulk ∗bulk) GPIOD_API

  Retrieve a set of lines and store them in a line bulk object.
- int gpiod_chip_get_all_lines (struct gpiod_chip ∗chip, struct gpiod_line_bulk ∗bulk) GPIOD_API

  Retrieve all lines exposed by a chip and store them in a bulk object.
- struct gpiod_line ∗ gpiod_chip_find_line (struct gpiod_chip ∗chip, const char ∗name) GPIOD_API

  Find a GPIO line by name among lines associated with given GPIO chip.
- int gpiod_chip_find_lines (struct gpiod_chip ∗chip, const char ∗∗names, struct gpiod_line_bulk ∗bulk) GPIOD_API

  Find a set of GPIO lines by names among lines exposed by this chip.

6.3.1 Detailed Description

Functions and data structures dealing with GPIO chips.

6.3.2 Function Documentation

6.3.2.1 gpiod_chip_close()

```
void gpiod_chip_close (
            struct gpiod_chip ∗ chip )
```

Close a GPIO chip handle and release all allocated resources.

Parameters

chip	The GPIO chip object.

6.3.2.2 gpiod_chip_find_line()

```
struct gpiod_line ∗ gpiod_chip_find_line (
            struct gpiod_chip ∗ chip,
            const char ∗ name )
```

Find a GPIO line by name among lines associated with given GPIO chip.

Parameters

chip	The GPIO chip object.
name	The name of the GPIO line.

Returns

Pointer to the GPIO line handle or NULL if the line could not be found or an error occurred.

Note

In case a line with given name is not associated with given chip, the function sets errno to ENOENT.

Attention

GPIO line names are not unique in the linux kernel, neither globally nor within a single chip. This function finds the first line with given name.

6.3.2.3 gpiod_chip_find_lines()

```
int gpiod_chip_find_lines (
            struct gpiod_chip * chip,
            const char ** names,
            struct gpiod_line_bulk * bulk )
```

Find a set of GPIO lines by names among lines exposed by this chip.

Parameters

chip	The GPIO chip object.
names	Array of pointers to C-strings containing the names of the lines to lookup. Must end with a NULL-pointer.
bulk	Line bulk object in which the located lines will be stored.

Returns

0 if all lines were located, -1 on error.

Note

If at least one line from the list could not be found among the lines exposed by this chip, the function sets errno to ENOENT.

Attention

GPIO line names are not unique in the linux kernel, neither globally nor within a single chip. This function finds the first line with given name.

6.3.2.4 gpiod_chip_get_all_lines()

```
int gpiod_chip_get_all_lines (
            struct gpiod_chip * chip,
            struct gpiod_line_bulk * bulk )
```

Retrieve all lines exposed by a chip and store them in a bulk object.

Parameters

chip	The GPIO chip object.
bulk	Line bulk object in which to store the line handles.

Returns

0 on success, -1 on error.

6.3.2.5 gpiod_chip_get_line()

```
struct gpiod_line * gpiod_chip_get_line (
            struct gpiod_chip * chip,
            unsigned int offset )
```

Get the handle to the GPIO line at given offset.

Parameters

chip	The GPIO chip object.
offset	The offset of the GPIO line.

Returns

Pointer to the GPIO line handle or NULL if an error occured.

6.3.2.6 gpiod_chip_get_lines()

```
int gpiod_chip_get_lines (
            struct gpiod_chip * chip,
            unsigned int * offsets,
            unsigned int num_offsets,
            struct gpiod_line_bulk * bulk )
```

Retrieve a set of lines and store them in a line bulk object.

Parameters

chip	The GPIO chip object.
offsets	Array of offsets of lines to retrieve.
num_offsets	Number of lines to retrieve.
bulk	Line bulk object in which to store the line handles.

Returns

0 on success, -1 on error.

6.3.2.7 gpiod_chip_label()

```
const char * gpiod_chip_label (
             struct gpiod_chip * chip )
```

Get the GPIO chip label as represented in the kernel.

Parameters

chip	The GPIO chip object.

Returns

Pointer to a human-readable string containing the chip label.

6.3.2.8 gpiod_chip_name()

```
const char * gpiod_chip_name (
             struct gpiod_chip * chip )
```

Get the GPIO chip name as represented in the kernel.

Parameters

chip	The GPIO chip object.

Returns

Pointer to a human-readable string containing the chip name.

6.3.2.9 gpiod_chip_num_lines()

```
unsigned int gpiod_chip_num_lines (
             struct gpiod_chip * chip )
```

Get the number of GPIO lines exposed by this chip.

Parameters

chip	The GPIO chip object.

Returns

Number of GPIO lines.

6.3.2.10 gpiod_chip_open()

```
struct gpiod_chip * gpiod_chip_open (
             const char * path )
```

Open a gpiochip by path.

Parameters

path	Path to the gpiochip device file.

Returns

GPIO chip handle or NULL if an error occurred.

6.3.2.11 gpiod_chip_open_by_label()

```
struct gpiod_chip * gpiod_chip_open_by_label (
             const char * label )
```

Open a gpiochip by label.

Parameters

label	Label of the gpiochip to open.

Returns

GPIO chip handle or NULL if the chip with given label was not found or an error occured.

Note

If the chip cannot be found but no other error occurred, errno is set to ENOENT.

6.3.2.12 gpiod_chip_open_by_name()

```
struct gpiod_chip * gpiod_chip_open_by_name (
             const char * name )
```

Open a gpiochip by name.

Parameters

name	Name of the gpiochip to open.

Returns

GPIO chip handle or NULL if an error occurred.

This routine appends name to '/dev/' to create the path.

6.3.2.13 gpiod_chip_open_by_number()

```
struct gpiod_chip * gpiod_chip_open_by_number (
             unsigned int num )
```

Open a gpiochip by number.

Parameters

num	Number of the gpiochip.

Returns

GPIO chip handle or NULL if an error occurred.

This routine appends num to '/dev/gpiochip' to create the path.

6.3.2.14 gpiod_chip_open_lookup()

```
struct gpiod_chip * gpiod_chip_open_lookup (
             const char * descr )
```

Open a gpiochip based on the best guess what the path is.

Parameters

descr	String describing the gpiochip.

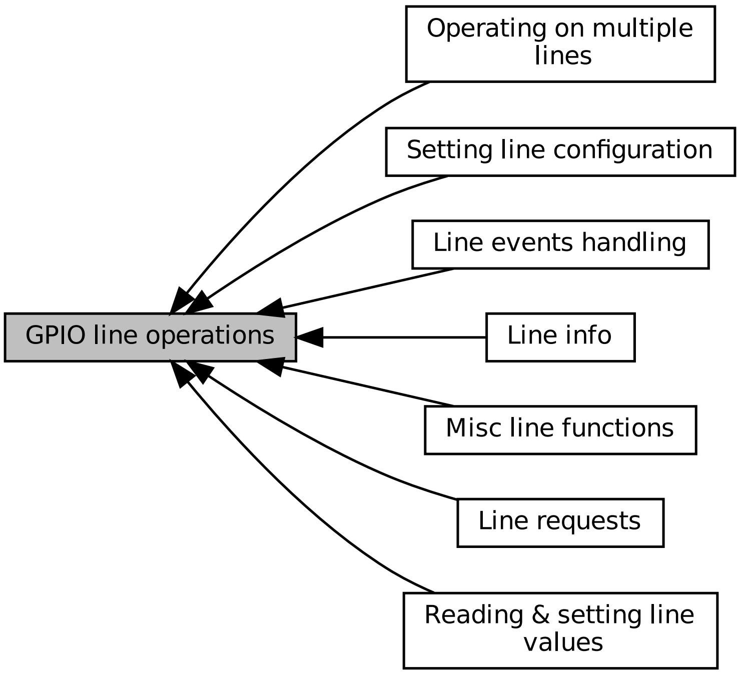
Returns

GPIO chip handle or NULL if an error occurred.

This routine tries to figure out whether the user passed it the path to the GPIO chip, its name, label or number as a string. Then it tries to open it using one of the gpiod_chip_open** variants.

6.4 GPIO line operations

Collaboration diagram for GPIO line operations:


Operating on multiple lines
Setting line configuration
Line events handling
GPIO line operations
Line info
Misc line functions
Line requests
Reading & setting line values

Modules

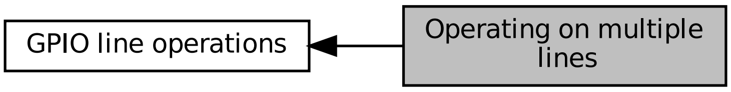
- Operating on multiple lines
- Line info
- Line requests
- Reading & setting line values
- Setting line configuration
- Line events handling
- Misc line functions

6.4.1 Detailed Description

Functions and data structures dealing with GPIO lines.

6.5 Operating on multiple lines

Collaboration diagram for Operating on multiple lines:



Classes

- struct gpiod_line_bulk

    Helper structure for storing a set of GPIO line objects.

Macros

- #define **GPIOD_LINE_BULK_MAX_LINES** 64

    Maximum number of GPIO lines that can be requested at once.
- #define GPIOD_LINE_BULK_INITIALIZER { { NULL }, 0 }

    Static initializer for GPIO bulk objects.
- #define gpiod_line_bulk_foreach_line(bulk, line, lineptr)

    Iterate over all line handles held by a line bulk object.
- #define gpiod_line_bulk_foreach_line_off(bulk, line, offset)

    Iterate over all line handles held by a line bulk object (integer counter variant).

6.5.1 Detailed Description

Convenience data structures and helper functions for storing and operating on multiple lines at once.

6.5.2 Macro Definition Documentation

6.5.2.1 gpiod_line_bulk_foreach_line

```
#define gpiod_line_bulk_foreach_line(
                bulk,
                line,
                lineptr )
```

Value:
```
    for ((lineptr) = (bulk)->lines, (line) = *(lineptr);          \
         (lineptr) <= (bulk)->lines + ((bulk)->num_lines - 1);   \
         (lineptr)++, (line) = *(lineptr))
```

Iterate over all line handles held by a line bulk object.

Parameters

bulk	Line bulk object.
line	GPIO line handle. On each iteration, the subsequent line handle is assigned to this pointer.
lineptr	Pointer to a GPIO line handle used to store the loop state.

6.5.2.2 gpiod_line_bulk_foreach_line_off

```
#define gpiod_line_bulk_foreach_line_off(
              bulk,
              line,
              offset )
```

Value:

```
    for ((offset) = 0, (line) = (bulk)->lines[0];           \
         (offset) < (bulk)->num_lines;               \
         (offset)++, (line) = (bulk)->lines[(offset)])
```

Iterate over all line handles held by a line bulk object (integer counter variant).

Parameters

bulk	Line bulk object.
line	GPIO line handle. On each iteration, the subsequent line handle is assigned to this pointer.
offset	An integer variable used to store the loop state.

This is a variant of gpiod_line_bulk_foreach_line which uses an integer variable (either signed or unsigned) to store the loop state. This offset variable is guaranteed to correspond to the offset of the current line in the bulk->lines array.

6.5.2.3 GPIOD_LINE_BULK_INITIALIZER

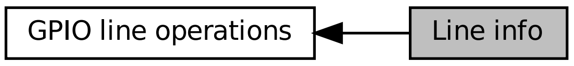
```
#define GPIOD_LINE_BULK_INITIALIZER { { NULL }, 0 }
```

Static initializer for GPIO bulk objects.

This macro simply sets the internally held number of lines to 0.

6.6 Line info

Collaboration diagram for Line info:



Enumerations

- enum { GPIOD_LINE_DIRECTION_INPUT = 1 , GPIOD_LINE_DIRECTION_OUTPUT }

  Possible direction settings.
- enum { GPIOD_LINE_ACTIVE_STATE_HIGH = 1 , GPIOD_LINE_ACTIVE_STATE_LOW }

  Possible active state settings.
- enum { GPIOD_LINE_BIAS_AS_IS = 1 , GPIOD_LINE_BIAS_DISABLE , GPIOD_LINE_BIAS_PULL_UP , GPIOD_LINE_BIAS_PULL_DOWN }

  Possible internal bias settings.

Functions

- unsigned int gpiod_line_offset (struct gpiod_line *line) GPIOD_API

  Read the GPIO line offset.
- const char * gpiod_line_name (struct gpiod_line *line) GPIOD_API

  Read the GPIO line name.
- const char * gpiod_line_consumer (struct gpiod_line *line) GPIOD_API

  Read the GPIO line consumer name.
- int gpiod_line_direction (struct gpiod_line *line) GPIOD_API

  Read the GPIO line direction setting.
- int gpiod_line_active_state (struct gpiod_line *line) GPIOD_API

  Read the GPIO line active state setting.
- int gpiod_line_bias (struct gpiod_line *line) GPIOD_API

  Read the GPIO line bias setting.
- bool gpiod_line_is_used (struct gpiod_line *line) GPIOD_API

  Check if the line is currently in use.
- bool gpiod_line_is_open_drain (struct gpiod_line *line) GPIOD_API

  Check if the line is an open-drain GPIO.
- bool gpiod_line_is_open_source (struct gpiod_line *line) GPIOD_API

  Check if the line is an open-source GPIO.
- int gpiod_line_update (struct gpiod_line *line) GPIOD_API

  Re-read the line info.
- bool gpiod_line_needs_update (struct gpiod_line *line) GPIOD_API GPIOD_DEPRECATED

  Check if the line info needs to be updated.

