

# libgpod

## 2.2.1

Generated by Doxygen 1.12.0



<b>1 Topic Index</b>	<b>1</b>
1.1 Topics	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Topic Documentation</b>	<b>7</b>
4.1 GPIO chips	7
4.1.1 Detailed Description	7
4.1.2 Function Documentation	8
4.1.2.1 gpiod_chip_close()	8
4.1.2.2 gpiod_chip_get_fd()	8
4.1.2.3 gpiod_chip_get_info()	8
4.1.2.4 gpiod_chip_get_line_info()	8
4.1.2.5 gpiod_chip_get_line_offset_from_name()	9
4.1.2.6 gpiod_chip_get_path()	9
4.1.2.7 gpiod_chip_open()	9
4.1.2.8 gpiod_chip_read_info_event()	10
4.1.2.9 gpiod_chip_request_lines()	10
4.1.2.10 gpiod_chip_unwatch_line_info()	10
4.1.2.11 gpiod_chip_wait_info_event()	11
4.1.2.12 gpiod_chip_watch_line_info()	11
4.2 Chip info	12
4.2.1 Detailed Description	12
4.2.2 Function Documentation	12
4.2.2.1 gpiod_chip_info_free()	12
4.2.2.2 gpiod_chip_info_get_label()	12
4.2.2.3 gpiod_chip_info_get_name()	13
4.2.2.4 gpiod_chip_info_get_num_lines()	13
4.3 Line definitions	13
4.3.1 Detailed Description	14
4.3.2 Enumeration Type Documentation	14
4.3.2.1 gpiod_line_bias	14
4.3.2.2 gpiod_line_clock	14
4.3.2.3 gpiod_line_direction	14
4.3.2.4 gpiod_line_drive	14
4.3.2.5 gpiod_line_edge	15
4.3.2.6 gpiod_line_value	15
4.4 Line info	15
4.4.1 Detailed Description	16

4.4.2 Function Documentation	16
4.4.2.1 <code>gpiod_line_info_copy()</code>	16
4.4.2.2 <code>gpiod_line_info_free()</code>	16
4.4.2.3 <code>gpiod_line_info_get_bias()</code>	17
4.4.2.4 <code>gpiod_line_info_get_consumer()</code>	17
4.4.2.5 <code>gpiod_line_info_get_debounce_period_us()</code>	17
4.4.2.6 <code>gpiod_line_info_get_direction()</code>	18
4.4.2.7 <code>gpiod_line_info_get_drive()</code>	19
4.4.2.8 <code>gpiod_line_info_get_edge_detection()</code>	19
4.4.2.9 <code>gpiod_line_info_get_event_clock()</code>	19
4.4.2.10 <code>gpiod_line_info_get_name()</code>	20
4.4.2.11 <code>gpiod_line_info_get_offset()</code>	20
4.4.2.12 <code>gpiod_line_info_is_active_low()</code>	20
4.4.2.13 <code>gpiod_line_info_is_debounced()</code>	21
4.4.2.14 <code>gpiod_line_info_is_used()</code>	21
4.5 Line status watch events	21
4.5.1 Detailed Description	22
4.5.2 Enumeration Type Documentation	22
4.5.2.1 <code>gpiod_info_event_type</code>	22
4.5.3 Function Documentation	22
4.5.3.1 <code>gpiod_info_event_free()</code>	22
4.5.3.2 <code>gpiod_info_event_get_event_type()</code>	22
4.5.3.3 <code>gpiod_info_event_get_line_info()</code>	23
4.5.3.4 <code>gpiod_info_event_get_timestamp_ns()</code>	23
4.6 Line settings objects	24
4.6.1 Detailed Description	25
4.6.2 Function Documentation	25
4.6.2.1 <code>gpiod_line_settings_copy()</code>	25
4.6.2.2 <code>gpiod_line_settings_free()</code>	25
4.6.2.3 <code>gpiod_line_settings_get_active_low()</code>	25
4.6.2.4 <code>gpiod_line_settings_get_bias()</code>	25
4.6.2.5 <code>gpiod_line_settings_get_debounce_period_us()</code>	26
4.6.2.6 <code>gpiod_line_settings_get_direction()</code>	26
4.6.2.7 <code>gpiod_line_settings_get_drive()</code>	26
4.6.2.8 <code>gpiod_line_settings_get_edge_detection()</code>	27
4.6.2.9 <code>gpiod_line_settings_get_event_clock()</code>	27
4.6.2.10 <code>gpiod_line_settings_get_output_value()</code>	27
4.6.2.11 <code>gpiod_line_settings_new()</code>	28
4.6.2.12 <code>gpiod_line_settings_reset()</code>	28
4.6.2.13 <code>gpiod_line_settings_set_active_low()</code>	28
4.6.2.14 <code>gpiod_line_settings_set_bias()</code>	28
4.6.2.15 <code>gpiod_line_settings_set_debounce_period_us()</code>	29

4.6.2.16	<code>gpiod_line_settings_set_direction()</code>	29
4.6.2.17	<code>gpiod_line_settings_set_drive()</code>	29
4.6.2.18	<code>gpiod_line_settings_set_edge_detection()</code>	30
4.6.2.19	<code>gpiod_line_settings_set_event_clock()</code>	30
4.6.2.20	<code>gpiod_line_settings_set_output_value()</code>	30
4.7	Line configuration objects	31
4.7.1	Detailed Description	31
4.7.2	Function Documentation	32
4.7.2.1	<code>gpiod_line_config_add_line_settings()</code>	32
4.7.2.2	<code>gpiod_line_config_free()</code>	32
4.7.2.3	<code>gpiod_line_config_get_configured_offsets()</code>	32
4.7.2.4	<code>gpiod_line_config_get_line_settings()</code>	33
4.7.2.5	<code>gpiod_line_config_get_num_configured_offsets()</code>	33
4.7.2.6	<code>gpiod_line_config_new()</code>	33
4.7.2.7	<code>gpiod_line_config_reset()</code>	33
4.7.2.8	<code>gpiod_line_config_set_output_values()</code>	34
4.8	Request configuration objects	34
4.8.1	Detailed Description	35
4.8.2	Function Documentation	35
4.8.2.1	<code>gpiod_request_config_free()</code>	35
4.8.2.2	<code>gpiod_request_config_get_consumer()</code>	35
4.8.2.3	<code>gpiod_request_config_get_event_buffer_size()</code>	35
4.8.2.4	<code>gpiod_request_config_new()</code>	36
4.8.2.5	<code>gpiod_request_config_set_consumer()</code>	36
4.8.2.6	<code>gpiod_request_config_set_event_buffer_size()</code>	36
4.9	Line request operations	37
4.9.1	Detailed Description	37
4.9.2	Function Documentation	37
4.9.2.1	<code>gpiod_line_request_get_chip_name()</code>	37
4.9.2.2	<code>gpiod_line_request_get_fd()</code>	38
4.9.2.3	<code>gpiod_line_request_get_num_requested_lines()</code>	38
4.9.2.4	<code>gpiod_line_request_get_requested_offsets()</code>	38
4.9.2.5	<code>gpiod_line_request_get_value()</code>	39
4.9.2.6	<code>gpiod_line_request_get_values()</code>	39
4.9.2.7	<code>gpiod_line_request_get_values_subset()</code>	40
4.9.2.8	<code>gpiod_line_request_read_edge_events()</code>	40
4.9.2.9	<code>gpiod_line_request_reconfigure_lines()</code>	40
4.9.2.10	<code>gpiod_line_request_release()</code>	41
4.9.2.11	<code>gpiod_line_request_set_value()</code>	41
4.9.2.12	<code>gpiod_line_request_set_values()</code>	41
4.9.2.13	<code>gpiod_line_request_set_values_subset()</code>	42
4.9.2.14	<code>gpiod_line_request_wait_edge_events()</code>	42

4.10 Line edge events handling . . . . .	43
4.10.1 Detailed Description . . . . .	43
4.10.2 Enumeration Type Documentation . . . . .	43
4.10.2.1 <code>gpiod_edge_event_type</code> . . . . .	43
4.10.3 Function Documentation . . . . .	44
4.10.3.1 <code>gpiod_edge_event_buffer_free()</code> . . . . .	44
4.10.3.2 <code>gpiod_edge_event_buffer_get_capacity()</code> . . . . .	44
4.10.3.3 <code>gpiod_edge_event_buffer_get_event()</code> . . . . .	44
4.10.3.4 <code>gpiod_edge_event_buffer_get_num_events()</code> . . . . .	45
4.10.3.5 <code>gpiod_edge_event_buffer_new()</code> . . . . .	45
4.10.3.6 <code>gpiod_edge_event_copy()</code> . . . . .	45
4.10.3.7 <code>gpiod_edge_event_free()</code> . . . . .	46
4.10.3.8 <code>gpiod_edge_event_get_event_type()</code> . . . . .	47
4.10.3.9 <code>gpiod_edge_event_get_global_seqno()</code> . . . . .	47
4.10.3.10 <code>gpiod_edge_event_get_line_offset()</code> . . . . .	47
4.10.3.11 <code>gpiod_edge_event_get_line_seqno()</code> . . . . .	47
4.10.3.12 <code>gpiod_edge_event_get_timestamp_ns()</code> . . . . .	48
4.11 Stuff that didn't fit anywhere else . . . . .	48
4.11.1 Detailed Description . . . . .	48
4.11.2 Function Documentation . . . . .	49
4.11.2.1 <code>gpiod_api_version()</code> . . . . .	49
4.11.2.2 <code>gpiod_is_gpiochip_device()</code> . . . . .	49
<b>5 Class Documentation</b> . . . . .	<b>51</b>
5.1 <code>gpiod_chip</code> Struct Reference . . . . .	51
5.1.1 Detailed Description . . . . .	51
5.2 <code>gpiod_chip_info</code> Struct Reference . . . . .	51
5.2.1 Detailed Description . . . . .	51
5.3 <code>gpiod_edge_event</code> Struct Reference . . . . .	51
5.3.1 Detailed Description . . . . .	52
5.4 <code>gpiod_edge_event_buffer</code> Struct Reference . . . . .	52
5.4.1 Detailed Description . . . . .	52
5.5 <code>gpiod_info_event</code> Struct Reference . . . . .	52
5.5.1 Detailed Description . . . . .	52
5.6 <code>gpiod_line_config</code> Struct Reference . . . . .	52
5.6.1 Detailed Description . . . . .	52
5.7 <code>gpiod_line_info</code> Struct Reference . . . . .	53
5.7.1 Detailed Description . . . . .	53
5.8 <code>gpiod_line_request</code> Struct Reference . . . . .	53
5.8.1 Detailed Description . . . . .	53
5.9 <code>gpiod_line_settings</code> Struct Reference . . . . .	53
5.9.1 Detailed Description . . . . .	53