6.6.1 Detailed Description

Definitions and functions for retrieving kernel information about both requested and free lines.

6.6.2 Enumeration Type Documentation

6.6.2.1 anonymous enum

`anonymous enum`

Possible direction settings.

Enumerator

GPIOD_LINE_DIRECTION_INPUT	Direction is input - we're reading the state of a GPIO line.
GPIOD_LINE_DIRECTION_OUTPUT	Direction is output - we're driving the GPIO line.

6.6.2.2 anonymous enum

```
anonymous enum
```

Possible active state settings.

Enumerator

GPIOD_LINE_ACTIVE_STATE_HIGH	The active state of a GPIO is active-high.
GPIOD_LINE_ACTIVE_STATE_LOW	The active state of a GPIO is active-low.

6.6.2.3 anonymous enum

```
anonymous enum
```

Possible internal bias settings.

Enumerator

GPIOD_LINE_BIAS_AS_IS	The internal bias state is unknown.
GPIOD_LINE_BIAS_DISABLE	The internal bias is disabled.
GPIOD_LINE_BIAS_PULL_UP	The internal pull-up bias is enabled.
GPIOD_LINE_BIAS_PULL_DOWN	The internal pull-down bias is enabled.

6.6.3 Function Documentation

6.6.3.1 gpiod_line_active_state()

```
int gpiod_line_active_state (
            struct gpiod_line * line )
```

Read the GPIO line active state setting.

Parameters

line	GPIO line object.

Returns

Returns GPIOD_LINE_ACTIVE_STATE_HIGH or GPIOD_LINE_ACTIVE_STATE_LOW.

6.6.3.2 gpiod_line_bias()

```
int gpiod_line_bias (
            struct gpiod_line * line )
```

Read the GPIO line bias setting.

Parameters

line	GPIO line object.

Returns

Returns GPIOD_LINE_BIAS_PULL_UP, GPIOD_LINE_BIAS_PULL_DOWN, GPIOD_LINE_BIAS_DISABLE or GPIOD_LINE_BIAS_AS_IS.

6.6.3.3 gpiod_line_consumer()

```
const char * gpiod_line_consumer (
            struct gpiod_line * line )
```

Read the GPIO line consumer name.

Parameters

line	GPIO line object.

Returns

Name of the GPIO consumer name as it is represented in the kernel. This routine returns a pointer to a null-terminated string or NULL if the line is not used.

6.6.3.4 gpiod_line_direction()

```
int gpiod_line_direction (
             struct gpiod_line * line )
```

Read the GPIO line direction setting.

Parameters

line	GPIO line object.

Returns

Returns GPIOD_LINE_DIRECTION_INPUT or GPIOD_LINE_DIRECTION_OUTPUT.

6.6.3.5 gpiod_line_is_open_drain()

```
bool gpiod_line_is_open_drain (
             struct gpiod_line * line )
```

Check if the line is an open-drain GPIO.

Parameters

line	GPIO line object.

Returns

True if the line is an open-drain GPIO, false otherwise.

6.6.3.6 gpiod_line_is_open_source()

```
bool gpiod_line_is_open_source (
             struct gpiod_line * line )
```

Check if the line is an open-source GPIO.

Parameters

line	GPIO line object.

Returns

True if the line is an open-source GPIO, false otherwise.

6.6.3.7 gpiod_line_is_used()

```
bool gpiod_line_is_used (
            struct gpiod_line * line )
```

Check if the line is currently in use.

Parameters

line	GPIO line object.

Returns

True if the line is in use, false otherwise.

The user space can't know exactly why a line is busy. It may have been requested by another process or hogged by the kernel. It only matters that the line is used and we can't request it.

6.6.3.8 gpiod_line_name()

```
const char * gpiod_line_name (
            struct gpiod_line * line )
```

Read the GPIO line name.

Parameters

line	GPIO line object.

Returns

Name of the GPIO line as it is represented in the kernel. This routine returns a pointer to a null-terminated string or NULL if the line is unnamed.

6.6.3.9 gpiod_line_needs_update()

```
bool gpiod_line_needs_update (
            struct gpiod_line * line )
```

Check if the line info needs to be updated.

Parameters

line	GPIO line object.

Returns

Always returns false.

Deprecated This mechanism no longer exists in the library and this function does nothing.

6.6.3.10 gpiod_line_offset()

```
unsigned int gpiod_line_offset (
             struct gpiod_line * line )
```

Read the GPIO line offset.

Parameters

line	GPIO line object.

Returns

Line offset.

6.6.3.11 gpiod_line_update()

```
int gpiod_line_update (
             struct gpiod_line * line )
```

Re-read the line info.

Parameters

line	GPIO line object.

Returns

0 if the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

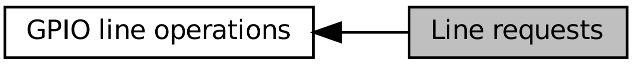
The line info is initially retrieved from the kernel by gpiod_chip_get_line() and is later re-read after every successful request. Users can use this function to manually re-read the line info when needed.

We currently have no mechanism provided by the kernel for keeping the line info synchronized and for the sake of speed and simplicity of this low-level library we don't want to re-read the line info automatically everytime a property is retrieved. Any daemon using this library must track the state of lines on its own and call this routine if needed.

The state of requested lines is kept synchronized (or rather cannot be changed by external agents while the ownership of the line is taken) so there's no need to call this function in that case.

6.7 Line requests

Collaboration diagram for Line requests:



Classes

- struct gpiod_line_request_config

  Structure holding configuration of a line request.

Enumerations

- enum {
  GPIOD_LINE_REQUEST_DIRECTION_AS_IS = 1 , GPIOD_LINE_REQUEST_DIRECTION_INPUT ,
  GPIOD_LINE_REQUEST_DIRECTION_OUTPUT , GPIOD_LINE_REQUEST_EVENT_FALLING_EDGE
  ,
  GPIOD_LINE_REQUEST_EVENT_RISING_EDGE , GPIOD_LINE_REQUEST_EVENT_BOTH_EDGES }

  Available types of requests.
- enum {
  GPIOD_LINE_REQUEST_FLAG_OPEN_DRAIN = GPIOD_BIT(0) , GPIOD_LINE_REQUEST_FLAG_OPEN_SOURCE
  = GPIOD_BIT(1) , GPIOD_LINE_REQUEST_FLAG_ACTIVE_LOW = GPIOD_BIT(2) , GPIOD_LINE_REQUEST_FLAG_BIAS_
  = GPIOD_BIT(3) ,
  GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_DOWN = GPIOD_BIT(4) , GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_UP
  = GPIOD_BIT(5) }

  Miscellaneous GPIO request flags.

Functions

- int gpiod_line_request (struct gpiod_line ∗line, const struct gpiod_line_request_config ∗config, int default_val)
  GPIOD_API

  Reserve a single line.
- int gpiod_line_request_input (struct gpiod_line ∗line, const char ∗consumer) GPIOD_API

  Reserve a single line, set the direction to input.
- int gpiod_line_request_output (struct gpiod_line ∗line, const char ∗consumer, int default_val) GPIOD_API

  Reserve a single line, set the direction to output.
- int gpiod_line_request_rising_edge_events (struct gpiod_line ∗line, const char ∗consumer) GPIOD_API

  Request rising edge event notifications on a single line.
- int gpiod_line_request_falling_edge_events (struct gpiod_line ∗line, const char ∗consumer) GPIOD_API

  Request falling edge event notifications on a single line.
- int gpiod_line_request_both_edges_events (struct gpiod_line ∗line, const char ∗consumer) GPIOD_API

  Request all event type notifications on a single line.

- int gpiod_line_request_input_flags (struct gpiod_line ∗line, const char ∗consumer, int flags) GPIOD_API

  Reserve a single line, set the direction to input.
- int gpiod_line_request_output_flags (struct gpiod_line ∗line, const char ∗consumer, int flags, int default_val) GPIOD_API

  Reserve a single line, set the direction to output.
- int gpiod_line_request_rising_edge_events_flags (struct gpiod_line ∗line, const char ∗consumer, int flags) GPIOD_API

  Request rising edge event notifications on a single line.
- int gpiod_line_request_falling_edge_events_flags (struct gpiod_line ∗line, const char ∗consumer, int flags) GPIOD_API

  Request falling edge event notifications on a single line.
- int gpiod_line_request_both_edges_events_flags (struct gpiod_line ∗line, const char ∗consumer, int flags) GPIOD_API

  Request all event type notifications on a single line.
- int gpiod_line_request_bulk (struct gpiod_line_bulk ∗bulk, const struct gpiod_line_request_config ∗config, const int ∗default_vals) GPIOD_API

  Reserve a set of GPIO lines.
- int gpiod_line_request_bulk_input (struct gpiod_line_bulk ∗bulk, const char ∗consumer) GPIOD_API

  Reserve a set of GPIO lines, set the direction to input.
- int gpiod_line_request_bulk_output (struct gpiod_line_bulk ∗bulk, const char ∗consumer, const int ∗default←_vals) GPIOD_API

  Reserve a set of GPIO lines, set the direction to output.
- int gpiod_line_request_bulk_rising_edge_events (struct gpiod_line_bulk ∗bulk, const char ∗consumer) GPIOD_API

  Request rising edge event notifications on a set of lines.
- int gpiod_line_request_bulk_falling_edge_events (struct gpiod_line_bulk ∗bulk, const char ∗consumer) GPIOD_API

  Request falling edge event notifications on a set of lines.
- int gpiod_line_request_bulk_both_edges_events (struct gpiod_line_bulk ∗bulk, const char ∗consumer) GPIOD_API

  Request all event type notifications on a set of lines.
- int gpiod_line_request_bulk_input_flags (struct gpiod_line_bulk ∗bulk, const char ∗consumer, int flags) GPIOD_API

  Reserve a set of GPIO lines, set the direction to input.
- int gpiod_line_request_bulk_output_flags (struct gpiod_line_bulk ∗bulk, const char ∗consumer, int flags, const int ∗default_vals) GPIOD_API

  Reserve a set of GPIO lines, set the direction to output.
- int gpiod_line_request_bulk_rising_edge_events_flags (struct gpiod_line_bulk ∗bulk, const char ∗consumer, int flags) GPIOD_API

  Request rising edge event notifications on a set of lines.
- int gpiod_line_request_bulk_falling_edge_events_flags (struct gpiod_line_bulk ∗bulk, const char ∗consumer, int flags) GPIOD_API

  Request falling edge event notifications on a set of lines.
- int gpiod_line_request_bulk_both_edges_events_flags (struct gpiod_line_bulk ∗bulk, const char ∗consumer, int flags) GPIOD_API

  Request all event type notifications on a set of lines.
- void gpiod_line_release (struct gpiod_line ∗line) GPIOD_API

  Release a previously reserved line.
- void gpiod_line_release_bulk (struct gpiod_line_bulk ∗bulk) GPIOD_API

  Release a set of previously reserved lines.
- bool gpiod_line_is_requested (struct gpiod_line ∗line) GPIOD_API

  Check if the calling user has ownership of this line.
- bool gpiod_line_is_free (struct gpiod_line ∗line) GPIOD_API

  Check if the calling user has neither requested ownership of this line nor configured any event notifications.

6.7.1 Detailed Description

Interface for requesting GPIO lines from userspace for both values and events.

6.7.2 Enumeration Type Documentation

6.7.2.1 anonymous enum

```
anonymous enum
```

Available types of requests.

Enumerator

GPIOD_LINE_REQUEST_DIRECTION_AS_IS	Request the line(s), but don't change current direction.
GPIOD_LINE_REQUEST_DIRECTION_INPUT	Request the line(s) for reading the GPIO line state.
GPIOD_LINE_REQUEST_DIRECTION_OUTPUT	Request the line(s) for setting the GPIO line state.
GPIOD_LINE_REQUEST_EVENT_FALLING_EDGE	Only watch falling edge events.
GPIOD_LINE_REQUEST_EVENT_RISING_EDGE	Only watch rising edge events.
GPIOD_LINE_REQUEST_EVENT_BOTH_EDGES	Monitor both types of events.

6.7.2.2 anonymous enum

```
anonymous enum
```

Miscellaneous GPIO request flags.

Enumerator

GPIOD_LINE_REQUEST_FLAG_OPEN_DRAIN	The line is an open-drain port.
GPIOD_LINE_REQUEST_FLAG_OPEN_SOURCE	The line is an open-source port.
GPIOD_LINE_REQUEST_FLAG_ACTIVE_LOW	The active state of the line is low (high is the default).
GPIOD_LINE_REQUEST_FLAG_BIAS_DISABLE	The line has neither either pull-up nor pull-down resistor.
GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_DOWN	The line has pull-down resistor enabled.
GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_UP	The line has pull-up resistor enabled.

6.7.3 Function Documentation

6.7.3.1 gpiod_line_is_free()

```
bool gpiod_line_is_free (
            struct gpiod_line * line )
```

Check if the calling user has neither requested ownership of this line nor configured any event notifications.

Parameters

line	GPIO line object.

Returns

True if given line is free, false otherwise.

6.7.3.2 gpiod_line_is_requested()

```
bool gpiod_line_is_requested (
            struct gpiod_line * line )
```

Check if the calling user has ownership of this line.

Parameters

line	GPIO line object.

Returns

True if given line was requested, false otherwise.

6.7.3.3 gpiod_line_release()

```
void gpiod_line_release (
            struct gpiod_line * line )
```

Release a previously reserved line.

Parameters

line	GPIO line object.

6.7.3.4 gpiod_line_release_bulk()

```
void gpiod_line_release_bulk (
             struct gpiod_line_bulk * bulk )
```

Release a set of previously reserved lines.

Parameters

bulk	Set of GPIO lines to release.

If the lines were not previously requested together, the behavior is undefined.

6.7.3.5 gpiod_line_request()

```
int gpiod_line_request (
             struct gpiod_line * line,
             const struct gpiod_line_request_config * config,
             int default_val )
```

Reserve a single line.

Parameters

line	GPIO line object.
config	Request options.
default_val	Initial line value - only relevant if we're setting the direction to output.

Returns

0 if the line was properly reserved. In case of an error this routine returns -1 and sets the last error number.

If this routine succeeds, the caller takes ownership of the GPIO line until it's released.

6.7.3.6 gpiod_line_request_both_edges_events()

```
int gpiod_line_request_both_edges_events (
             struct gpiod_line * line,
             const char * consumer )
```

Request all event type notifications on a single line.

Parameters

line	GPIO line object.
consumer	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.7 gpiod_line_request_both_edges_events_flags()

```
int gpiod_line_request_both_edges_events_flags (
            struct gpiod_line * line,
            const char * consumer,
            int flags )
```

Request all event type notifications on a single line.

Parameters

line	GPIO line object.
consumer	Name of the consumer.
flags	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.8 gpiod_line_request_bulk()

```
int gpiod_line_request_bulk (
            struct gpiod_line_bulk * bulk,
            const struct gpiod_line_request_config * config,
            const int * default_vals )
```

Reserve a set of GPIO lines.

Parameters

bulk	Set of GPIO lines to reserve.
config	Request options.
default_vals	Initial line values - only relevant if we're setting the direction to output.

Returns

0 if the all lines were properly requested. In case of an error this routine returns -1 and sets the last error number.

If this routine succeeds, the caller takes ownership of the GPIO lines until they're released. All the requested lines must be prodivided by the same gpiochip.

6.7.3.9 gpiod_line_request_bulk_both_edges_events()

```
int gpiod_line_request_bulk_both_edges_events (
            struct gpiod_line_bulk * bulk,
            const char * consumer )
```

Request all event type notifications on a set of lines.

Parameters

bulk	Set of GPIO lines to request.
consumer	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.10 gpiod_line_request_bulk_both_edges_events_flags()

```
int gpiod_line_request_bulk_both_edges_events_flags (
            struct gpiod_line_bulk * bulk,
            const char * consumer,
            int flags )
```

Request all event type notifications on a set of lines.

Parameters

bulk	Set of GPIO lines to request.
consumer	Name of the consumer.
flags	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.11 gpiod_line_request_bulk_falling_edge_events()

```
int gpiod_line_request_bulk_falling_edge_events (
            struct gpiod_line_bulk * bulk,
            const char * consumer )
```

Request falling edge event notifications on a set of lines.

Parameters

bulk	Set of GPIO lines to request.
consumer	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.12 gpiod_line_request_bulk_falling_edge_events_flags()

```
int gpiod_line_request_bulk_falling_edge_events_flags (
            struct gpiod_line_bulk * bulk,
            const char * consumer,
            int flags )
```

Request falling edge event notifications on a set of lines.

Parameters

bulk	Set of GPIO lines to request.
consumer	Name of the consumer.
flags	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.13 gpiod_line_request_bulk_input()

```
int gpiod_line_request_bulk_input (
            struct gpiod_line_bulk * bulk,
            const char * consumer )
```

Reserve a set of GPIO lines, set the direction to input.

Parameters

bulk	Set of GPIO lines to reserve.
consumer	Name of the consumer.

Returns

0 if the lines were properly reserved, -1 on failure.

6.7.3.14 gpiod_line_request_bulk_input_flags()

```
int gpiod_line_request_bulk_input_flags (
             struct gpiod_line_bulk * bulk,
             const char * consumer,
             int flags )
```

Reserve a set of GPIO lines, set the direction to input.

Parameters

bulk	Set of GPIO lines to reserve.
consumer	Name of the consumer.
flags	Additional request flags.

Returns

0 if the lines were properly reserved, -1 on failure.

6.7.3.15 gpiod_line_request_bulk_output()

```
int gpiod_line_request_bulk_output (
             struct gpiod_line_bulk * bulk,
             const char * consumer,
             const int * default_vals )
```

Reserve a set of GPIO lines, set the direction to output.

Parameters

bulk	Set of GPIO lines to reserve.
consumer	Name of the consumer.
default_vals	Initial line values.

Returns

0 if the lines were properly reserved, -1 on failure.

6.7.3.16 gpiod_line_request_bulk_output_flags()

```
int gpiod_line_request_bulk_output_flags (
             struct gpiod_line_bulk * bulk,
             const char * consumer,
             int flags,
             const int * default_vals )
```

Reserve a set of GPIO lines, set the direction to output.

Parameters

bulk	Set of GPIO lines to reserve.
consumer	Name of the consumer.
flags	Additional request flags.
default_vals	Initial line values.

Returns

0 if the lines were properly reserved, -1 on failure.

6.7.3.17 gpiod_line_request_bulk_rising_edge_events()

```
int gpiod_line_request_bulk_rising_edge_events (
            struct gpiod_line_bulk * bulk,
            const char * consumer )
```

Request rising edge event notifications on a set of lines.

Parameters

bulk	Set of GPIO lines to request.
consumer	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.18 gpiod_line_request_bulk_rising_edge_events_flags()

```
int gpiod_line_request_bulk_rising_edge_events_flags (
            struct gpiod_line_bulk * bulk,
            const char * consumer,
            int flags )
```

Request rising edge event notifications on a set of lines.

Parameters

bulk	Set of GPIO lines to request.
consumer	Name of the consumer.
flags	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.19 gpiod_line_request_falling_edge_events()

```
int gpiod_line_request_falling_edge_events (
            struct gpiod_line * line,
            const char * consumer )
```

Request falling edge event notifications on a single line.

Parameters

line	GPIO line object.
consumer	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.20 gpiod_line_request_falling_edge_events_flags()

```
int gpiod_line_request_falling_edge_events_flags (
            struct gpiod_line * line,
            const char * consumer,
            int flags )
```

Request falling edge event notifications on a single line.

Parameters

line	GPIO line object.
consumer	Name of the consumer.
flags	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.21 gpiod_line_request_input()

```
int gpiod_line_request_input (
            struct gpiod_line * line,
            const char * consumer )
```

Reserve a single line, set the direction to input.

Parameters

line	GPIO line object.
consumer	Name of the consumer.

Returns

0 if the line was properly reserved, -1 on failure.

6.7.3.22 gpiod_line_request_input_flags()

```
int gpiod_line_request_input_flags (
             struct gpiod_line * line,
             const char * consumer,
             int flags )
```

Reserve a single line, set the direction to input.

Parameters

line	GPIO line object.
consumer	Name of the consumer.
flags	Additional request flags.

Returns

0 if the line was properly reserved, -1 on failure.

6.7.3.23 gpiod_line_request_output()

```
int gpiod_line_request_output (
             struct gpiod_line * line,
             const char * consumer,
             int default_val )
```

Reserve a single line, set the direction to output.

Parameters

line	GPIO line object.
consumer	Name of the consumer.
default_val	Initial line value.

Returns

0 if the line was properly reserved, -1 on failure.

6.7.3.24 gpiod_line_request_output_flags()

```
int gpiod_line_request_output_flags (
            struct gpiod_line * line,
            const char * consumer,
            int flags,
            int default_val )
```

Reserve a single line, set the direction to output.

Parameters

line	GPIO line object.
consumer	Name of the consumer.
flags	Additional request flags.
default_val	Initial line value.

Returns

0 if the line was properly reserved, -1 on failure.

6.7.3.25 gpiod_line_request_rising_edge_events()

```
int gpiod_line_request_rising_edge_events (
            struct gpiod_line * line,
            const char * consumer )
```

Request rising edge event notifications on a single line.

Parameters

line	GPIO line object.
consumer	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

6.7.3.26 gpiod_line_request_rising_edge_events_flags()

```
int gpiod_line_request_rising_edge_events_flags (
             struct gpiod_line * line,
             const char * consumer,
             int flags )
```

Request rising edge event notifications on a single line.

Parameters

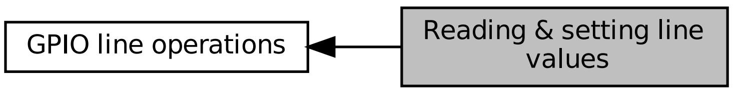
line	GPIO line object.
consumer	Name of the consumer.
flags	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

6.8 Reading & setting line values

Collaboration diagram for Reading & setting line values:



Functions

- int gpiod_line_get_value (struct gpiod_line ∗line) GPIOD_API

  Read current value of a single GPIO line.
- int gpiod_line_get_value_bulk (struct gpiod_line_bulk ∗bulk, int ∗values) GPIOD_API

  Read current values of a set of GPIO lines.
- int gpiod_line_set_value (struct gpiod_line ∗line, int value) GPIOD_API

  Set the value of a single GPIO line.
- int gpiod_line_set_value_bulk (struct gpiod_line_bulk ∗bulk, const int ∗values) GPIOD_API

  Set the values of a set of GPIO lines.

6.8.1 Detailed Description

Functions allowing to read and set GPIO line values for single lines and in bulk.

6.8.2 Function Documentation

6.8.2.1 gpiod_line_get_value()

```
int gpiod_line_get_value (
            struct gpiod_line * line )
```

Read current value of a single GPIO line.

Parameters

line	GPIO line object.

Returns

0 or 1 if the operation succeeds. On error this routine returns -1 and sets the last error number.

6.8.2.2 gpiod_line_get_value_bulk()

```
int gpiod_line_get_value_bulk (
            struct gpiod_line_bulk * bulk,
            int * values )
```

Read current values of a set of GPIO lines.

Parameters

bulk	Set of GPIO lines to reserve.
values	An array big enough to hold line_bulk->num_lines values.

Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

If succeeds, this routine fills the values array with a set of values in the same order, the lines are added to line_bulk. If the lines were not previously requested together, the behavior is undefined.

6.8.2.3 gpiod_line_set_value()

```
int gpiod_line_set_value (
            struct gpiod_line * line,
            int value )
```

Set the value of a single GPIO line.

Parameters

line	GPIO line object.
value	New value.

Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

6.8.2.4 gpiod_line_set_value_bulk()

```
int gpiod_line_set_value_bulk (
            struct gpiod_line_bulk * bulk,
            const int * values )
```

Set the values of a set of GPIO lines.

Parameters

bulk	Set of GPIO lines to reserve.
values	An array holding line_bulk->num_lines new values for lines. A NULL pointer is interpreted as a logical low for all lines.

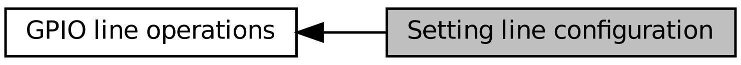
Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

If the lines were not previously requested together, the behavior is undefined.

6.9 Setting line configuration

Collaboration diagram for Setting line configuration:



Functions

- int gpiod_line_set_config (struct gpiod_line ∗line, int direction, int flags, int value) GPIOD_API

    Update the configuration of a single GPIO line.
- int gpiod_line_set_config_bulk (struct gpiod_line_bulk ∗bulk, int direction, int flags, const int ∗values) GPIOD_API

    Update the configuration of a set of GPIO lines.
- int gpiod_line_set_flags (struct gpiod_line ∗line, int flags) GPIOD_API

    Update the configuration flags of a single GPIO line.
- int gpiod_line_set_flags_bulk (struct gpiod_line_bulk ∗bulk, int flags) GPIOD_API

    Update the configuration flags of a set of GPIO lines.
- int gpiod_line_set_direction_input (struct gpiod_line ∗line) GPIOD_API

    Set the direction of a single GPIO line to input.
- int gpiod_line_set_direction_input_bulk (struct gpiod_line_bulk ∗bulk) GPIOD_API

    Set the direction of a set of GPIO lines to input.
- int gpiod_line_set_direction_output (struct gpiod_line ∗line, int value) GPIOD_API

    Set the direction of a single GPIO line to output.
- int gpiod_line_set_direction_output_bulk (struct gpiod_line_bulk ∗bulk, const int ∗values) GPIOD_API

    Set the direction of a set of GPIO lines to output.