---

5.10 gpod_request_config Struct Reference . . . . .	53
5.10.1 Detailed Description . . . . .	53
<b>6 File Documentation</b>	<b>55</b>
6.1 gpod.h File Reference . . . . .	55
6.2 gpod.h . . . . .	60
<b>Index</b>	<b>65</b>





# Chapter 1

## Topic Index

### 1.1 Topics

Here is a list of all topics with brief descriptions:

GPIO chips . . . . .	7
Chip info . . . . .	12
Line definitions . . . . .	13
Line info . . . . .	15
Line status watch events . . . . .	21
Line settings objects . . . . .	24
Line configuration objects . . . . .	31
Request configuration objects . . . . .	34
Line request operations . . . . .	37
Line edge events handling . . . . .	43
Stuff that didn't fit anywhere else . . . . .	48



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">gpiod_chip</a>	51
<a href="#">gpiod_chip_info</a>	51
<a href="#">gpiod_edge_event</a>	51
<a href="#">gpiod_edge_event_buffer</a>	52
<a href="#">gpiod_info_event</a>	52
<a href="#">gpiod_line_config</a>	52
<a href="#">gpiod_line_info</a>	53
<a href="#">gpiod_line_request</a>	53
<a href="#">gpiod_line_settings</a>	53
<a href="#">gpiod_request_config</a>	53



## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">gpiod.h</a> . . . . .	55
-----------------------------------	----



# Chapter 4

## Topic Documentation

### 4.1 GPIO chips

#### Functions

- struct `gpiod_chip` \* `gpiod_chip_open` (const char \*path)  
*Open a chip by path.*
- void `gpiod_chip_close` (struct `gpiod_chip` \*chip)  
*Close the chip and release all associated resources.*
- struct `gpiod_chip_info` \* `gpiod_chip_get_info` (struct `gpiod_chip` \*chip)  
*Get information about the chip.*
- const char \* `gpiod_chip_get_path` (struct `gpiod_chip` \*chip)  
*Get the path used to open the chip.*
- struct `gpiod_line_info` \* `gpiod_chip_get_line_info` (struct `gpiod_chip` \*chip, unsigned int offset)  
*Get a snapshot of information about a line.*
- struct `gpiod_line_info` \* `gpiod_chip_watch_line_info` (struct `gpiod_chip` \*chip, unsigned int offset)  
*Get a snapshot of the status of a line and start watching it for future changes.*
- int `gpiod_chip_unwatch_line_info` (struct `gpiod_chip` \*chip, unsigned int offset)  
*Stop watching a line for status changes.*
- int `gpiod_chip_get_fd` (struct `gpiod_chip` \*chip)  
*Get the file descriptor associated with the chip.*
- int `gpiod_chip_wait_info_event` (struct `gpiod_chip` \*chip, int64\_t timeout\_ns)  
*Wait for line status change events on any of the watched lines on the chip.*
- struct `gpiod_info_event` \* `gpiod_chip_read_info_event` (struct `gpiod_chip` \*chip)  
*Read a single line status change event from the chip.*
- int `gpiod_chip_get_line_offset_from_name` (struct `gpiod_chip` \*chip, const char \*name)  
*Map a line's name to its offset within the chip.*
- struct `gpiod_line_request` \* `gpiod_chip_request_lines` (struct `gpiod_chip` \*chip, struct `gpiod_request_config` \*req\_cfg, struct `gpiod_line_config` \*line\_cfg)  
*Request a set of lines for exclusive usage.*

#### 4.1.1 Detailed Description

Functions and data structures for GPIO chip operations.

A GPIO chip object is associated with an open file descriptor to the GPIO character device. It exposes basic information about the chip and allows callers to retrieve information about each line, watch lines for state changes and make line requests.

## 4.1.2 Function Documentation

### 4.1.2.1 `gpiod_chip_close()`

```
void gpiod_chip_close (  
    struct gpiod\_chip * chip)
```

Close the chip and release all associated resources.

#### Parameters

<i>chip</i>	Chip to close.
-------------	----------------

### 4.1.2.2 `gpiod_chip_get_fd()`

```
int gpiod_chip_get_fd (  
    struct gpiod\_chip * chip)
```

Get the file descriptor associated with the chip.

#### Parameters

<i>chip</i>	GPIO chip object.
-------------	-------------------

#### Returns

File descriptor number for the chip.

This function never fails. The returned file descriptor must not be closed by the caller. Call [gpiod\\_chip\\_close](#) to close the file descriptor by closing the chip owning it.

### 4.1.2.3 `gpiod_chip_get_info()`

```
struct gpiod\_chip\_info * gpiod_chip_get_info (  
    struct gpiod\_chip * chip)
```

Get information about the chip.

#### Parameters

<i>chip</i>	GPIO chip object.
-------------	-------------------

#### Returns

New GPIO chip info object or NULL if an error occurred. The returned object must be freed by the caller using [gpiod\\_chip\\_info\\_free](#).

### 4.1.2.4 `gpiod_chip_get_line_info()`

```
struct gpiod\_line\_info * gpiod_chip_get_line_info (  
    struct gpiod\_chip * chip,  
    unsigned int offset)
```

Get a snapshot of information about a line.



## Parameters

<i>chip</i>	GPIO chip object.
<i>offset</i>	The offset of the GPIO line.

## Returns

New GPIO line info object or NULL if an error occurred. The returned object must be freed by the caller using [gpiod\\_line\\_info\\_free](#).

**4.1.2.5 gpiod\_chip\_get\_line\_offset\_from\_name()**

```
int gpiod_chip_get_line_offset_from_name (  
    struct gpiod\_chip * chip,  
    const char * name)
```

Map a line's name to its offset within the chip.

## Parameters

<i>chip</i>	GPIO chip object.
<i>name</i>	Name of the GPIO line to map.

## Returns

Offset of the line within the chip or -1 on error.

## Note

If a line with given name is not exposed by the chip, the function sets `errno` to `ENOENT`.

**4.1.2.6 gpiod\_chip\_get\_path()**

```
const char * gpiod_chip_get_path (  
    struct gpiod\_chip * chip)
```

Get the path used to open the chip.

## Parameters

<i>chip</i>	GPIO chip object.
-------------	-------------------

## Returns

Path to the file passed as argument to [gpiod\\_chip\\_open](#). The returned pointer is valid for the lifetime of the chip object and must not be freed by the caller.

**4.1.2.7 gpiod\_chip\_open()**

```
struct gpiod\_chip * gpiod_chip_open (  
    const char * path)
```

Open a chip by path.

## Parameters

<i>path</i>	Path to the gpiochip device file.
-------------	-----------------------------------

## Returns

GPIO chip object or NULL if an error occurred. The returned object must be closed by the caller using [gpiod\\_chip\\_close](#).

**4.1.2.8 gpiod\_chip\_read\_info\_event()**

```
struct gpiod_info_event * gpiod_chip_read_info_event (
    struct gpiod_chip * chip)
```

Read a single line status change event from the chip.

## Parameters

<i>chip</i>	GPIO chip object.
-------------	-------------------

## Returns

Newly read watch event object or NULL on error. The event must be freed by the caller using [gpiod\\_info\\_event\\_free](#).

## Note

If no events are pending, this function will block.

**4.1.2.9 gpiod\_chip\_request\_lines()**

```
struct gpiod_line_request * gpiod_chip_request_lines (
    struct gpiod_chip * chip,
    struct gpiod_request_config * req_cfg,
    struct gpiod_line_config * line_cfg)
```

Request a set of lines for exclusive usage.

## Parameters

<i>chip</i>	GPIO chip object.
<i>req_cfg</i>	Request config object. Can be NULL for default settings.
<i>line_cfg</i>	Line config object.

## Returns

New line request object or NULL if an error occurred. The request must be released by the caller using [gpiod\\_line\\_request\\_release](#).

**4.1.2.10 gpiod\_chip\_unwatch\_line\_info()**

```
int gpiod_chip_unwatch_line_info (
    struct gpiod_chip * chip,
    unsigned int offset)
```

Stop watching a line for status changes.

## Parameters

<i>chip</i>	GPIO chip object.
<i>offset</i>	The offset of the line to stop watching.

## Returns

0 on success, -1 on failure.

**4.1.2.11 `gpiod_chip_wait_info_event()`**

```
int gpiod_chip_wait_info_event (  
    struct gpiod\_chip * chip,  
    int64_t timeout_ns)
```

Wait for line status change events on any of the watched lines on the chip.

## Parameters

<i>chip</i>	GPIO chip object.
<i>timeout_ns</i>	Wait time limit in nanoseconds. If set to 0, the function returns immediately. If set to a negative number, the function blocks indefinitely until an event becomes available.

## Returns

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

**4.1.2.12 `gpiod_chip_watch_line_info()`**

```
struct gpiod\_line\_info * gpiod_chip_watch_line_info (  
    struct gpiod\_chip * chip,  
    unsigned int offset)
```

Get a snapshot of the status of a line and start watching it for future changes.

## Parameters

<i>chip</i>	GPIO chip object.
<i>offset</i>	The offset of the GPIO line.

## Returns

New GPIO line info object or NULL if an error occurred. The returned object must be freed by the caller using [gpiod\\_line\\_info\\_free](#).

## Note

Line status does not include the line value. To monitor the line value the line must be requested as an input with edge detection set.

## 4.2 Chip info

### Functions

- void `gpiod_chip_info_free` (struct `gpiod_chip_info` \*info)  
*Free a chip info object and release all associated resources.*
- const char \* `gpiod_chip_info_get_name` (struct `gpiod_chip_info` \*info)  
*Get the name of the chip as represented in the kernel.*
- const char \* `gpiod_chip_info_get_label` (struct `gpiod_chip_info` \*info)  
*Get the label of the chip as represented in the kernel.*
- size\_t `gpiod_chip_info_get_num_lines` (struct `gpiod_chip_info` \*info)  
*Get the number of lines exposed by the chip.*

### 4.2.1 Detailed Description

Functions for retrieving kernel information about chips.

Line info object contains an immutable snapshot of a chip's status.

The chip info contains all the publicly available information about a chip.

Some accessor methods return pointers. Those pointers refer to internal fields. The lifetimes of those fields are tied to the lifetime of the containing chip info object. Such pointers remain valid until `gpiod_chip_info_free` is called on the containing chip info object. They must not be freed by the caller.

### 4.2.2 Function Documentation

#### 4.2.2.1 `gpiod_chip_info_free()`

```
void gpiod_chip_info_free (  
    struct gpiod_chip_info * info)
```

Free a chip info object and release all associated resources.

#### Parameters

<i>info</i>	GPIO chip info object to free.
-------------	--------------------------------

#### 4.2.2.2 `gpiod_chip_info_get_label()`

```
const char * gpiod_chip_info_get_label (  
    struct gpiod_chip_info * info)
```

Get the label of the chip as represented in the kernel.

#### Parameters

<i>info</i>	GPIO chip info object.
-------------	------------------------

#### Returns

Valid pointer to a human-readable string containing the chip label. The string lifetime is tied to the chip info object so the pointer must not be freed by the caller.

4.2.2.3 `gpiod_chip_info_get_name()`

```
const char * gpiod_chip_info_get_name (
    struct gpiod_chip_info * info)
```

Get the name of the chip as represented in the kernel.

## Parameters

<i>info</i>	GPIO chip info object.
-------------	------------------------

## Returns

Valid pointer to a human-readable string containing the chip name. The string lifetime is tied to the chip info object so the pointer must not be freed by the caller.

4.2.2.4 `gpiod_chip_info_get_num_lines()`

```
size_t gpiod_chip_info_get_num_lines (
    struct gpiod_chip_info * info)
```

Get the number of lines exposed by the chip.

## Parameters

<i>info</i>	GPIO chip info object.
-------------	------------------------

## Returns

Number of GPIO lines.

## 4.3 Line definitions

## Enumerations

- enum `gpiod_line_value` { `GPIO_LINE_VALUE_ERROR` = -1 , `GPIO_LINE_VALUE_INACTIVE` = 0 , `GPIO_LINE_VALUE_ACTIVE` = 1 }
- Logical line state.*
- enum `gpiod_line_direction` { `GPIO_LINE_DIRECTION_AS_IS` = 1 , `GPIO_LINE_DIRECTION_INPUT` , `GPIO_LINE_DIRECTION_OUTPUT` }
- Direction settings.*
- enum `gpiod_line_edge` { `GPIO_LINE_EDGE_NONE` = 1 , `GPIO_LINE_EDGE_RISING` , `GPIO_LINE_EDGE_FALLING` , `GPIO_LINE_EDGE_BOTH` }
- Edge detection settings.*
- enum `gpiod_line_bias` { `GPIO_LINE_BIAS_AS_IS` = 1 , `GPIO_LINE_BIAS_UNKNOWN` , `GPIO_LINE_BIAS_DISABLED` , `GPIO_LINE_BIAS_PULL_UP` , `GPIO_LINE_BIAS_PULL_DOWN` }
- Internal bias settings.*
- enum `gpiod_line_drive` { `GPIO_LINE_DRIVE_PUSH_PULL` = 1 , `GPIO_LINE_DRIVE_OPEN_DRAIN` , `GPIO_LINE_DRIVE_OPEN_SOURCE` }
- Drive settings.*
- enum `gpiod_line_clock` { `GPIO_LINE_CLOCK_MONOTONIC` = 1 , `GPIO_LINE_CLOCK_REALTIME` , `GPIO_LINE_CLOCK_HTE` }
- Clock settings.*

### 4.3.1 Detailed Description

These defines are used across the API.

### 4.3.2 Enumeration Type Documentation

#### 4.3.2.1 `gpiod_line_bias`

enum `gpiod_line_bias`

Internal bias settings.

Enumerator

<code>GPIOD_LINE_BIAS_AS_IS</code>	Don't change the bias setting when applying line config.
<code>GPIOD_LINE_BIAS_UNKNOWN</code>	The internal bias state is unknown.
<code>GPIOD_LINE_BIAS_DISABLED</code>	The internal bias is disabled.
<code>GPIOD_LINE_BIAS_PULL_UP</code>	The internal pull-up bias is enabled.
<code>GPIOD_LINE_BIAS_PULL_DOWN</code>	The internal pull-down bias is enabled.

#### 4.3.2.2 `gpiod_line_clock`

enum `gpiod_line_clock`

Clock settings.

Enumerator

<code>GPIOD_LINE_CLOCK_MONOTONIC</code>	Line uses the monotonic clock for edge event timestamps.
<code>GPIOD_LINE_CLOCK_REALTIME</code>	Line uses the realtime clock for edge event timestamps.
<code>GPIOD_LINE_CLOCK_HTE</code>	Line uses the hardware timestamp engine for event timestamps.

#### 4.3.2.3 `gpiod_line_direction`

enum `gpiod_line_direction`

Direction settings.

Enumerator

<code>GPIOD_LINE_DIRECTION_AS_IS</code>	Request the line(s), but don't change direction.
<code>GPIOD_LINE_DIRECTION_INPUT</code>	Direction is input - for reading the value of an externally driven GPIO line.
<code>GPIOD_LINE_DIRECTION_OUTPUT</code>	Direction is output - for driving the GPIO line.

#### 4.3.2.4 `gpiod_line_drive`

enum `gpiod_line_drive`

Drive settings.

## Enumerator

GPIOD_LINE_DRIVE_PUSH_PULL	Drive setting is push-pull.
GPIOD_LINE_DRIVE_OPEN_DRAIN	Line output is open-drain.
GPIOD_LINE_DRIVE_OPEN_SOURCE	Line output is open-source.

4.3.2.5 `gpiod_line_edge`

enum `gpiod_line_edge`

Edge detection settings.

## Enumerator

GPIOD_LINE_EDGE_NONE	Line edge detection is disabled.
GPIOD_LINE_EDGE_RISING	Line detects rising edge events.
GPIOD_LINE_EDGE_FALLING	Line detects falling edge events.
GPIOD_LINE_EDGE_BOTH	Line detects both rising and falling edge events.

4.3.2.6 `gpiod_line_value`

enum `gpiod_line_value`

Logical line state.

## Enumerator

GPIOD_LINE_VALUE_ERROR	Returned to indicate an error when reading the value.
GPIOD_LINE_VALUE_INACTIVE	Line is logically inactive.
GPIOD_LINE_VALUE_ACTIVE	Line is logically active.

## 4.4 Line info

## Functions

- void `gpiod_line_info_free` (struct `gpiod_line_info` \*info)  
*Free a line info object and release all associated resources.*
- struct `gpiod_line_info` \* `gpiod_line_info_copy` (struct `gpiod_line_info` \*info)  
*Copy a line info object.*
- unsigned int `gpiod_line_info_get_offset` (struct `gpiod_line_info` \*info)  
*Get the offset of the line.*
- const char \* `gpiod_line_info_get_name` (struct `gpiod_line_info` \*info)  
*Get the name of the line.*
- bool `gpiod_line_info_is_used` (struct `gpiod_line_info` \*info)  
*Check if the line is in use.*

- `const char * gpiod_line_info_get_consumer` (`struct gpiod_line_info *info`)  
*Get the name of the consumer of the line.*
- `enum gpiod_line_direction gpiod_line_info_get_direction` (`struct gpiod_line_info *info`)  
*Get the direction setting of the line.*
- `enum gpiod_line_edge gpiod_line_info_get_edge_detection` (`struct gpiod_line_info *info`)  
*Get the edge detection setting of the line.*
- `enum gpiod_line_bias gpiod_line_info_get_bias` (`struct gpiod_line_info *info`)  
*Get the bias setting of the line.*
- `enum gpiod_line_drive gpiod_line_info_get_drive` (`struct gpiod_line_info *info`)  
*Get the drive setting of the line.*
- `bool gpiod_line_info_is_active_low` (`struct gpiod_line_info *info`)  
*Check if the logical value of the line is inverted compared to the physical.*
- `bool gpiod_line_info_is_debounced` (`struct gpiod_line_info *info`)  
*Check if the line is debounced (either by hardware or by the kernel software debouncer).*
- `unsigned long gpiod_line_info_get_debounce_period_us` (`struct gpiod_line_info *info`)  
*Get the debounce period of the line, in microseconds.*
- `enum gpiod_line_clock gpiod_line_info_get_event_clock` (`struct gpiod_line_info *info`)  
*Get the event clock setting used for edge event timestamps for the line.*

#### 4.4.1 Detailed Description

Functions for retrieving kernel information about both requested and free lines.