6.9.1 Detailed Description

Functions allowing modification of config options of GPIO lines requested from user-space.

6.9.2 Function Documentation

6.9.2.1 gpiod_line_set_config()

```
int gpiod_line_set_config (
             struct gpiod_line ∗ line,
             int direction,
             int flags,
             int value )
```

Update the configuration of a single GPIO line.

Parameters

line	GPIO line object.
direction	Updated direction which may be one of GPIOD_LINE_REQUEST_DIRECTION_AS_IS, GPIOD_LINE_REQUEST_DIRECTION_INPUT, or GPIOD_LINE_REQUEST_DIRECTION_OUTPUT.
flags	Replacement flags.
value	The new output value for the line when direction is GPIOD_LINE_REQUEST_DIRECTION_OUTPUT.

Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

6.9.2.2 gpiod_line_set_config_bulk()

```
int gpiod_line_set_config_bulk (
             struct gpiod_line_bulk * bulk,
             int direction,
             int flags,
             const int * values )
```

Update the configuration of a set of GPIO lines.

Parameters

bulk	Set of GPIO lines.
direction	Updated direction which may be one of GPIOD_LINE_REQUEST_DIRECTION_AS_IS, GPIOD_LINE_REQUEST_DIRECTION_INPUT, or GPIOD_LINE_REQUEST_DIRECTION_OUTPUT.
flags	Replacement flags.
values	An array holding line_bulk->num_lines new logical values for lines when direction is GPIOD_LINE_REQUEST_DIRECTION_OUTPUT. A NULL pointer is interpreted as a logical low for all lines.

Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

If the lines were not previously requested together, the behavior is undefined.

6.9.2.3 gpiod_line_set_direction_input()

```
int gpiod_line_set_direction_input (
             struct gpiod_line * line )
```

Set the direction of a single GPIO line to input.

Parameters

line	GPIO line object.

Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

6.9.2.4 gpiod_line_set_direction_input_bulk()

```
int gpiod_line_set_direction_input_bulk (
             struct gpiod_line_bulk * bulk )
```

Set the direction of a set of GPIO lines to input.

Parameters

bulk	Set of GPIO lines.

Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

If the lines were not previously requested together, the behavior is undefined.

6.9.2.5 gpiod_line_set_direction_output()

```
int gpiod_line_set_direction_output (
             struct gpiod_line * line,
             int value )
```

Set the direction of a single GPIO line to output.

Parameters

line	GPIO line object.
value	The logical value output on the line.

Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

6.9.2.6 gpiod_line_set_direction_output_bulk()

```
int gpiod_line_set_direction_output_bulk (
             struct gpiod_line_bulk * bulk,
             const int * values )
```

Set the direction of a set of GPIO lines to output.

Parameters

bulk	Set of GPIO lines.
values	An array holding line_bulk->num_lines new logical values for lines. A NULL pointer is interpreted as a logical low for all lines.

Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

If the lines were not previously requested together, the behavior is undefined.

6.9.2.7 gpiod_line_set_flags()

```
int gpiod_line_set_flags (
            struct gpiod_line * line,
            int flags )
```

Update the configuration flags of a single GPIO line.

Parameters

line	GPIO line object.
flags	Replacement flags.

Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

6.9.2.8 gpiod_line_set_flags_bulk()

```
int gpiod_line_set_flags_bulk (
            struct gpiod_line_bulk * bulk,
            int flags )
```

Update the configuration flags of a set of GPIO lines.

Parameters

bulk	Set of GPIO lines.
flags	Replacement flags.

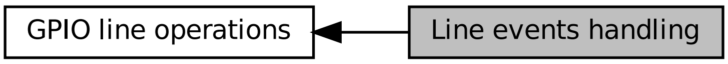
Returns

0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

If the lines were not previously requested together, the behavior is undefined.

6.10 Line events handling

Collaboration diagram for Line events handling:



Classes

- struct gpiod_line_event

  Structure holding event info.

Enumerations

- enum { GPIOD_LINE_EVENT_RISING_EDGE = 1 , GPIOD_LINE_EVENT_FALLING_EDGE }

  Event types.

Functions

- int gpiod_line_event_wait (struct gpiod_line ∗line, const struct timespec ∗timeout) GPIOD_API

  Wait for an event on a single line.
- int gpiod_line_event_wait_bulk (struct gpiod_line_bulk ∗bulk, const struct timespec ∗timeout, struct gpiod_line_bulk ∗event_bulk) GPIOD_API

  Wait for events on a set of lines.
- int gpiod_line_event_read (struct gpiod_line ∗line, struct gpiod_line_event ∗event) GPIOD_API

  Read next pending event from the GPIO line.
- int gpiod_line_event_read_multiple (struct gpiod_line ∗line, struct gpiod_line_event ∗events, unsigned int num_events) GPIOD_API

  Read up to a certain number of events from the GPIO line.
- int gpiod_line_event_get_fd (struct gpiod_line ∗line) GPIOD_API

  Get the event file descriptor.
- int gpiod_line_event_read_fd (int fd, struct gpiod_line_event ∗event) GPIOD_API

  Read the last GPIO event directly from a file descriptor.
- int gpiod_line_event_read_fd_multiple (int fd, struct gpiod_line_event ∗events, unsigned int num_events) GPIOD_API

  Read up to a certain number of events directly from a file descriptor.

6.10.1 Detailed Description

Structures and functions allowing to poll lines for events and read them, both for individual lines as well as in bulk. Also contains functions for retrieving the associated file descriptors and operate on them for easy integration with standard unix interfaces.

6.10.2 Enumeration Type Documentation

6.10.2.1 anonymous enum

```
anonymous enum
```

Event types.

Enumerator

GPIOD_LINE_EVENT_RISING_EDGE	Rising edge event.
GPIOD_LINE_EVENT_FALLING_EDGE	Falling edge event.

6.10.3 Function Documentation

6.10.3.1 gpiod_line_event_get_fd()

```
int gpiod_line_event_get_fd (
            struct gpiod_line * line )
```

Get the event file descriptor.

Parameters

line	GPIO line object.

Returns

Number of the event file descriptor or -1 if the user tries to retrieve the descriptor from a line that wasn't configured for event monitoring.

Users may want to poll the event file descriptor on their own. This routine allows to access it.

6.10.3.2 gpiod_line_event_read()

```
int gpiod_line_event_read (
            struct gpiod_line * line,
            struct gpiod_line_event * event )
```

Read next pending event from the GPIO line.

Parameters

line	GPIO line object.
event	Buffer to which the event data will be copied.

Returns

0 if the event was read correctly, -1 on error.

Note

This function will block if no event was queued for this line.

6.10.3.3 gpiod_line_event_read_fd()

```
int gpiod_line_event_read_fd (
              int fd,
              struct gpiod_line_event * event )
```

Read the last GPIO event directly from a file descriptor.

Parameters

fd	File descriptor.
event	Buffer in which the event data will be stored.

Returns

0 if the event was read correctly, -1 on error.

Users who directly poll the file descriptor for incoming events can also directly read the event data from it using this routine. This function translates the kernel representation of the event to the libgpiod format.

6.10.3.4 gpiod_line_event_read_fd_multiple()

```
int gpiod_line_event_read_fd_multiple (
              int fd,
              struct gpiod_line_event * events,
              unsigned int num_events )
```

Read up to a certain number of events directly from a file descriptor.

Parameters

fd	File descriptor.
events	Buffer to which the event data will be copied. Must hold at least the amount of events specified in num_events.
num_events	Specifies how many events can be stored in the buffer.

Returns

On success returns the number of events stored in the buffer, on failure -1 is returned.

6.10.3.5 gpiod_line_event_read_multiple()

```
int gpiod_line_event_read_multiple (
            struct gpiod_line * line,
            struct gpiod_line_event * events,
            unsigned int num_events )
```

Read up to a certain number of events from the GPIO line.

Parameters

line	GPIO line object.
events	Buffer to which the event data will be copied. Must hold at least the amount of events specified in num_events.
num_events	Specifies how many events can be stored in the buffer.

Returns

On success returns the number of events stored in the buffer, on failure -1 is returned.

6.10.3.6 gpiod_line_event_wait()

```
int gpiod_line_event_wait (
            struct gpiod_line * line,
            const struct timespec * timeout )
```

Wait for an event on a single line.

Parameters

line	GPIO line object.
timeout	Wait time limit.

Returns

0 if wait timed out, -1 if an error occurred, 1 if an event occurred.

6.10.3.7 gpiod_line_event_wait_bulk()

```
int gpiod_line_event_wait_bulk (
            struct gpiod_line_bulk * bulk,
            const struct timespec * timeout,
            struct gpiod_line_bulk * event_bulk )
```

Wait for events on a set of lines.

Parameters

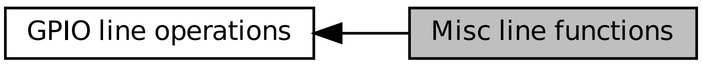
bulk	Set of GPIO lines to monitor.
timeout	Wait time limit.
event_bulk	Bulk object in which to store the line handles on which events occurred. Can be NULL.

Returns

0 if wait timed out, -1 if an error occurred, 1 if at least one event occurred.

6.11 Misc line functions

Collaboration diagram for Misc line functions:



Functions

- struct gpiod_line ∗ gpiod_line_get (const char ∗device, unsigned int offset) GPIOD_API

  Get a GPIO line handle by GPIO chip description and offset.
- struct gpiod_line ∗ gpiod_line_find (const char ∗name) GPIOD_API

  Find a GPIO line by its name.
- void gpiod_line_close_chip (struct gpiod_line ∗line) GPIOD_API

  Close a GPIO chip owning this line and release all resources.
- struct gpiod_chip ∗ gpiod_line_get_chip (struct gpiod_line ∗line) GPIOD_API

  Get the handle to the GPIO chip controlling this line.

6.11.1 Detailed Description

Functions that didn't fit anywhere else.

6.11.2 Function Documentation

6.11.2.1 gpiod_line_close_chip()

```
void gpiod_line_close_chip (
            struct gpiod_line * line )
```

Close a GPIO chip owning this line and release all resources.

Parameters

line	GPIO line object

After this function returns, the line must no longer be used.

6.11.2.2 gpiod_line_find()

```
struct gpiod_line * gpiod_line_find (
            const char * name )
```

Find a GPIO line by its name.

Parameters

name	Name of the GPIO line.

Returns

Returns the GPIO line handle if the line exists in the system or NULL if it couldn't be located or an error occurred.

Attention

GPIO lines are not unique in the linux kernel, neither globally nor within a single chip. This function finds the first line with given name.

If this routine succeeds, the user must manually close the GPIO chip owning this line to avoid memory leaks. If the line could not be found, this functions sets errno to ENOENT.

6.11.2.3 gpiod_line_get()

```
struct gpiod_line * gpiod_line_get (
            const char * device,
            unsigned int offset )
```

Get a GPIO line handle by GPIO chip description and offset.

Parameters

device	String describing the gpiochip.
offset	The offset of the GPIO line.

Returns

GPIO line handle or NULL if an error occurred.

This routine provides a shorter alternative to calling gpiod_chip_open_lookup and gpiod_chip_get_line.

If this function succeeds, the caller is responsible for closing the associated GPIO chip.

6.11.2.4 gpiod_line_get_chip()

```
struct gpiod_chip * gpiod_line_get_chip (
            struct gpiod_line * line )
```

Get the handle to the GPIO chip controlling this line.

Parameters

line	The GPIO line object.

Returns

Pointer to the GPIO chip handle controlling this line.

6.12 Iterators for GPIO chips and lines

Macros

- #define gpiod_foreach_chip(iter, chip)
  Iterate over all GPIO chips present in the system.
- #define gpiod_foreach_chip_noclose(iter, chip)
  Iterate over all chips present in the system without closing them.
- #define gpiod_foreach_line(iter, line)
  Iterate over all GPIO lines of a single chip.

Functions

- struct gpiod_chip_iter ∗ gpiod_chip_iter_new (void) GPIOD_API
  Create a new gpiochip iterator.
- void gpiod_chip_iter_free (struct gpiod_chip_iter ∗iter) GPIOD_API
  Release all resources allocated for the gpiochip iterator and close the most recently opened gpiochip (if any).
- void gpiod_chip_iter_free_noclose (struct gpiod_chip_iter ∗iter) GPIOD_API
  Release all resources allocated for the gpiochip iterator but don't close the most recently opened gpiochip (if any).

- struct gpiod_chip ∗ gpiod_chip_iter_next (struct gpiod_chip_iter ∗iter) GPIOD_API

    Get the next gpiochip handle.
- struct gpiod_chip ∗ gpiod_chip_iter_next_noclose (struct gpiod_chip_iter ∗iter) GPIOD_API

    Get the next gpiochip handle without closing the previous one.
- struct gpiod_line_iter ∗ gpiod_line_iter_new (struct gpiod_chip ∗chip) GPIOD_API

    Create a new line iterator.
- void gpiod_line_iter_free (struct gpiod_line_iter ∗iter) GPIOD_API

    Free all resources associated with a GPIO line iterator.
- struct gpiod_line ∗ gpiod_line_iter_next (struct gpiod_line_iter ∗iter) GPIOD_API

    Get the next GPIO line handle.