Line info object contains an immutable snapshot of a line's status.

The line info contains all the publicly available information about a line, which does not include the line value. The line must be requested to access the line value.

Some accessor methods return pointers. Those pointers refer to internal fields. The lifetimes of those fields are tied to the lifetime of the containing line info object. Such pointers remain valid until `gpiod_line_info_free` is called on the containing line info object. They must not be freed by the caller.

#### 4.4.2 Function Documentation

##### 4.4.2.1 `gpiod_line_info_copy()`

```
struct gpiod_line_info * gpiod_line_info_copy (
    struct gpiod_line_info * info)
```

Copy a line info object.

##### Parameters

<code>info</code>	Line info to copy.
-------------------	--------------------

##### Returns

Copy of the line info or NULL on error. The returned object must be freed by the caller using `gpiod_line_info_free`.

##### 4.4.2.2 `gpiod_line_info_free()`

```
void gpiod_line_info_free (
    struct gpiod_line_info * info)
```

Free a line info object and release all associated resources.



## Parameters

<i>info</i>	GPIO line info object to free.
-------------	--------------------------------

#### 4.4.2.3 `gpiod_line_info_get_bias()`

```
enum gpiod\_line\_bias gpiod_line_info_get_bias (  
    struct gpiod\_line\_info * info)
```

Get the bias setting of the line.

## Parameters

<i>info</i>	GPIO line object.
-------------	-------------------

## Returns

Returns [GPIOD\\_LINE\\_BIAS\\_PULL\\_UP](#), [GPIOD\\_LINE\\_BIAS\\_PULL\\_DOWN](#), [GPIOD\\_LINE\\_BIAS\\_DISABLED](#) or [GPIOD\\_LINE\\_BIAS\\_UNKNOWN](#).

#### 4.4.2.4 `gpiod_line_info_get_consumer()`

```
const char * gpiod_line_info_get_consumer (  
    struct gpiod\_line\_info * info)
```

Get the name of the consumer of the line.

## Parameters

<i>info</i>	GPIO line info object.
-------------	------------------------

## Returns

Name of the GPIO consumer as it is represented in the kernel. This function returns a valid pointer to a null-terminated string or NULL if the consumer name is not set. The string lifetime is tied to the line info object so the pointer must not be freed.

#### 4.4.2.5 `gpiod_line_info_get_debounce_period_us()`

```
unsigned long gpiod_line_info_get_debounce_period_us (  
    struct gpiod\_line\_info * info)
```

Get the debounce period of the line, in microseconds.

## Parameters

<i>info</i>	GPIO line info object.
-------------	------------------------

## Returns

Debounce period in microseconds. 0 if the line is not debounced.

#### 4.4.2.6 `gpiod_line_info_get_direction()`

```
enum gpiod_line_direction gpiod_line_info_get_direction (  
    struct gpiod_line_info * info)
```

Get the direction setting of the line.

## Parameters

<i>info</i>	GPIO line info object.
-------------	------------------------

## Returns

Returns `GPIO_LINE_DIRECTION_INPUT` or `GPIO_LINE_DIRECTION_OUTPUT`.

#### 4.4.2.7 `gpiod_line_info_get_drive()`

```
enum gpiod_line_drive gpiod_line_info_get_drive (  
    struct gpiod_line_info * info)
```

Get the drive setting of the line.

## Parameters

<i>info</i>	GPIO line info object.
-------------	------------------------

## Returns

Returns `GPIO_LINE_DRIVE_PUSH_PULL`, `GPIO_LINE_DRIVE_OPEN_DRAIN` or `GPIO_LINE_DRIVE_OPEN_SOURCE`.

#### 4.4.2.8 `gpiod_line_info_get_edge_detection()`

```
enum gpiod_line_edge gpiod_line_info_get_edge_detection (  
    struct gpiod_line_info * info)
```

Get the edge detection setting of the line.

## Parameters

<i>info</i>	GPIO line info object.
-------------	------------------------

## Returns

Returns `GPIO_LINE_EDGE_NONE`, `GPIO_LINE_EDGE_RISING`, `GPIO_LINE_EDGE_FALLING` or `GPIO_LINE_EDGE_BOTH`.

#### 4.4.2.9 `gpiod_line_info_get_event_clock()`

```
enum gpiod_line_clock gpiod_line_info_get_event_clock (  
    struct gpiod_line_info * info)
```

Get the event clock setting used for edge event timestamps for the line.

**Parameters**

<i>info</i>	GPIO line info object.
-------------	------------------------

**Returns**

Returns [GPIO\\_LINE\\_CLOCK\\_MONOTONIC](#), [GPIO\\_LINE\\_CLOCK\\_HTE](#) or [GPIO\\_LINE\\_CLOCK\\_REALTIME](#).

**4.4.2.10 `gpiod_line_info_get_name()`**

```
const char * gpiod_line_info_get_name (  
    struct gpiod\_line\_info * info)
```

Get the name of the line.

**Parameters**

<i>info</i>	GPIO line info object.
-------------	------------------------

**Returns**

Name of the GPIO line as it is represented in the kernel. This function returns a valid pointer to a null-terminated string or NULL if the line is unnamed. The string lifetime is tied to the line info object so the pointer must not be freed.

**4.4.2.11 `gpiod_line_info_get_offset()`**

```
unsigned int gpiod_line_info_get_offset (  
    struct gpiod\_line\_info * info)
```

Get the offset of the line.

**Parameters**

<i>info</i>	GPIO line info object.
-------------	------------------------

**Returns**

Offset of the line within the parent chip.

The offset uniquely identifies the line on the chip. The combination of the chip and offset uniquely identifies the line within the system.

**4.4.2.12 `gpiod_line_info_is_active_low()`**

```
bool gpiod_line_info_is_active_low (  
    struct gpiod\_line\_info * info)
```

Check if the logical value of the line is inverted compared to the physical.

## Parameters

info	GPIO line object.
------	-------------------

## Returns

True if the line is "active-low", false otherwise.

**4.4.2.13 `gpiod_line_info_is_debounced()`**

```
bool gpiod_line_info_is_debounced (  
    struct gpiod_line_info * info)
```

Check if the line is debounced (either by hardware or by the kernel software debouncer).

## Parameters

info	GPIO line info object.
------	------------------------

## Returns

True if the line is debounced, false otherwise.

**4.4.2.14 `gpiod_line_info_is_used()`**

```
bool gpiod_line_info_is_used (  
    struct gpiod_line_info * info)
```

Check if the line is in use.

## Parameters

info	GPIO line object.
------	-------------------

## Returns

True if the line is in use, false otherwise.

The exact reason a line is busy cannot be determined from user space. It may have been requested by another process or hogged by the kernel. It only matters that the line is used and can't be requested until released by the existing consumer.

## 4.5 Line status watch events

## Enumerations

- enum `gpiod_info_event_type` { `GPIOD_INFO_EVENT_LINE_REQUESTED` = 1, `GPIOD_INFO_EVENT_LINE_RELEASED`, `GPIOD_INFO_EVENT_LINE_CONFIG_CHANGED` }

*Line status change event types.*

## Functions

- void `gpiod_info_event_free` (struct `gpiod_info_event` \*event)  
*Free the info event object and release all associated resources.*
- enum `gpiod_info_event_type` `gpiod_info_event_get_event_type` (struct `gpiod_info_event` \*event)  
*Get the event type of the status change event.*
- uint64\_t `gpiod_info_event_get_timestamp_ns` (struct `gpiod_info_event` \*event)  
*Get the timestamp of the event.*
- struct `gpiod_line_info` \* `gpiod_info_event_get_line_info` (struct `gpiod_info_event` \*event)  
*Get the snapshot of line-info associated with the event.*

### 4.5.1 Detailed Description

Accessors for the info event objects allowing to monitor changes in GPIO line status.

Callers are notified about changes in a line's status due to GPIO uAPI calls. Each info event contains information about the event itself (timestamp, type) as well as a snapshot of line's status in the form of a line-info object.

### 4.5.2 Enumeration Type Documentation

#### 4.5.2.1 `gpiod_info_event_type`

```
enum gpiod_info_event_type
```

Line status change event types.

Enumerator

<code>GPIOD_INFO_EVENT_LINE_REQUESTED</code>	Line has been requested.
<code>GPIOD_INFO_EVENT_LINE_RELEASED</code>	Previously requested line has been released.
<code>GPIOD_INFO_EVENT_LINE_CONFIG_CHANGED</code>	Line configuration has changed.

### 4.5.3 Function Documentation

#### 4.5.3.1 `gpiod_info_event_free()`

```
void gpiod_info_event_free (  
    struct gpiod_info_event * event)
```

Free the info event object and release all associated resources.

Parameters

<i>event</i>	Info event to free.
--------------	---------------------

#### 4.5.3.2 `gpiod_info_event_get_event_type()`

```
enum gpiod_info_event_type gpiod_info_event_get_event_type (  
    struct gpiod_info_event * event)
```

Get the event type of the status change event.

## Parameters

<i>event</i>	Line status watch event.
--------------	--------------------------

## Returns

One of [GPIOD\\_INFO\\_EVENT\\_LINE\\_REQUESTED](#), [GPIOD\\_INFO\\_EVENT\\_LINE\\_RELEASED](#) or [GPIOD\\_INFO\\_EVENT\\_LINE](#)

### 4.5.3.3 `gpiod_info_event_get_line_info()`

```
struct gpiod\_line\_info * gpiod_info_event_get_line_info (  
    struct gpiod\_info\_event * event)
```

Get the snapshot of line-info associated with the event.

## Parameters

<i>event</i>	Line info event object.
--------------	-------------------------

## Returns

Returns a pointer to the line-info object associated with the event. The object lifetime is tied to the event object, so the pointer must be not be freed by the caller.

## Warning

Thread-safety: Since the line-info object is tied to the event, different threads may not operate on the event and line-info at the same time. The line-info can be copied using [gpiod\\_line\\_info\\_copy](#) in order to create a standalone object - which then may safely be used from a different thread concurrently.

### 4.5.3.4 `gpiod_info_event_get_timestamp_ns()`

```
uint64_t gpiod_info_event_get_timestamp_ns (  
    struct gpiod\_info\_event * event)
```

Get the timestamp of the event.

## Parameters

<i>event</i>	Line status watch event.
--------------	--------------------------

## Returns

Timestamp in nanoseconds, read from the monotonic clock.

## 4.6 Line settings objects

### Functions

- struct `gpiod_line_settings` \* `gpiod_line_settings_new` (void)  
*Create a new line settings object.*
- void `gpiod_line_settings_free` (struct `gpiod_line_settings` \*settings)  
*Free the line settings object and release all associated resources.*
- void `gpiod_line_settings_reset` (struct `gpiod_line_settings` \*settings)  
*Reset the line settings object to its default values.*
- struct `gpiod_line_settings` \* `gpiod_line_settings_copy` (struct `gpiod_line_settings` \*settings)  
*Copy the line settings object.*
- int `gpiod_line_settings_set_direction` (struct `gpiod_line_settings` \*settings, enum `gpiod_line_direction` direction)  
*Set direction.*
- enum `gpiod_line_direction` `gpiod_line_settings_get_direction` (struct `gpiod_line_settings` \*settings)  
*Get direction.*
- int `gpiod_line_settings_set_edge_detection` (struct `gpiod_line_settings` \*settings, enum `gpiod_line_edge` edge)  
*Set edge detection.*
- enum `gpiod_line_edge` `gpiod_line_settings_get_edge_detection` (struct `gpiod_line_settings` \*settings)  
*Get edge detection.*
- int `gpiod_line_settings_set_bias` (struct `gpiod_line_settings` \*settings, enum `gpiod_line_bias` bias)  
*Set bias.*
- enum `gpiod_line_bias` `gpiod_line_settings_get_bias` (struct `gpiod_line_settings` \*settings)  
*Get bias.*
- int `gpiod_line_settings_set_drive` (struct `gpiod_line_settings` \*settings, enum `gpiod_line_drive` drive)  
*Set drive.*
- enum `gpiod_line_drive` `gpiod_line_settings_get_drive` (struct `gpiod_line_settings` \*settings)  
*Get drive.*
- void `gpiod_line_settings_set_active_low` (struct `gpiod_line_settings` \*settings, bool active\_low)  
*Set active-low setting.*
- bool `gpiod_line_settings_get_active_low` (struct `gpiod_line_settings` \*settings)  
*Get active-low setting.*
- void `gpiod_line_settings_set_debounce_period_us` (struct `gpiod_line_settings` \*settings, unsigned long period)  
*Set debounce period.*
- unsigned long `gpiod_line_settings_get_debounce_period_us` (struct `gpiod_line_settings` \*settings)  
*Get debounce period.*
- int `gpiod_line_settings_set_event_clock` (struct `gpiod_line_settings` \*settings, enum `gpiod_line_clock` event\_clock)  
*Set event clock.*
- enum `gpiod_line_clock` `gpiod_line_settings_get_event_clock` (struct `gpiod_line_settings` \*settings)  
*Get event clock setting.*
- int `gpiod_line_settings_set_output_value` (struct `gpiod_line_settings` \*settings, enum `gpiod_line_value` value)  
*Set the output value.*
- enum `gpiod_line_value` `gpiod_line_settings_get_output_value` (struct `gpiod_line_settings` \*settings)  
*Get the output value.*



### 4.6.1 Detailed Description

Functions for manipulating line settings objects.

Line settings object contains a set of line properties that can be used when requesting lines or reconfiguring an existing request.

Mutators in general can only fail if the new property value is invalid. The return values can be safely ignored - the object remains valid even after a mutator fails and simply uses the sane default appropriate for given property.

### 4.6.2 Function Documentation

#### 4.6.2.1 `gpiod_line_settings_copy()`

```
struct gpiod\_line\_settings * gpiod_line_settings_copy (  
    struct gpiod\_line\_settings * settings)
```

Copy the line settings object.

Parameters

<i>settings</i>	Line settings object to copy.
-----------------	-------------------------------

Returns

New line settings object that must be freed using [gpiod\\_line\\_settings\\_free](#) or NULL on failure.

#### 4.6.2.2 `gpiod_line_settings_free()`

```
void gpiod_line_settings_free (  
    struct gpiod\_line\_settings * settings)
```

Free the line settings object and release all associated resources.

Parameters

<i>settings</i>	Line settings object.
-----------------	-----------------------

#### 4.6.2.3 `gpiod_line_settings_get_active_low()`

```
bool gpiod_line_settings_get_active_low (  
    struct gpiod\_line\_settings * settings)
```

Get active-low setting.

Parameters

<i>settings</i>	Line settings object.
-----------------	-----------------------

Returns

True if active-low is enabled, false otherwise.

#### 4.6.2.4 `gpiod_line_settings_get_bias()`

```
enum gpiod\_line\_bias gpiod_line_settings_get_bias (  
    struct gpiod\_line\_settings * settings)
```

Get bias.

**Parameters**

<i>settings</i>	Line settings object.
-----------------	-----------------------

**Returns**

Current bias setting.

**4.6.2.5 gpiod\_line\_settings\_get\_debounce\_period\_us()**

```
unsigned long gpiod_line_settings_get_debounce_period_us (  
    struct gpiod\_line\_settings * settings)
```

Get debounce period.

**Parameters**

<i>settings</i>	Line settings object.
-----------------	-----------------------

**Returns**

Current debounce period in microseconds.

**4.6.2.6 gpiod\_line\_settings\_get\_direction()**

```
enum gpiod\_line\_direction gpiod_line_settings_get_direction (  
    struct gpiod\_line\_settings * settings)
```

Get direction.

**Parameters**

<i>settings</i>	Line settings object.
-----------------	-----------------------

**Returns**

Current direction.

**4.6.2.7 gpiod\_line\_settings\_get\_drive()**

```
enum gpiod\_line\_drive gpiod_line_settings_get_drive (  
    struct gpiod\_line\_settings * settings)
```

Get drive.

## Parameters

<i>settings</i>	Line settings object.
-----------------	-----------------------

## Returns

Current drive setting.

#### 4.6.2.8 `gpiod_line_settings_get_edge_detection()`

```
enum gpiod_line_edge gpiod_line_settings_get_edge_detection (  
    struct gpiod_line_settings * settings)
```

Get edge detection.

## Parameters

<i>settings</i>	Line settings object.
-----------------	-----------------------

## Returns

Current edge detection setting.

#### 4.6.2.9 `gpiod_line_settings_get_event_clock()`

```
enum gpiod_line_clock gpiod_line_settings_get_event_clock (  
    struct gpiod_line_settings * settings)
```

Get event clock setting.

## Parameters

<i>settings</i>	Line settings object.
-----------------	-----------------------

## Returns

Current event clock setting.

#### 4.6.2.10 `gpiod_line_settings_get_output_value()`

```
enum gpiod_line_value gpiod_line_settings_get_output_value (  
    struct gpiod_line_settings * settings)
```

Get the output value.

**Parameters**

<i>settings</i>	Line settings object.
-----------------	-----------------------

**Returns**

Current output value.

**4.6.2.11 `gpiod_line_settings_new()`**

```
struct gpiod\_line\_settings * gpiod_line_settings_new (  
    void )
```

Create a new line settings object.

**Returns**

New line settings object or NULL on error. The returned object must be freed by the caller using [gpiod\\_line\\_settings\\_free](#).

**4.6.2.12 `gpiod_line_settings_reset()`**

```
void gpiod_line_settings_reset (  
    struct gpiod\_line\_settings * settings)
```

Reset the line settings object to its default values.

**Parameters**

<i>settings</i>	Line settings object.
-----------------	-----------------------

**4.6.2.13 `gpiod_line_settings_set_active_low()`**

```
void gpiod_line_settings_set_active_low (  
    struct gpiod\_line\_settings * settings,  
    bool active_low)
```

Set active-low setting.

**Parameters**

<i>settings</i>	Line settings object.
<i>active_low</i>	New active-low setting.