6.12.1 Detailed Description

These functions and data structures allow easy iterating over GPIO chips and lines.

6.12.2 Macro Definition Documentation

6.12.2.1 gpiod_foreach_chip

```
#define gpiod_foreach_chip(
                iter,
                chip )
```

Value:

```
    for ((chip) = gpiod_chip_iter_next(iter);           \
         (chip);                                        \
         (chip) = gpiod_chip_iter_next(iter))
```

Iterate over all GPIO chips present in the system.

Parameters

iter	An initialized GPIO chip iterator.
chip	Pointer to a GPIO chip handle. On each iteration the newly opened chip handle is assigned to this argument.

The user must not close the GPIO chip manually - instead the previous chip handle is closed automatically on the next iteration. The last chip to be opened is closed internally by gpiod_chip_iter_free.

6.12.2.2 gpiod_foreach_chip_noclose

```
#define gpiod_foreach_chip_noclose(
                iter,
                chip )
```

Value:

```
    for ((chip) = gpiod_chip_iter_next_noclose(iter);       \
         (chip);                                             \
         (chip) = gpiod_chip_iter_next_noclose(iter))
```

Iterate over all chips present in the system without closing them.

Parameters

iter	An initialized GPIO chip iterator.
chip	Pointer to a GPIO chip handle. On each iteration the newly opened chip handle is assigned to this argument.

The user must close all the GPIO chips manually after use, until then, the chips remain open. Free the iterator by calling gpiod_chip_iter_free_noclose to avoid closing the last chip automatically.

6.12.2.3 gpiod_foreach_line

```
#define gpiod_foreach_line(
                iter,
                line )
```

Value:

```
    for ((line) = gpiod_line_iter_next(iter);           \
         (line);                                         \
         (line) = gpiod_line_iter_next(iter))
```

Iterate over all GPIO lines of a single chip.

Parameters

iter	An initialized GPIO line iterator.
line	Pointer to a GPIO line handle - on each iteration, the next GPIO line will be assigned to this argument.

6.12.3 Function Documentation

6.12.3.1 gpiod_chip_iter_free()

```
void gpiod_chip_iter_free (
             struct gpiod_chip_iter * iter )
```

Release all resources allocated for the gpiochip iterator and close the most recently opened gpiochip (if any).

Parameters

iter	The gpiochip iterator object.

6.12.3.2 gpiod_chip_iter_free_noclose()

```
void gpiod_chip_iter_free_noclose (
            struct gpiod_chip_iter * iter )
```

Release all resources allocated for the gpiochip iterator but don't close the most recently opened gpiochip (if any).

Parameters

iter	The gpiochip iterator object.

Users may want to break the loop when iterating over gpiochips and keep the most recently opened chip active while freeing the iterator data. This routine enables that.

6.12.3.3 gpiod_chip_iter_new()

```
struct gpiod_chip_iter * gpiod_chip_iter_new (
            void )
```

Create a new gpiochip iterator.

Returns

Pointer to a new chip iterator object or NULL if an error occurred.

Internally this routine scans the /dev/ directory for GPIO chip device files, opens them and stores their the handles until gpiod_chip_iter_free or gpiod_chip_iter_free_noclose is called.

6.12.3.4 gpiod_chip_iter_next()

```
struct gpiod_chip * gpiod_chip_iter_next (
            struct gpiod_chip_iter * iter )
```

Get the next gpiochip handle.

Parameters

iter	The gpiochip iterator object.

Returns

Pointer to the next open gpiochip handle or NULL if no more chips are present in the system.

Note

The previous chip handle will be closed using gpiod_chip_iter_free.

6.12.3.5 gpiod_chip_iter_next_noclose()

```
struct gpiod_chip * gpiod_chip_iter_next_noclose (
            struct gpiod_chip_iter * iter )
```

Get the next gpiochip handle without closing the previous one.

Parameters

iter	The gpiochip iterator object.

Returns

Pointer to the next open gpiochip handle or NULL if no more chips are present in the system.

Note

This function works just like gpiod_chip_iter_next but doesn't close the most recently opened chip handle.

6.12.3.6 gpiod_line_iter_free()

```
void gpiod_line_iter_free (
            struct gpiod_line_iter * iter )
```

Free all resources associated with a GPIO line iterator.

Parameters

iter	Line iterator object.

6.12.3.7 gpiod_line_iter_new()

```
struct gpiod_line_iter * gpiod_line_iter_new (
            struct gpiod_chip * chip )
```

Create a new line iterator.

Parameters

chip	Active gpiochip handle over the lines of which we want to iterate.

Returns

New line iterator or NULL if an error occurred.

6.12.3.8 gpiod_line_iter_next()

```
struct gpiod_line * gpiod_line_iter_next (
             struct gpiod_line_iter * iter )
```

Get the next GPIO line handle.

Parameters

iter	The GPIO line iterator object.

Returns

Pointer to the next GPIO line handle or NULL if there are no more lines left.

6.13 Stuff that didn't fit anywhere else

Functions

- const char * gpiod_version_string (void) GPIOD_API

  Get the API version of the library as a human-readable string.

6.13.1 Detailed Description

Various libgpiod-related functions.

6.13.2 Function Documentation

6.13.2.1 gpiod_version_string()

```
const char * gpiod_version_string (
             void  )
```

Get the API version of the library as a human-readable string.

Returns

Human-readable string containing the library version.

Chapter 7

Class Documentation

7.1 gpiod_ctxless_event_poll_fd Struct Reference

Helper structure for the ctxless event loop poll callback.

```
#include <gpiod.h>
```

Public Attributes

- int fd
- bool event

7.1.1 Detailed Description

Helper structure for the ctxless event loop poll callback.

7.1.2 Member Data Documentation

7.1.2.1 event

```
bool gpiod_ctxless_event_poll_fd::event
```

Indicates whether an event occurred on this file descriptor.

7.1.2.2 fd

```
int gpiod_ctxless_event_poll_fd::fd
```

File descriptor number.

The documentation for this struct was generated from the following file:

- gpiod.h

7.2 gpiod_line_bulk Struct Reference

Helper structure for storing a set of GPIO line objects.

```
#include <gpiod.h>
```

Public Attributes

- struct gpiod_line ∗ lines [GPIOD_LINE_BULK_MAX_LINES]
- unsigned int num_lines

7.2.1 Detailed Description

Helper structure for storing a set of GPIO line objects.

This structure is used in all operations involving sets of GPIO lines. If a bulk object is being passed to a function while containing zero lines, the result is undefined.

7.2.2 Member Data Documentation

7.2.2.1 lines

```
struct gpiod_line* gpiod_line_bulk::lines[GPIOD_LINE_BULK_MAX_LINES]
```

Buffer for line pointers.

7.2.2.2 num_lines

```
unsigned int gpiod_line_bulk::num_lines
```

Number of lines currently held in this structure.

The documentation for this struct was generated from the following file:

- gpiod.h

7.3 gpiod_line_event Struct Reference

Structure holding event info.

```
#include <gpiod.h>
```

Public Attributes

- struct timespec ts
- int event_type

7.3.1 Detailed Description

Structure holding event info.

7.3.2 Member Data Documentation

7.3.2.1 event_type

`int gpiod_line_event::event_type`

Type of the event that occurred.

7.3.2.2 ts

`struct timespec gpiod_line_event::ts`

Best estimate of time of event occurrence.

The documentation for this struct was generated from the following file:

- gpiod.h

7.4 gpiod_line_request_config Struct Reference

Structure holding configuration of a line request.

`#include <gpiod.h>`

Public Attributes

- const char ∗ consumer
- int request_type
- int flags

7.4.1 Detailed Description

Structure holding configuration of a line request.

7.4.2 Member Data Documentation

7.4.2.1 consumer

```
const char* gpiod_line_request_config::consumer
```

Name of the consumer.

7.4.2.2 flags

```
int gpiod_line_request_config::flags
```

Other configuration flags.

7.4.2.3 request_type

```
int gpiod_line_request_config::request_type
```

Request type.

The documentation for this struct was generated from the following file:

- gpiod.h

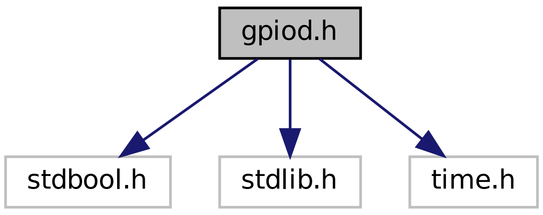
Chapter 8

File Documentation

8.1 gpiod.h File Reference

```
#include <stdbool.h>
#include <stdlib.h>
#include <time.h>
```

Include dependency graph for gpiod.h:



Classes

- struct gpiod_ctxless_event_poll_fd

  Helper structure for the ctxless event loop poll callback.
- struct gpiod_line_bulk

  Helper structure for storing a set of GPIO line objects.
- struct gpiod_line_request_config

  Structure holding configuration of a line request.
- struct gpiod_line_event

  Structure holding event info.

Macros

- #define **GPIOD_API** __attribute__((visibility("default")))

  Makes symbol visible.
- #define **GPIOD_UNUSED** __attribute__((unused))

  Marks a function argument or variable as potentially unused.
- #define GPIOD_BIT(nr) (1UL << (nr))

  Shift 1 by given offset.
- #define **GPIOD_DEPRECATED** __attribute__((deprecated))

  Marks a public function as deprecated.
- #define **GPIOD_LINE_BULK_MAX_LINES** 64

  Maximum number of GPIO lines that can be requested at once.
- #define GPIOD_LINE_BULK_INITIALIZER { { NULL }, 0 }

  Static initializer for GPIO bulk objects.
- #define gpiod_line_bulk_foreach_line(bulk, line, lineptr)

  Iterate over all line handles held by a line bulk object.
- #define gpiod_line_bulk_foreach_line_off(bulk, line, offset)

  Iterate over all line handles held by a line bulk object (integer counter variant).
- #define gpiod_foreach_chip(iter, chip)

  Iterate over all GPIO chips present in the system.
- #define gpiod_foreach_chip_noclose(iter, chip)

  Iterate over all chips present in the system without closing them.
- #define gpiod_foreach_line(iter, line)

  Iterate over all GPIO lines of a single chip.

Typedefs

- typedef void(∗ **gpiod_ctxless_set_value_cb**) (void ∗)

  Simple set value callback signature.
- typedef int(∗ gpiod_ctxless_event_handle_cb) (int, unsigned int, const struct timespec ∗, void ∗)

  Simple event callback signature.
- typedef int(∗ gpiod_ctxless_event_poll_cb) (unsigned int, struct gpiod_ctxless_event_poll_fd ∗, const struct timespec ∗, void ∗)

  Simple event poll callback signature.

Enumerations

- enum {
  GPIOD_CTXLESS_FLAG_OPEN_DRAIN = GPIOD_BIT(0) , GPIOD_CTXLESS_FLAG_OPEN_SOURCE = GPIOD_BIT(1) , GPIOD_CTXLESS_FLAG_BIAS_DISABLE = GPIOD_BIT(2) , GPIOD_CTXLESS_FLAG_BIAS_PULL_DOWN = GPIOD_BIT(3) ,
  GPIOD_CTXLESS_FLAG_BIAS_PULL_UP = GPIOD_BIT(4) }

  Miscellaneous GPIO flags.
- enum { GPIOD_CTXLESS_EVENT_RISING_EDGE = 1 , GPIOD_CTXLESS_EVENT_FALLING_EDGE , **GPIOD_CTXLESS_EVENT_BOTH_EDGES** }

  Event types that the ctxless event monitor can wait for.
- enum { GPIOD_CTXLESS_EVENT_CB_TIMEOUT = 1 , GPIOD_CTXLESS_EVENT_CB_RISING_EDGE , GPIOD_CTXLESS_EVENT_CB_FALLING_EDGE }

  Event types that can be passed to the ctxless event callback.
- enum { GPIOD_CTXLESS_EVENT_CB_RET_ERR = -1 , GPIOD_CTXLESS_EVENT_CB_RET_OK = 0 , GPIOD_CTXLESS_EVENT_CB_RET_STOP = 1 }

Return status values that the ctxless event callback can return.

- enum { GPIOD_CTXLESS_EVENT_POLL_RET_STOP = -2 , GPIOD_CTXLESS_EVENT_POLL_RET_ERR = -1 , GPIOD_CTXLESS_EVENT_POLL_RET_TIMEOUT = 0 }

  Return status values that the ctxless event poll callback can return.
- enum { GPIOD_LINE_DIRECTION_INPUT = 1 , GPIOD_LINE_DIRECTION_OUTPUT }

  Possible direction settings.
- enum { GPIOD_LINE_ACTIVE_STATE_HIGH = 1 , GPIOD_LINE_ACTIVE_STATE_LOW }

  Possible active state settings.
- enum { GPIOD_LINE_BIAS_AS_IS = 1 , GPIOD_LINE_BIAS_DISABLE , GPIOD_LINE_BIAS_PULL_UP , GPIOD_LINE_BIAS_PULL_DOWN }

  Possible internal bias settings.
- enum {
  GPIOD_LINE_REQUEST_DIRECTION_AS_IS = 1 , GPIOD_LINE_REQUEST_DIRECTION_INPUT , GPIOD_LINE_REQUEST_DIRECTION_OUTPUT , GPIOD_LINE_REQUEST_EVENT_FALLING_EDGE ,
  GPIOD_LINE_REQUEST_EVENT_RISING_EDGE , GPIOD_LINE_REQUEST_EVENT_BOTH_EDGES }

  Available types of requests.
- enum {
  GPIOD_LINE_REQUEST_FLAG_OPEN_DRAIN = GPIOD_BIT(0) , GPIOD_LINE_REQUEST_FLAG_OPEN_SOURCE = GPIOD_BIT(1) , GPIOD_LINE_REQUEST_FLAG_ACTIVE_LOW = GPIOD_BIT(2) , GPIOD_LINE_REQUEST_FLAG_BIAS_
  = GPIOD_BIT(3) ,
  GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_DOWN = GPIOD_BIT(4) , GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_UP = GPIOD_BIT(5) }

  Miscellaneous GPIO request flags.
- enum { GPIOD_LINE_EVENT_RISING_EDGE = 1 , GPIOD_LINE_EVENT_FALLING_EDGE }

  Event types.

Functions

- int gpiod_ctxless_get_value (const char ∗device, unsigned int offset, bool active_low, const char ∗consumer) GPIOD_API

  Read current value from a single GPIO line.
- int gpiod_ctxless_get_value_ext (const char ∗device, unsigned int offset, bool active_low, const char ∗consumer, int flags) GPIOD_API

  Read current value from a single GPIO line.
- int gpiod_ctxless_get_value_multiple (const char ∗device, const unsigned int ∗offsets, int ∗values, unsigned int num_lines, bool active_low, const char ∗consumer) GPIOD_API

  Read current values from a set of GPIO lines.
- int gpiod_ctxless_get_value_multiple_ext (const char ∗device, const unsigned int ∗offsets, int ∗values, unsigned int num_lines, bool active_low, const char ∗consumer, int flags) GPIOD_API

  Read current values from a set of GPIO lines.
- int gpiod_ctxless_set_value (const char ∗device, unsigned int offset, int value, bool active_low, const char ∗consumer, gpiod_ctxless_set_value_cb cb, void ∗data) GPIOD_API

  Set value of a single GPIO line.
- int gpiod_ctxless_set_value_ext (const char ∗device, unsigned int offset, int value, bool active_low, const char ∗consumer, gpiod_ctxless_set_value_cb cb, void ∗data, int flags) GPIOD_API

  Set value of a single GPIO line.
- int gpiod_ctxless_set_value_multiple (const char ∗device, const unsigned int ∗offsets, const int ∗values, unsigned int num_lines, bool active_low, const char ∗consumer, gpiod_ctxless_set_value_cb cb, void ∗data) GPIOD_API

  Set values of multiple GPIO lines.

- int gpiod_ctxless_set_value_multiple_ext (const char ∗device, const unsigned int ∗offsets, const int ∗values, unsigned int num_lines, bool active_low, const char ∗consumer, gpiod_ctxless_set_value_cb cb, void ∗data, int flags) GPIOD_API

    Set values of multiple GPIO lines.
- int gpiod_ctxless_event_loop (const char ∗device, unsigned int offset, bool active_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data) GPIOD_API GPIOD_DEPRECATED

    Wait for events on a single GPIO line.
- int gpiod_ctxless_event_loop_multiple (const char ∗device, const unsigned int ∗offsets, unsigned int num←_lines, bool active_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data) GPIOD_API GPIOD_DEPRECATED

    Wait for events on multiple GPIO lines.
- int gpiod_ctxless_event_monitor (const char ∗device, int event_type, unsigned int offset, bool active←_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data) GPIOD_API

    Wait for events on a single GPIO line.
- int gpiod_ctxless_event_monitor_ext (const char ∗device, int event_type, unsigned int offset, bool active←_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data, int flags) GPIOD_API

    Wait for events on a single GPIO line.
- int gpiod_ctxless_event_monitor_multiple (const char ∗device, int event_type, const unsigned int ∗offsets, unsigned int num_lines, bool active_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data) GPIOD_API

    Wait for events on multiple GPIO lines.
- int gpiod_ctxless_event_monitor_multiple_ext (const char ∗device, int event_type, const unsigned int ∗offsets, unsigned int num_lines, bool active_low, const char ∗consumer, const struct timespec ∗timeout, gpiod_ctxless_event_poll_cb poll_cb, gpiod_ctxless_event_handle_cb event_cb, void ∗data, int flags) GPIOD_API

    Wait for events on multiple GPIO lines.
- int gpiod_ctxless_find_line (const char ∗name, char ∗chipname, size_t chipname_size, unsigned int ∗offset) GPIOD_API

    Determine the chip name and line offset of a line with given name.
- struct gpiod_chip ∗ gpiod_chip_open (const char ∗path) GPIOD_API

    Open a gpiochip by path.
- struct gpiod_chip ∗ gpiod_chip_open_by_name (const char ∗name) GPIOD_API

    Open a gpiochip by name.
- struct gpiod_chip ∗ gpiod_chip_open_by_number (unsigned int num) GPIOD_API

    Open a gpiochip by number.
- struct gpiod_chip ∗ gpiod_chip_open_by_label (const char ∗label) GPIOD_API

    Open a gpiochip by label.
- struct gpiod_chip ∗ gpiod_chip_open_lookup (const char ∗descr) GPIOD_API

    Open a gpiochip based on the best guess what the path is.
- void gpiod_chip_close (struct gpiod_chip ∗chip) GPIOD_API

    Close a GPIO chip handle and release all allocated resources.
- const char ∗ gpiod_chip_name (struct gpiod_chip ∗chip) GPIOD_API

    Get the GPIO chip name as represented in the kernel.
- const char ∗ gpiod_chip_label (struct gpiod_chip ∗chip) GPIOD_API

    Get the GPIO chip label as represented in the kernel.
- unsigned int gpiod_chip_num_lines (struct gpiod_chip ∗chip) GPIOD_API

    Get the number of GPIO lines exposed by this chip.
- struct gpiod_line ∗ gpiod_chip_get_line (struct gpiod_chip ∗chip, unsigned int offset) GPIOD_API

    Get the handle to the GPIO line at given offset.

- int gpiod_chip_get_lines (struct gpiod_chip ∗chip, unsigned int ∗offsets, unsigned int num_offsets, struct gpiod_line_bulk ∗bulk) GPIOD_API

    Retrieve a set of lines and store them in a line bulk object.
- int gpiod_chip_get_all_lines (struct gpiod_chip ∗chip, struct gpiod_line_bulk ∗bulk) GPIOD_API

    Retrieve all lines exposed by a chip and store them in a bulk object.
- struct gpiod_line ∗ gpiod_chip_find_line (struct gpiod_chip ∗chip, const char ∗name) GPIOD_API

    Find a GPIO line by name among lines associated with given GPIO chip.
- int gpiod_chip_find_lines (struct gpiod_chip ∗chip, const char ∗∗names, struct gpiod_line_bulk ∗bulk) GPIOD_API

    Find a set of GPIO lines by names among lines exposed by this chip.
- unsigned int gpiod_line_offset (struct gpiod_line ∗line) GPIOD_API

    Read the GPIO line offset.
- const char ∗ gpiod_line_name (struct gpiod_line ∗line) GPIOD_API

    Read the GPIO line name.
- const char ∗ gpiod_line_consumer (struct gpiod_line ∗line) GPIOD_API

    Read the GPIO line consumer name.
- int gpiod_line_direction (struct gpiod_line ∗line) GPIOD_API

    Read the GPIO line direction setting.
- int gpiod_line_active_state (struct gpiod_line ∗line) GPIOD_API

    Read the GPIO line active state setting.
- int gpiod_line_bias (struct gpiod_line ∗line) GPIOD_API

    Read the GPIO line bias setting.
- bool gpiod_line_is_used (struct gpiod_line ∗line) GPIOD_API

    Check if the line is currently in use.
- bool gpiod_line_is_open_drain (struct gpiod_line ∗line) GPIOD_API

    Check if the line is an open-drain GPIO.
- bool gpiod_line_is_open_source (struct gpiod_line ∗line) GPIOD_API

    Check if the line is an open-source GPIO.
- int gpiod_line_update (struct gpiod_line ∗line) GPIOD_API

    Re-read the line info.
- bool gpiod_line_needs_update (struct gpiod_line ∗line) GPIOD_API GPIOD_DEPRECATED

    Check if the line info needs to be updated.
- int gpiod_line_request (struct gpiod_line ∗line, const struct gpiod_line_request_config ∗config, int default_val) GPIOD_API

    Reserve a single line.
- int gpiod_line_request_input (struct gpiod_line ∗line, const char ∗consumer) GPIOD_API

    Reserve a single line, set the direction to input.
- int gpiod_line_request_output (struct gpiod_line ∗line, const char ∗consumer, int default_val) GPIOD_API

    Reserve a single line, set the direction to output.
- int gpiod_line_request_rising_edge_events (struct gpiod_line ∗line, const char ∗consumer) GPIOD_API

    Request rising edge event notifications on a single line.
- int gpiod_line_request_falling_edge_events (struct gpiod_line ∗line, const char ∗consumer) GPIOD_API

    Request falling edge event notifications on a single line.
- int gpiod_line_request_both_edges_events (struct gpiod_line ∗line, const char ∗consumer) GPIOD_API

    Request all event type notifications on a single line.
- int gpiod_line_request_input_flags (struct gpiod_line ∗line, const char ∗consumer, int flags) GPIOD_API

    Reserve a single line, set the direction to input.
- int gpiod_line_request_output_flags (struct gpiod_line ∗line, const char ∗consumer, int flags, int default_val) GPIOD_API

    Reserve a single line, set the direction to output.

- int gpiod_line_request_rising_edge_events_flags (struct gpiod_line ∗line, const char ∗consumer, int flags) GPIOD_API

  Request rising edge event notifications on a single line.
- int gpiod_line_request_falling_edge_events_flags (struct gpiod_line ∗line, const char ∗consumer, int flags) GPIOD_API

  Request falling edge event notifications on a single line.
- int gpiod_line_request_both_edges_events_flags (struct gpiod_line ∗line, const char ∗consumer, int flags) GPIOD_API

  Request all event type notifications on a single line.
- int gpiod_line_request_bulk (struct gpiod_line_bulk ∗bulk, const struct gpiod_line_request_config ∗config, const int ∗default_vals) GPIOD_API

  Reserve a set of GPIO lines.
- int gpiod_line_request_bulk_input (struct gpiod_line_bulk ∗bulk, const char ∗consumer) GPIOD_API

  Reserve a set of GPIO lines, set the direction to input.
- int gpiod_line_request_bulk_output (struct gpiod_line_bulk ∗bulk, const char ∗consumer, const int ∗default←_vals) GPIOD_API

  Reserve a set of GPIO lines, set the direction to output.
- int gpiod_line_request_bulk_rising_edge_events (struct gpiod_line_bulk ∗bulk, const char ∗consumer) GPIOD_API

  Request rising edge event notifications on a set of lines.
- int gpiod_line_request_bulk_falling_edge_events (struct gpiod_line_bulk ∗bulk, const char ∗consumer) GPIOD_API

  Request falling edge event notifications on a set of lines.
- int gpiod_line_request_bulk_both_edges_events (struct gpiod_line_bulk ∗bulk, const char ∗consumer) GPIOD_API

  Request all event type notifications on a set of lines.
- int gpiod_line_request_bulk_input_flags (struct gpiod_line_bulk ∗bulk, const char ∗consumer, int flags) GPIOD_API

  Reserve a set of GPIO lines, set the direction to input.
- int gpiod_line_request_bulk_output_flags (struct gpiod_line_bulk ∗bulk, const char ∗consumer, int flags, const int ∗default_vals) GPIOD_API

  Reserve a set of GPIO lines, set the direction to output.
- int gpiod_line_request_bulk_rising_edge_events_flags (struct gpiod_line_bulk ∗bulk, const char ∗consumer, int flags) GPIOD_API

  Request rising edge event notifications on a set of lines.
- int gpiod_line_request_bulk_falling_edge_events_flags (struct gpiod_line_bulk ∗bulk, const char ∗consumer, int flags) GPIOD_API

  Request falling edge event notifications on a set of lines.
- int gpiod_line_request_bulk_both_edges_events_flags (struct gpiod_line_bulk ∗bulk, const char ∗consumer, int flags) GPIOD_API

  Request all event type notifications on a set of lines.
- void gpiod_line_release (struct gpiod_line ∗line) GPIOD_API

  Release a previously reserved line.
- void gpiod_line_release_bulk (struct gpiod_line_bulk ∗bulk) GPIOD_API