**4.6.2.14 `gpiod_line_settings_set_bias()`**

```
int gpiod_line_settings_set_bias (  
    struct gpiod\_line\_settings * settings,  
    enum gpiod\_line\_bias bias)
```

Set bias.

## Parameters

<i>settings</i>	Line settings object.
<i>bias</i>	New bias.

## Returns

0 on success, -1 on failure.

**4.6.2.15** `gpiod_line_settings_set_debounce_period_us()`

```
void gpiod_line_settings_set_debounce_period_us (  
    struct gpiod_line_settings * settings,  
    unsigned long period)
```

Set debounce period.

## Parameters

<i>settings</i>	Line settings object.
<i>period</i>	New debounce period in microseconds.

**4.6.2.16** `gpiod_line_settings_set_direction()`

```
int gpiod_line_settings_set_direction (  
    struct gpiod_line_settings * settings,  
    enum gpiod_line_direction direction)
```

Set direction.

## Parameters

<i>settings</i>	Line settings object.
<i>direction</i>	New direction.

## Returns

0 on success, -1 on error.

**4.6.2.17** `gpiod_line_settings_set_drive()`

```
int gpiod_line_settings_set_drive (  
    struct gpiod_line_settings * settings,  
    enum gpiod_line_drive drive)
```

Set drive.

**Parameters**

<i>settings</i>	Line settings object.
<i>drive</i>	New drive setting.

**Returns**

0 on success, -1 on failure.

**4.6.2.18 `gpiod_line_settings_set_edge_detection()`**

```
int gpiod_line_settings_set_edge_detection (  
    struct gpiod\_line\_settings * settings,  
    enum gpiod\_line\_edge edge)
```

Set edge detection.

**Parameters**

<i>settings</i>	Line settings object.
<i>edge</i>	New edge detection setting.

**Returns**

0 on success, -1 on failure.

**4.6.2.19 `gpiod_line_settings_set_event_clock()`**

```
int gpiod_line_settings_set_event_clock (  
    struct gpiod\_line\_settings * settings,  
    enum gpiod\_line\_clock event_clock)
```

Set event clock.

**Parameters**

<i>settings</i>	Line settings object.
<i>event_clock</i>	New event clock.

**Returns**

0 on success, -1 on failure.

**4.6.2.20 `gpiod_line_settings_set_output_value()`**

```
int gpiod_line_settings_set_output_value (  
    struct gpiod\_line\_settings * settings,  
    enum gpiod\_line\_value value)
```

Set the output value.

## Parameters

<i>settings</i>	Line settings object.
<i>value</i>	New output value.

## Returns

0 on success, -1 on failure.

## 4.7 Line configuration objects

## Functions

- struct [gpiod\\_line\\_config](#) \* [gpiod\\_line\\_config\\_new](#) (void)  
*Create a new line config object.*
- void [gpiod\\_line\\_config\\_free](#) (struct [gpiod\\_line\\_config](#) \*config)  
*Free the line config object and release all associated resources.*
- void [gpiod\\_line\\_config\\_reset](#) (struct [gpiod\\_line\\_config](#) \*config)  
*Reset the line config object.*
- int [gpiod\\_line\\_config\\_add\\_line\\_settings](#) (struct [gpiod\\_line\\_config](#) \*config, const unsigned int \*offsets, size\_t num\_offsets, struct [gpiod\\_line\\_settings](#) \*settings)  
*Add line settings for a set of offsets.*
- struct [gpiod\\_line\\_settings](#) \* [gpiod\\_line\\_config\\_get\\_line\\_settings](#) (struct [gpiod\\_line\\_config](#) \*config, unsigned int offset)  
*Get line settings for offset.*
- int [gpiod\\_line\\_config\\_set\\_output\\_values](#) (struct [gpiod\\_line\\_config](#) \*config, const enum [gpiod\\_line\\_value](#) \*values, size\_t num\_values)  
*Set output values for a number of lines.*
- size\_t [gpiod\\_line\\_config\\_get\\_num\\_configured\\_offsets](#) (struct [gpiod\\_line\\_config](#) \*config)  
*Get the number of configured line offsets.*
- size\_t [gpiod\\_line\\_config\\_get\\_configured\\_offsets](#) (struct [gpiod\\_line\\_config](#) \*config, unsigned int \*offsets, size\_t max\_offsets)  
*Get configured offsets.*

### 4.7.1 Detailed Description

Functions for manipulating line configuration objects.

The line-config object contains the configuration for lines that can be used in two cases:

- when making a line request
- when reconfiguring a set of already requested lines.

A new line-config object is empty. Using it in a request will lead to an error. In order to a line-config to become useful, it needs to be assigned at least one offset-to-settings mapping by calling [gpiod\\_line\\_config\\_add\\_line\\_settings](#).

When calling [gpiod\\_chip\\_request\\_lines](#), the library will request all offsets that were assigned settings in the order that they were assigned. If any of the offsets was duplicated, the last one will take precedence.

## 4.7.2 Function Documentation

### 4.7.2.1 `gpiod_line_config_add_line_settings()`

```
int gpiod_line_config_add_line_settings (
    struct gpiod_line_config * config,
    const unsigned int * offsets,
    size_t num_offsets,
    struct gpiod_line_settings * settings)
```

Add line settings for a set of offsets.

#### Parameters

<i>config</i>	Line config object.
<i>offsets</i>	Array of offsets for which to apply the settings.
<i>num_offsets</i>	Number of offsets stored in the offsets array.
<i>settings</i>	Line settings to apply.

#### Returns

0 on success, -1 on failure.

### 4.7.2.2 `gpiod_line_config_free()`

```
void gpiod_line_config_free (
    struct gpiod_line_config * config)
```

Free the line config object and release all associated resources.

#### Parameters

<i>config</i>	Line config object to free.
---------------	-----------------------------

### 4.7.2.3 `gpiod_line_config_get_configured_offsets()`

```
size_t gpiod_line_config_get_configured_offsets (
    struct gpiod_line_config * config,
    unsigned int * offsets,
    size_t max_offsets)
```

Get configured offsets.

#### Parameters

<i>config</i>	Line config object.
<i>offsets</i>	Array to store offsets.
<i>max_offsets</i>	Number of offsets that can be stored in the offsets array.

#### Returns

Number of offsets stored in the offsets array.

If *max\_offsets* is lower than the number of lines actually requested (this value can be retrieved using [gpiod\\_line\\_config\\_get\\_num\\_configured\\_offsets](#)), then only up to *max\_lines* offsets will be stored in *offsets*.



#### 4.7.2.4 `gpiod_line_config_get_line_settings()`

```
struct gpiod\_line\_settings * gpiod_line_config_get_line_settings (
    struct gpiod\_line\_config * config,
    unsigned int offset)
```

Get line settings for offset.

##### Parameters

<i>config</i>	Line config object.
<i>offset</i>	Offset for which to get line settings.

##### Returns

New line settings object (must be freed by the caller) or NULL on error.

#### 4.7.2.5 `gpiod_line_config_get_num_configured_offsets()`

```
size_t gpiod_line_config_get_num_configured_offsets (
    struct gpiod\_line\_config * config)
```

Get the number of configured line offsets.

##### Parameters

<i>config</i>	Line config object.
---------------	---------------------

##### Returns

Number of offsets for which line settings have been added.

#### 4.7.2.6 `gpiod_line_config_new()`

```
struct gpiod\_line\_config * gpiod_line_config_new (
    void )
```

Create a new line config object.

##### Returns

New line config object or NULL on error. The returned object must be freed by the caller using [gpiod\\_line\\_config\\_free](#).

#### 4.7.2.7 `gpiod_line_config_reset()`

```
void gpiod_line_config_reset (
    struct gpiod\_line\_config * config)
```

Reset the line config object.

## Parameters

<i>config</i>	Line config object to free.
---------------	-----------------------------

Resets the entire configuration stored in the object. This is useful if the user wants to reuse the object without reallocating it.

#### 4.7.2.8 `gpiod_line_config_set_output_values()`

```
int gpiod_line_config_set_output_values (
    struct gpiod_line_config * config,
    const enum gpiod_line_value * values,
    size_t num_values)
```

Set output values for a number of lines.

## Parameters

<i>config</i>	Line config object.
<i>values</i>	Buffer containing the output values.
<i>num_values</i>	Number of values in the buffer.

## Returns

0 on success, -1 on error.

This is a helper that allows users to set multiple (potentially different) output values at once while using the same line settings object. Instead of modifying the output value in the settings object and calling [gpiod\\_line\\_config\\_add\\_line\\_settings](#) multiple times, we can specify the settings, add them for a set of offsets and then call this function to set the output values.

Values set by this function override whatever values were specified in the regular line settings.

Each value must be associated with the line identified by the corresponding entry in the offset array filled by [gpiod\\_line\\_request\\_get\\_requested\\_offsets](#).

## 4.8 Request configuration objects

## Functions

- struct [gpiod\\_request\\_config](#) \* [gpiod\\_request\\_config\\_new](#) (void)  
*Create a new request config object.*
- void [gpiod\\_request\\_config\\_free](#) (struct [gpiod\\_request\\_config](#) \*config)  
*Free the request config object and release all associated resources.*
- void [gpiod\\_request\\_config\\_set\\_consumer](#) (struct [gpiod\\_request\\_config](#) \*config, const char \*consumer)  
*Set the consumer name for the request.*
- const char \* [gpiod\\_request\\_config\\_get\\_consumer](#) (struct [gpiod\\_request\\_config](#) \*config)  
*Get the consumer name configured in the request config.*
- void [gpiod\\_request\\_config\\_set\\_event\\_buffer\\_size](#) (struct [gpiod\\_request\\_config](#) \*config, size\_t event\_buffer↔\_size)  
*Set the size of the kernel event buffer for the request.*
- size\_t [gpiod\\_request\\_config\\_get\\_event\\_buffer\\_size](#) (struct [gpiod\\_request\\_config](#) \*config)  
*Get the edge event buffer size for the request config.*

### 4.8.1 Detailed Description

Functions for manipulating request configuration objects.