  Release a set of previously reserved lines.
- bool gpiod_line_is_requested (struct gpiod_line ∗line) GPIOD_API

  Check if the calling user has ownership of this line.
- bool gpiod_line_is_free (struct gpiod_line ∗line) GPIOD_API

  Check if the calling user has neither requested ownership of this line nor configured any event notifications.
- int gpiod_line_get_value (struct gpiod_line ∗line) GPIOD_API

  Read current value of a single GPIO line.
- int gpiod_line_get_value_bulk (struct gpiod_line_bulk ∗bulk, int ∗values) GPIOD_API

Read current values of a set of GPIO lines.
- int gpiod_line_set_value (struct gpiod_line ∗line, int value) GPIOD_API

  Set the value of a single GPIO line.
- int gpiod_line_set_value_bulk (struct gpiod_line_bulk ∗bulk, const int ∗values) GPIOD_API

  Set the values of a set of GPIO lines.
- int gpiod_line_set_config (struct gpiod_line ∗line, int direction, int flags, int value) GPIOD_API

  Update the configuration of a single GPIO line.
- int gpiod_line_set_config_bulk (struct gpiod_line_bulk ∗bulk, int direction, int flags, const int ∗values) GPIOD_API

  Update the configuration of a set of GPIO lines.
- int gpiod_line_set_flags (struct gpiod_line ∗line, int flags) GPIOD_API

  Update the configuration flags of a single GPIO line.
- int gpiod_line_set_flags_bulk (struct gpiod_line_bulk ∗bulk, int flags) GPIOD_API

  Update the configuration flags of a set of GPIO lines.
- int gpiod_line_set_direction_input (struct gpiod_line ∗line) GPIOD_API

  Set the direction of a single GPIO line to input.
- int gpiod_line_set_direction_input_bulk (struct gpiod_line_bulk ∗bulk) GPIOD_API

  Set the direction of a set of GPIO lines to input.
- int gpiod_line_set_direction_output (struct gpiod_line ∗line, int value) GPIOD_API

  Set the direction of a single GPIO line to output.
- int gpiod_line_set_direction_output_bulk (struct gpiod_line_bulk ∗bulk, const int ∗values) GPIOD_API

  Set the direction of a set of GPIO lines to output.
- int gpiod_line_event_wait (struct gpiod_line ∗line, const struct timespec ∗timeout) GPIOD_API

  Wait for an event on a single line.
- int gpiod_line_event_wait_bulk (struct gpiod_line_bulk ∗bulk, const struct timespec ∗timeout, struct gpiod_line_bulk ∗event_bulk) GPIOD_API

  Wait for events on a set of lines.
- int gpiod_line_event_read (struct gpiod_line ∗line, struct gpiod_line_event ∗event) GPIOD_API

  Read next pending event from the GPIO line.
- int gpiod_line_event_read_multiple (struct gpiod_line ∗line, struct gpiod_line_event ∗events, unsigned int num_events) GPIOD_API

  Read up to a certain number of events from the GPIO line.
- int gpiod_line_event_get_fd (struct gpiod_line ∗line) GPIOD_API

  Get the event file descriptor.
- int gpiod_line_event_read_fd (int fd, struct gpiod_line_event ∗event) GPIOD_API

  Read the last GPIO event directly from a file descriptor.
- int gpiod_line_event_read_fd_multiple (int fd, struct gpiod_line_event ∗events, unsigned int num_events) GPIOD_API

  Read up to a certain number of events directly from a file descriptor.
- struct gpiod_line ∗ gpiod_line_get (const char ∗device, unsigned int offset) GPIOD_API

  Get a GPIO line handle by GPIO chip description and offset.
- struct gpiod_line ∗ gpiod_line_find (const char ∗name) GPIOD_API

  Find a GPIO line by its name.
- void gpiod_line_close_chip (struct gpiod_line ∗line) GPIOD_API

  Close a GPIO chip owning this line and release all resources.
- struct gpiod_chip ∗ gpiod_line_get_chip (struct gpiod_line ∗line) GPIOD_API

  Get the handle to the GPIO chip controlling this line.
- struct gpiod_chip_iter ∗ gpiod_chip_iter_new (void) GPIOD_API

  Create a new gpiochip iterator.
- void gpiod_chip_iter_free (struct gpiod_chip_iter ∗iter) GPIOD_API

  Release all resources allocated for the gpiochip iterator and close the most recently opened gpiochip (if any).

- void gpiod_chip_iter_free_noclose (struct gpiod_chip_iter ∗iter) GPIOD_API

  Release all resources allocated for the gpiochip iterator but don't close the most recently opened gpiochip (if any).
- struct gpiod_chip ∗ gpiod_chip_iter_next (struct gpiod_chip_iter ∗iter) GPIOD_API

  Get the next gpiochip handle.
- struct gpiod_chip ∗ gpiod_chip_iter_next_noclose (struct gpiod_chip_iter ∗iter) GPIOD_API

  Get the next gpiochip handle without closing the previous one.
- struct gpiod_line_iter ∗ gpiod_line_iter_new (struct gpiod_chip ∗chip) GPIOD_API

  Create a new line iterator.
- void gpiod_line_iter_free (struct gpiod_line_iter ∗iter) GPIOD_API

  Free all resources associated with a GPIO line iterator.
- struct gpiod_line ∗ gpiod_line_iter_next (struct gpiod_line_iter ∗iter) GPIOD_API

  Get the next GPIO line handle.
- const char ∗ gpiod_version_string (void) GPIOD_API

  Get the API version of the library as a human-readable string.