Request config objects are used to pass a set of options to the kernel at the time of the line request. The mutators don't return error values. If the values are invalid, in general they are silently adjusted to acceptable ranges.

### 4.8.2 Function Documentation

#### 4.8.2.1 `gpiod_request_config_free()`

```
void gpiod_request_config_free (  
    struct gpiod_request_config * config)
```

Free the request config object and release all associated resources.

##### Parameters

<i>config</i>	Line config object.
---------------	---------------------

#### 4.8.2.2 `gpiod_request_config_get_consumer()`

```
const char * gpiod_request_config_get_consumer (  
    struct gpiod_request_config * config)
```

Get the consumer name configured in the request config.

##### Parameters

<i>config</i>	Request config object.
---------------	------------------------

##### Returns

Consumer name stored in the request config.

#### 4.8.2.3 `gpiod_request_config_get_event_buffer_size()`

```
size_t gpiod_request_config_get_event_buffer_size (  
    struct gpiod_request_config * config)
```

Get the edge event buffer size for the request config.

##### Parameters

<i>config</i>	Request config object.
---------------	------------------------

##### Returns

Edge event buffer size setting from the request config.

#### 4.8.2.4 `gpiod_request_config_new()`

```
struct gpiod\_request\_config * gpiod_request_config_new (
    void )
```

Create a new request config object.

##### Returns

New request config object or NULL on error. The returned object must be freed by the caller using [gpiod\\_request\\_config\\_free](#).

#### 4.8.2.5 `gpiod_request_config_set_consumer()`

```
void gpiod_request_config_set_consumer (
    struct gpiod\_request\_config * config,
    const char * consumer)
```

Set the consumer name for the request.

##### Parameters

<i>config</i>	Request config object.
<i>consumer</i>	Consumer name.

##### Note

If the consumer string is too long, it will be truncated to the max accepted length.

#### 4.8.2.6 `gpiod_request_config_set_event_buffer_size()`

```
void gpiod_request_config_set_event_buffer_size (
    struct gpiod\_request\_config * config,
    size_t event_buffer_size)
```

Set the size of the kernel event buffer for the request.

##### Parameters

<i>config</i>	Request config object.
<i>event_buffer_size</i>	New event buffer size.

##### Note

The kernel may adjust the value if it's too high. If set to 0, the default value will be used.

The kernel buffer is distinct from and independent of the user space buffer ([gpiod\\_edge\\_event\\_buffer\\_new](#)).

## 4.9 Line request operations

### Functions

- void `gpod_line_request_release` (struct `gpod_line_request` \*request)  
*Release the requested lines and free all associated resources.*
- const char \* `gpod_line_request_get_chip_name` (struct `gpod_line_request` \*request)  
*Get the name of the chip this request was made on.*
- size\_t `gpod_line_request_get_num_requested_lines` (struct `gpod_line_request` \*request)  
*Get the number of lines in the request.*
- size\_t `gpod_line_request_get_requested_offsets` (struct `gpod_line_request` \*request, unsigned int \*offsets, size\_t max\_offsets)  
*Get the offsets of the lines in the request.*
- enum `gpod_line_value` `gpod_line_request_get_value` (struct `gpod_line_request` \*request, unsigned int offset)  
*Get the value of a single requested line.*
- int `gpod_line_request_get_values_subset` (struct `gpod_line_request` \*request, size\_t num\_values, const unsigned int \*offsets, enum `gpod_line_value` \*values)  
*Get the values of a subset of requested lines.*
- int `gpod_line_request_get_values` (struct `gpod_line_request` \*request, enum `gpod_line_value` \*values)  
*Get the values of all requested lines.*
- int `gpod_line_request_set_value` (struct `gpod_line_request` \*request, unsigned int offset, enum `gpod_line_value` value)  
*Set the value of a single requested line.*
- int `gpod_line_request_set_values_subset` (struct `gpod_line_request` \*request, size\_t num\_values, const unsigned int \*offsets, const enum `gpod_line_value` \*values)  
*Set the values of a subset of requested lines.*
- int `gpod_line_request_set_values` (struct `gpod_line_request` \*request, const enum `gpod_line_value` \*values)  
*Set the values of all lines associated with a request.*
- int `gpod_line_request_reconfigure_lines` (struct `gpod_line_request` \*request, struct `gpod_line_config` \*config)  
*Update the configuration of lines associated with a line request.*
- int `gpod_line_request_get_fd` (struct `gpod_line_request` \*request)  
*Get the file descriptor associated with a line request.*
- int `gpod_line_request_wait_edge_events` (struct `gpod_line_request` \*request, int64\_t timeout\_ns)  
*Wait for edge events on any of the requested lines.*
- int `gpod_line_request_read_edge_events` (struct `gpod_line_request` \*request, struct `gpod_edge_event_buffer` \*buffer, size\_t max\_events)  
*Read a number of edge events from a line request.*

### 4.9.1 Detailed Description

Functions allowing interactions with requested lines.

### 4.9.2 Function Documentation

#### 4.9.2.1 `gpod_line_request_get_chip_name()`

```
const char * gpod_line_request_get_chip_name (
    struct gpod_line_request * request)
```

Get the name of the chip this request was made on.

**Parameters**

<i>request</i>	Line request object.
----------------	----------------------

**Returns**

Name the GPIO chip device. The returned pointer is valid for the lifetime of the request object and must not be freed by the caller.

**4.9.2.2 `gpiod_line_request_get_fd()`**

```
int gpiod_line_request_get_fd (  
    struct gpiod\_line\_request * request)
```

Get the file descriptor associated with a line request.

**Parameters**

<i>request</i>	GPIO line request.
----------------	--------------------

**Returns**

The file descriptor associated with the request. This function never fails. The returned file descriptor must not be closed by the caller. Call [gpiod\\_line\\_request\\_release](#) to close the file.

**4.9.2.3 `gpiod_line_request_get_num_requested_lines()`**

```
size_t gpiod_line_request_get_num_requested_lines (  
    struct gpiod\_line\_request * request)
```

Get the number of lines in the request.

**Parameters**

<i>request</i>	Line request object.
----------------	----------------------

**Returns**

Number of requested lines.

**4.9.2.4 `gpiod_line_request_get_requested_offsets()`**

```
size_t gpiod_line_request_get_requested_offsets (  
    struct gpiod\_line\_request * request,  
    unsigned int * offsets,  
    size_t max_offsets)
```

Get the offsets of the lines in the request.

## Parameters

<i>request</i>	Line request object.
<i>offsets</i>	Array to store offsets.
<i>max_offsets</i>	Number of offsets that can be stored in the offsets array.

## Returns

Number of offsets stored in the offsets array.

If `max_offsets` is lower than the number of lines actually requested (this value can be retrieved using [gpiod\\_line\\_request\\_get\\_num\\_requested\\_lines](#)), then only up to `max_lines` offsets will be stored in offsets.

#### 4.9.2.5 gpiod\_line\_request\_get\_value()

```
enum gpiod_line_value gpiod_line_request_get_value (  
    struct gpiod_line_request * request,  
    unsigned int offset)
```

Get the value of a single requested line.

## Parameters

<i>request</i>	Line request object.
<i>offset</i>	The offset of the line of which the value should be read.

## Returns

Returns 1 or 0 on success and -1 on error.

#### 4.9.2.6 gpiod\_line\_request\_get\_values()

```
int gpiod_line_request_get_values (  
    struct gpiod_line_request * request,  
    enum gpiod_line_value * values)
```

Get the values of all requested lines.

## Parameters

<i>request</i>	GPIO line request.
<i>values</i>	Array in which the values will be stored. Must be sized to hold the number of lines filled by <a href="#">gpiod_line_request_get_num_requested_lines</a> . Each value is associated with the line identified by the corresponding entry in the offset array filled by <a href="#">gpiod_line_request_get_requested_offsets</a> .

## Returns

0 on success, -1 on failure.

#### 4.9.2.7 `gpiod_line_request_get_values_subset()`

```
int gpiod_line_request_get_values_subset (
    struct gpiod_line_request * request,
    size_t num_values,
    const unsigned int * offsets,
    enum gpiod_line_value * values)
```

Get the values of a subset of requested lines.

##### Parameters

<i>request</i>	GPIO line request.
<i>num_values</i>	Number of lines for which to read values.
<i>offsets</i>	Array of offsets identifying the subset of requested lines from which to read values.
<i>values</i>	Array in which the values will be stored. Must be sized to hold <code>num_values</code> entries. Each value is associated with the line identified by the corresponding entry in <code>offsets</code> .

##### Returns

0 on success, -1 on failure.

#### 4.9.2.8 `gpiod_line_request_read_edge_events()`

```
int gpiod_line_request_read_edge_events (
    struct gpiod_line_request * request,
    struct gpiod_edge_event_buffer * buffer,
    size_t max_events)
```

Read a number of edge events from a line request.

##### Parameters

<i>request</i>	GPIO line request.
<i>buffer</i>	Edge event buffer, sized to hold at least <code>max_events</code> .
<i>max_events</i>	Maximum number of events to read.