8.2 gpiod.h

Go to the documentation of this file.

```
/* SPDX-License-Identifier:  LGPL-2.1-or-later */
/*
 * This file is part of libgpiod.
 *
 * Copyright (C) 2017-2018 Bartosz Golaszewski <bartekgola@gmail.com>
 */

#ifndef __LIBGPIOD_GPIOD_H__
#define __LIBGPIOD_GPIOD_H__

#include <stdbool.h>
#include <stdlib.h>
#include <time.h>

#ifdef __cplusplus
extern "C" {
#endif

struct gpiod_chip;
struct gpiod_line;
struct gpiod_chip_iter;
struct gpiod_line_iter;
struct gpiod_line_bulk;

#define GPIOD_API       __attribute__((visibility("default")))

#define GPIOD_UNUSED        __attribute__((unused))

#define GPIOD_BIT(nr)       (1UL « (nr))

#define GPIOD_DEPRECATED    __attribute__((deprecated))

enum {
    GPIOD_CTXLESS_FLAG_OPEN_DRAIN      = GPIOD_BIT(0),
    GPIOD_CTXLESS_FLAG_OPEN_SOURCE     = GPIOD_BIT(1),
    GPIOD_CTXLESS_FLAG_BIAS_DISABLE    = GPIOD_BIT(2),
    GPIOD_CTXLESS_FLAG_BIAS_PULL_DOWN  = GPIOD_BIT(3),
    GPIOD_CTXLESS_FLAG_BIAS_PULL_UP    = GPIOD_BIT(4),
};

int gpiod_ctxless_get_value(const char *device, unsigned int offset,
                bool active_low, const char *consumer) GPIOD_API;

int gpiod_ctxless_get_value_ext(const char *device, unsigned int offset,
                bool active_low, const char *consumer,
                int flags) GPIOD_API;

int gpiod_ctxless_get_value_multiple(const char *device,
                     const unsigned int *offsets, int *values,
                     unsigned int num_lines, bool active_low,
                     const char *consumer) GPIOD_API;

int gpiod_ctxless_get_value_multiple_ext(const char *device,
                     const unsigned int *offsets,
                     int *values, unsigned int num_lines,
                     bool active_low, const char *consumer,
                     int flags) GPIOD_API;

typedef void (*gpiod_ctxless_set_value_cb)(void *);

int gpiod_ctxless_set_value(const char *device, unsigned int offset, int value,
                bool active_low, const char *consumer,
                gpiod_ctxless_set_value_cb cb,
                void *data) GPIOD_API;

int gpiod_ctxless_set_value_ext(const char *device, unsigned int offset,
                int value, bool active_low,
                const char *consumer,
                gpiod_ctxless_set_value_cb cb,
                void *data, int flags) GPIOD_API;

int gpiod_ctxless_set_value_multiple(const char *device,
                     const unsigned int *offsets,
                     const int *values, unsigned int num_lines,
                     bool active_low, const char *consumer,
                     gpiod_ctxless_set_value_cb cb,
                     void *data) GPIOD_API;

int gpiod_ctxless_set_value_multiple_ext(const char *device,
                     const unsigned int *offsets,
                     const int *values,
                     unsigned int num_lines,
                     bool active_low,
                     const char *consumer,
                     gpiod_ctxless_set_value_cb cb,
                     void *data, int flags) GPIOD_API;

enum {
    GPIOD_CTXLESS_EVENT_RISING_EDGE = 1,
    GPIOD_CTXLESS_EVENT_FALLING_EDGE,
    GPIOD_CTXLESS_EVENT_BOTH_EDGES,
};

enum {
    GPIOD_CTXLESS_EVENT_CB_TIMEOUT = 1,
    GPIOD_CTXLESS_EVENT_CB_RISING_EDGE,
    GPIOD_CTXLESS_EVENT_CB_FALLING_EDGE,
};

enum {
    GPIOD_CTXLESS_EVENT_CB_RET_ERR = -1,
    GPIOD_CTXLESS_EVENT_CB_RET_OK = 0,
    GPIOD_CTXLESS_EVENT_CB_RET_STOP = 1,
};

typedef int (*gpiod_ctxless_event_handle_cb)(int, unsigned int,
                         const struct timespec *, void *);

enum {
    GPIOD_CTXLESS_EVENT_POLL_RET_STOP = -2,
    GPIOD_CTXLESS_EVENT_POLL_RET_ERR = -1,
    GPIOD_CTXLESS_EVENT_POLL_RET_TIMEOUT = 0,
};

struct gpiod_ctxless_event_poll_fd {
    int fd;
    bool event;
};

typedef int (*gpiod_ctxless_event_poll_cb)(unsigned int,
                struct gpiod_ctxless_event_poll_fd *,
                const struct timespec *, void *);

int gpiod_ctxless_event_loop(const char *device, unsigned int offset,
                 bool active_low, const char *consumer,
                 const struct timespec *timeout,
                 gpiod_ctxless_event_poll_cb poll_cb,
                 gpiod_ctxless_event_handle_cb event_cb,
                 void *data) GPIOD_API GPIOD_DEPRECATED;

int gpiod_ctxless_event_loop_multiple(const char *device,
                      const unsigned int *offsets,
                      unsigned int num_lines, bool active_low,
                      const char *consumer,
                      const struct timespec *timeout,
                      gpiod_ctxless_event_poll_cb poll_cb,
                      gpiod_ctxless_event_handle_cb event_cb,
                      void *data) GPIOD_API GPIOD_DEPRECATED;

int gpiod_ctxless_event_monitor(const char *device, int event_type,
                unsigned int offset, bool active_low,
                const char *consumer,
                const struct timespec *timeout,
                gpiod_ctxless_event_poll_cb poll_cb,
                gpiod_ctxless_event_handle_cb event_cb,
                void *data) GPIOD_API;

int gpiod_ctxless_event_monitor_ext(const char *device, int event_type,
                    unsigned int offset, bool active_low,
                    const char *consumer,
                    const struct timespec *timeout,
                    gpiod_ctxless_event_poll_cb poll_cb,
                    gpiod_ctxless_event_handle_cb event_cb,
                    void *data, int flags) GPIOD_API;

int gpiod_ctxless_event_monitor_multiple(
            const char *device, int event_type,
            const unsigned int *offsets,
            unsigned int num_lines, bool active_low,
            const char *consumer, const struct timespec *timeout,
            gpiod_ctxless_event_poll_cb poll_cb,
            gpiod_ctxless_event_handle_cb event_cb,
            void *data) GPIOD_API;

int gpiod_ctxless_event_monitor_multiple_ext(
            const char *device, int event_type,
            const unsigned int *offsets,
            unsigned int num_lines, bool active_low,
            const char *consumer, const struct timespec *timeout,
            gpiod_ctxless_event_poll_cb poll_cb,
            gpiod_ctxless_event_handle_cb event_cb,
            void *data, int flags) GPIOD_API;


int gpiod_ctxless_find_line(const char *name, char *chipname,
                size_t chipname_size,
                unsigned int *offset) GPIOD_API;

struct gpiod_chip *gpiod_chip_open(const char *path) GPIOD_API;

struct gpiod_chip *gpiod_chip_open_by_name(const char *name) GPIOD_API;

struct gpiod_chip *gpiod_chip_open_by_number(unsigned int num) GPIOD_API;

struct gpiod_chip *gpiod_chip_open_by_label(const char *label) GPIOD_API;

struct gpiod_chip *gpiod_chip_open_lookup(const char *descr) GPIOD_API;

void gpiod_chip_close(struct gpiod_chip *chip) GPIOD_API;

const char *gpiod_chip_name(struct gpiod_chip *chip) GPIOD_API;

const char *gpiod_chip_label(struct gpiod_chip *chip) GPIOD_API;

unsigned int gpiod_chip_num_lines(struct gpiod_chip *chip) GPIOD_API;

struct gpiod_line *
gpiod_chip_get_line(struct gpiod_chip *chip, unsigned int offset) GPIOD_API;

int gpiod_chip_get_lines(struct gpiod_chip *chip,
             unsigned int *offsets, unsigned int num_offsets,
             struct gpiod_line_bulk *bulk) GPIOD_API;

int gpiod_chip_get_all_lines(struct gpiod_chip *chip,
                 struct gpiod_line_bulk *bulk) GPIOD_API;

struct gpiod_line *
gpiod_chip_find_line(struct gpiod_chip *chip, const char *name) GPIOD_API;

int gpiod_chip_find_lines(struct gpiod_chip *chip, const char **names,
              struct gpiod_line_bulk *bulk) GPIOD_API;

#define GPIOD_LINE_BULK_MAX_LINES   64

struct gpiod_line_bulk {
    struct gpiod_line *lines[GPIOD_LINE_BULK_MAX_LINES];
    unsigned int num_lines;
};

#define GPIOD_LINE_BULK_INITIALIZER { { NULL }, 0 }

static inline void gpiod_line_bulk_init(struct gpiod_line_bulk *bulk)
{
    bulk->num_lines = 0;
}

static inline void gpiod_line_bulk_add(struct gpiod_line_bulk *bulk,
                       struct gpiod_line *line)
```

```c
{
    bulk->lines[bulk->num_lines++] = line;
}

static inline struct gpiod_line *
gpiod_line_bulk_get_line(struct gpiod_line_bulk *bulk, unsigned int offset)
{
    return bulk->lines[offset];
}

static inline unsigned int
gpiod_line_bulk_num_lines(struct gpiod_line_bulk *bulk)
{
    return bulk->num_lines;
}

#define gpiod_line_bulk_foreach_line(bulk, line, lineptr)       \
for ((lineptr) = (bulk)->lines, (line) = *(lineptr);        \
(lineptr) <= (bulk)->lines + ((bulk)->num_lines - 1);  \
(lineptr)++, (line) = *(lineptr))

#define gpiod_line_bulk_foreach_line_off(bulk, line, offset)        \
for ((offset) = 0, (line) = (bulk)->lines[0];           \
(offset) < (bulk)->num_lines;               \
(offset)++, (line) = (bulk)->lines[(offset)])

enum {
    GPIOD_LINE_DIRECTION_INPUT = 1,
    GPIOD_LINE_DIRECTION_OUTPUT,
};

enum {
    GPIOD_LINE_ACTIVE_STATE_HIGH = 1,
    GPIOD_LINE_ACTIVE_STATE_LOW,
};

enum {
    GPIOD_LINE_BIAS_AS_IS = 1,
    GPIOD_LINE_BIAS_DISABLE,
    GPIOD_LINE_BIAS_PULL_UP,
    GPIOD_LINE_BIAS_PULL_DOWN,
};

unsigned int gpiod_line_offset(struct gpiod_line *line) GPIOD_API;

const char *gpiod_line_name(struct gpiod_line *line) GPIOD_API;

const char *gpiod_line_consumer(struct gpiod_line *line) GPIOD_API;

int gpiod_line_direction(struct gpiod_line *line) GPIOD_API;

int gpiod_line_active_state(struct gpiod_line *line) GPIOD_API;

int gpiod_line_bias(struct gpiod_line *line) GPIOD_API;

bool gpiod_line_is_used(struct gpiod_line *line) GPIOD_API;

bool gpiod_line_is_open_drain(struct gpiod_line *line) GPIOD_API;

bool gpiod_line_is_open_source(struct gpiod_line *line) GPIOD_API;

int gpiod_line_update(struct gpiod_line *line) GPIOD_API;

bool
gpiod_line_needs_update(struct gpiod_line *line) GPIOD_API GPIOD_DEPRECATED;

enum {
    GPIOD_LINE_REQUEST_DIRECTION_AS_IS = 1,
    GPIOD_LINE_REQUEST_DIRECTION_INPUT,
    GPIOD_LINE_REQUEST_DIRECTION_OUTPUT,
    GPIOD_LINE_REQUEST_EVENT_FALLING_EDGE,
    GPIOD_LINE_REQUEST_EVENT_RISING_EDGE,
    GPIOD_LINE_REQUEST_EVENT_BOTH_EDGES,
};

enum {
    GPIOD_LINE_REQUEST_FLAG_OPEN_DRAIN    = GPIOD_BIT(0),
    GPIOD_LINE_REQUEST_FLAG_OPEN_SOURCE  = GPIOD_BIT(1),
    GPIOD_LINE_REQUEST_FLAG_ACTIVE_LOW    = GPIOD_BIT(2),
    GPIOD_LINE_REQUEST_FLAG_BIAS_DISABLE    = GPIOD_BIT(3),
    GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_DOWN    = GPIOD_BIT(4),
    GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_UP    = GPIOD_BIT(5),
};

struct gpiod_line_request_config {
    const char *consumer;
    int request_type;
    int flags;
};

int gpiod_line_request(struct gpiod_line *line,
                const struct gpiod_line_request_config *config,
                int default_val) GPIOD_API;

int gpiod_line_request_input(struct gpiod_line *line,
                  const char *consumer) GPIOD_API;

int gpiod_line_request_output(struct gpiod_line *line,
                   const char *consumer, int default_val) GPIOD_API;

int gpiod_line_request_rising_edge_events(struct gpiod_line *line,
                       const char *consumer) GPIOD_API;

int gpiod_line_request_falling_edge_events(struct gpiod_line *line,
                        const char *consumer) GPIOD_API;

int gpiod_line_request_both_edges_events(struct gpiod_line *line,
                      const char *consumer) GPIOD_API;

int gpiod_line_request_input_flags(struct gpiod_line *line,
                    const char *consumer, int flags) GPIOD_API;

int gpiod_line_request_output_flags(struct gpiod_line *line,
                     const char *consumer, int flags,
                     int default_val) GPIOD_API;

int gpiod_line_request_rising_edge_events_flags(struct gpiod_line *line,
                         const char *consumer,
                         int flags) GPIOD_API;

int gpiod_line_request_falling_edge_events_flags(struct gpiod_line *line,
                          const char *consumer,
                          int flags) GPIOD_API;

int gpiod_line_request_both_edges_events_flags(struct gpiod_line *line,
                        const char *consumer,
                        int flags) GPIOD_API;

int gpiod_line_request_bulk(struct gpiod_line_bulk *bulk,
                const struct gpiod_line_request_config *config,
                const int *default_vals) GPIOD_API;

int gpiod_line_request_bulk_input(struct gpiod_line_bulk *bulk,
                   const char *consumer) GPIOD_API;

int gpiod_line_request_bulk_output(struct gpiod_line_bulk *bulk,
                    const char *consumer,
                    const int *default_vals) GPIOD_API;

int gpiod_line_request_bulk_rising_edge_events(struct gpiod_line_bulk *bulk,
                        const char *consumer) GPIOD_API;

int gpiod_line_request_bulk_falling_edge_events(struct gpiod_line_bulk *bulk,
                         const char *consumer) GPIOD_API;

int gpiod_line_request_bulk_both_edges_events(struct gpiod_line_bulk *bulk,
                          const char *consumer) GPIOD_API;

int gpiod_line_request_bulk_input_flags(struct gpiod_line_bulk *bulk,
                     const char *consumer,
                     int flags) GPIOD_API;

int gpiod_line_request_bulk_output_flags(struct gpiod_line_bulk *bulk,
                      const char *consumer, int flags,
                      const int *default_vals) GPIOD_API;

int gpiod_line_request_bulk_rising_edge_events_flags(
                    struct gpiod_line_bulk *bulk,
                    const char *consumer,
                    int flags) GPIOD_API;

int gpiod_line_request_bulk_falling_edge_events_flags(
                    struct gpiod_line_bulk *bulk,
                    const char *consumer,
                    int flags) GPIOD_API;

int gpiod_line_request_bulk_both_edges_events_flags(
                    struct gpiod_line_bulk *bulk,
                    const char *consumer,
                    int flags) GPIOD_API;

void gpiod_line_release(struct gpiod_line *line) GPIOD_API;

void gpiod_line_release_bulk(struct gpiod_line_bulk *bulk) GPIOD_API;
```

```c

bool gpiod_line_is_requested(struct gpiod_line *line) GPIOD_API;

bool gpiod_line_is_free(struct gpiod_line *line) GPIOD_API;

int gpiod_line_get_value(struct gpiod_line *line) GPIOD_API;

int gpiod_line_get_value_bulk(struct gpiod_line_bulk *bulk,
                  int *values) GPIOD_API;

int gpiod_line_set_value(struct gpiod_line *line, int value) GPIOD_API;

int gpiod_line_set_value_bulk(struct gpiod_line_bulk *bulk,
                  const int *values) GPIOD_API;

int gpiod_line_set_config(struct gpiod_line *line, int direction,
              int flags, int value) GPIOD_API;

int gpiod_line_set_config_bulk(struct gpiod_line_bulk *bulk,
                   int direction, int flags,
                   const int *values) GPIOD_API;


int gpiod_line_set_flags(struct gpiod_line *line, int flags) GPIOD_API;

int gpiod_line_set_flags_bulk(struct gpiod_line_bulk *bulk,
                  int flags) GPIOD_API;

int gpiod_line_set_direction_input(struct gpiod_line *line) GPIOD_API;

int
gpiod_line_set_direction_input_bulk(struct gpiod_line_bulk *bulk) GPIOD_API;

int gpiod_line_set_direction_output(struct gpiod_line *line,
                    int value) GPIOD_API;

int gpiod_line_set_direction_output_bulk(struct gpiod_line_bulk *bulk,
                     const int *values) GPIOD_API;

enum {
    GPIOD_LINE_EVENT_RISING_EDGE = 1,
    GPIOD_LINE_EVENT_FALLING_EDGE,
};

struct gpiod_line_event {
    struct timespec ts;
    int event_type;
};

int gpiod_line_event_wait(struct gpiod_line *line,
              const struct timespec *timeout) GPIOD_API;

int gpiod_line_event_wait_bulk(struct gpiod_line_bulk *bulk,
                   const struct timespec *timeout,
                   struct gpiod_line_bulk *event_bulk) GPIOD_API;

int gpiod_line_event_read(struct gpiod_line *line,
              struct gpiod_line_event *event) GPIOD_API;

int gpiod_line_event_read_multiple(struct gpiod_line *line,
                   struct gpiod_line_event *events,
                   unsigned int num_events) GPIOD_API;

int gpiod_line_event_get_fd(struct gpiod_line *line) GPIOD_API;

int gpiod_line_event_read_fd(int fd, struct gpiod_line_event *event) GPIOD_API;

int gpiod_line_event_read_fd_multiple(int fd, struct gpiod_line_event *events,
                      unsigned int num_events) GPIOD_API;

struct gpiod_line *
gpiod_line_get(const char *device, unsigned int offset) GPIOD_API;

struct gpiod_line *gpiod_line_find(const char *name) GPIOD_API;

void gpiod_line_close_chip(struct gpiod_line *line) GPIOD_API;

struct gpiod_chip *gpiod_line_get_chip(struct gpiod_line *line) GPIOD_API;

struct gpiod_chip_iter *gpiod_chip_iter_new(void) GPIOD_API;

void gpiod_chip_iter_free(struct gpiod_chip_iter *iter) GPIOD_API;

void gpiod_chip_iter_free_noclose(struct gpiod_chip_iter *iter) GPIOD_API;

struct gpiod_chip *
gpiod_chip_iter_next(struct gpiod_chip_iter *iter) GPIOD_API;
```

```

struct gpiod_chip *
gpiod_chip_iter_next_noclose(struct gpiod_chip_iter *iter) GPIOD_API;

#define gpiod_foreach_chip(iter, chip)                      \
for ((chip) = gpiod_chip_iter_next(iter);               \
(chip);                            \
(chip) = gpiod_chip_iter_next(iter))

#define gpiod_foreach_chip_noclose(iter, chip)                  \
for ((chip) = gpiod_chip_iter_next_noclose(iter);       \
(chip);                            \
(chip) = gpiod_chip_iter_next_noclose(iter))

struct gpiod_line_iter *
gpiod_line_iter_new(struct gpiod_chip *chip) GPIOD_API;

void gpiod_line_iter_free(struct gpiod_line_iter *iter) GPIOD_API;

struct gpiod_line *
gpiod_line_iter_next(struct gpiod_line_iter *iter) GPIOD_API;

#define gpiod_foreach_line(iter, line)                      \
for ((line) = gpiod_line_iter_next(iter);               \
(line);                            \
(line) = gpiod_line_iter_next(iter))

const char *gpiod_version_string(void) GPIOD_API;

#ifdef __cplusplus
} /* extern "C" */
#endif

#endif /* __LIBGPIOD_GPIOD_H__ */
```

Index