##### Returns

On success returns the number of events read from the file descriptor, on failure return -1.

##### Note

This function will block if no event was queued for the line request.  
Any existing events in the buffer are overwritten. This is not an append operation.

#### 4.9.2.9 `gpiod_line_request_reconfigure_lines()`

```
int gpiod_line_request_reconfigure_lines (
    struct gpiod_line_request * request,
    struct gpiod_line_config * config)
```

Update the configuration of lines associated with a line request.



## Parameters

<i>request</i>	GPIO line request.
<i>config</i>	New line config to apply.

## Returns

0 on success, -1 on failure.

## Note

The new line configuration completely replaces the old.

Any requested lines without overrides are configured to the requested defaults.

Any configured overrides for lines that have not been requested are silently ignored.

**4.9.2.10 `gpiod_line_request_release()`**

```
void gpiod_line_request_release (  
    struct gpiod_line_request * request)
```

Release the requested lines and free all associated resources.

## Parameters

<i>request</i>	Line request object to release.
----------------	---------------------------------

**4.9.2.11 `gpiod_line_request_set_value()`**

```
int gpiod_line_request_set_value (  
    struct gpiod_line_request * request,  
    unsigned int offset,  
    enum gpiod_line_value value)
```

Set the value of a single requested line.

## Parameters

<i>request</i>	Line request object.
<i>offset</i>	The offset of the line for which the value should be set.
<i>value</i>	Value to set.

## Returns

0 on success, -1 on failure.

**4.9.2.12 `gpiod_line_request_set_values()`**

```
int gpiod_line_request_set_values (  
    struct gpiod_line_request * request,  
    const enum gpiod_line_value * values)
```

Set the values of all lines associated with a request.

## Parameters

<i>request</i>	GPIO line request.
<i>values</i>	Array containing the values to set. Must be sized to contain the number of lines filled by <a href="#">gpiod_line_request_get_num_requested_lines</a> . Each value is associated with the line identified by the corresponding entry in the offset array filled by <a href="#">gpiod_line_request_get_requested_offsets</a> .

## Returns

0 on success, -1 on failure.

4.9.2.13 `gpiod_line_request_set_values_subset()`

```
int gpiod_line_request_set_values_subset (
    struct gpiod\_line\_request * request,
    size_t num_values,
    const unsigned int * offsets,
    const enum gpiod\_line\_value * values)
```

Set the values of a subset of requested lines.

## Parameters

<i>request</i>	GPIO line request.
<i>num_values</i>	Number of lines for which to set values.
<i>offsets</i>	Array of offsets, containing the number of entries specified by <i>num_values</i> , identifying the requested lines for which to set values.
<i>values</i>	Array of values to set, containing the number of entries specified by <i>num_values</i> . Each value is associated with the line identified by the corresponding entry in <i>offsets</i> .

## Returns

0 on success, -1 on failure.

4.9.2.14 `gpiod_line_request_wait_edge_events()`

```
int gpiod_line_request_wait_edge_events (
    struct gpiod\_line\_request * request,
    int64_t timeout_ns)
```

Wait for edge events on any of the requested lines.

## Parameters

<i>request</i>	GPIO line request.
<i>timeout_ns</i>	Wait time limit in nanoseconds. If set to 0, the function returns immediately. If set to a negative number, the function blocks indefinitely until an event becomes available.

## Returns

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

Lines must have edge detection set for edge events to be emitted. By default edge detection is disabled.

## 4.10 Line edge events handling

### Enumerations

- enum `gpiod_edge_event_type` { `GPIOD_EDGE_EVENT_RISING_EDGE` = 1, `GPIOD_EDGE_EVENT_FALLING_EDGE` }
- Event types.*

### Functions

- void `gpiod_edge_event_free` (struct `gpiod_edge_event` \*event)  
*Free the edge event object.*
- struct `gpiod_edge_event` \* `gpiod_edge_event_copy` (struct `gpiod_edge_event` \*event)  
*Copy the edge event object.*
- enum `gpiod_edge_event_type` `gpiod_edge_event_get_event_type` (struct `gpiod_edge_event` \*event)  
*Get the event type.*
- uint64\_t `gpiod_edge_event_get_timestamp_ns` (struct `gpiod_edge_event` \*event)  
*Get the timestamp of the event.*
- unsigned int `gpiod_edge_event_get_line_offset` (struct `gpiod_edge_event` \*event)  
*Get the offset of the line which triggered the event.*
- unsigned long `gpiod_edge_event_get_global_seqno` (struct `gpiod_edge_event` \*event)  
*Get the global sequence number of the event.*
- unsigned long `gpiod_edge_event_get_line_seqno` (struct `gpiod_edge_event` \*event)  
*Get the event sequence number specific to the line.*
- struct `gpiod_edge_event_buffer` \* `gpiod_edge_event_buffer_new` (size\_t capacity)  
*Create a new edge event buffer.*
- size\_t `gpiod_edge_event_buffer_get_capacity` (struct `gpiod_edge_event_buffer` \*buffer)  
*Get the capacity (the max number of events that can be stored) of the event buffer.*
- void `gpiod_edge_event_buffer_free` (struct `gpiod_edge_event_buffer` \*buffer)  
*Free the edge event buffer and release all associated resources.*
- struct `gpiod_edge_event` \* `gpiod_edge_event_buffer_get_event` (struct `gpiod_edge_event_buffer` \*buffer, unsigned long index)  
*Get an event stored in the buffer.*
- size\_t `gpiod_edge_event_buffer_get_num_events` (struct `gpiod_edge_event_buffer` \*buffer)  
*Get the number of events a buffer has stored.*

### 4.10.1 Detailed Description

Functions and data types for handling edge events.

An edge event object contains information about a single line edge event. It contains the event type, timestamp and the offset of the line on which the event occurred as well as two sequence numbers (global for all lines in the associated request and local for this line only).

Edge events are stored into an edge-event buffer object to improve performance and to limit the number of memory allocations when a large number of events are being read.

### 4.10.2 Enumeration Type Documentation

#### 4.10.2.1 `gpiod_edge_event_type`

enum `gpiod_edge_event_type`

Event types.

## Enumerator

GPIOD_EDGE_EVENT_RISING_EDGE	Rising edge event.
GPIOD_EDGE_EVENT_FALLING_EDGE	Falling edge event.

### 4.10.3 Function Documentation

#### 4.10.3.1 `gpiod_edge_event_buffer_free()`

```
void gpiod_edge_event_buffer_free (
    struct gpiod\_edge\_event\_buffer * buffer)
```

Free the edge event buffer and release all associated resources.

## Parameters

<i>buffer</i>	Edge event buffer to free.
---------------	----------------------------

#### 4.10.3.2 `gpiod_edge_event_buffer_get_capacity()`

```
size_t gpiod_edge_event_buffer_get_capacity (
    struct gpiod\_edge\_event\_buffer * buffer)
```

Get the capacity (the max number of events that can be stored) of the event buffer.

## Parameters

<i>buffer</i>	Edge event buffer.
---------------	--------------------

## Returns

The capacity of the buffer.

#### 4.10.3.3 `gpiod_edge_event_buffer_get_event()`

```
struct gpiod\_edge\_event * gpiod_edge_event_buffer_get_event (
    struct gpiod\_edge\_event\_buffer * buffer,
    unsigned long index)
```

Get an event stored in the buffer.

## Parameters

<i>buffer</i>	Edge event buffer.
<i>index</i>	Index of the event in the buffer.

## Returns

Pointer to an event stored in the buffer. The lifetime of the event is tied to the buffer object. Users must not free the event returned by this function.

## Warning

Thread-safety: Since events are tied to the buffer instance, different threads may not operate on the buffer and any associated events at the same time. Events can be copied using [gpiod\\_edge\\_event\\_copy](#) in order to create a standalone objects - which each may safely be used from a different thread concurrently.

#### 4.10.3.4 `gpiod_edge_event_buffer_get_num_events()`

```
size_t gpiod_edge_event_buffer_get_num_events (
    struct gpiod_edge_event_buffer * buffer)
```

Get the number of events a buffer has stored.

##### Parameters

<i>buffer</i>	Edge event buffer.
---------------	--------------------

##### Returns

Number of events stored in the buffer.

#### 4.10.3.5 `gpiod_edge_event_buffer_new()`

```
struct gpiod_edge_event_buffer * gpiod_edge_event_buffer_new (
    size_t capacity)
```

Create a new edge event buffer.

##### Parameters

<i>capacity</i>	Number of events the buffer can store (min = 1, max = 1024).
-----------------	--

##### Returns

New edge event buffer or NULL on error.

##### Note

If capacity equals 0, it will be set to a default value of 64. If capacity is larger than 1024, it will be limited to 1024.

The user space buffer is independent of the kernel buffer ([gpiod\\_request\\_config\\_set\\_event\\_buffer\\_size](#)). As the user space buffer is filled from the kernel buffer, there is no benefit making the user space buffer larger than the kernel buffer. The default kernel buffer size for each request is (16 \* num\_lines).

#### 4.10.3.6 `gpiod_edge_event_copy()`

```
struct gpiod_edge_event * gpiod_edge_event_copy (
    struct gpiod_edge_event * event)
```

Copy the edge event object.

##### Parameters

<i>event</i>	Edge event to copy.
--------------	---------------------

##### Returns

Copy of the edge event or NULL on error. The returned object must be freed by the caller using [gpiod\\_edge\\_event\\_free](#).

#### 4.10.3.7 `gpiod_edge_event_free()`

```
void gpiod_edge_event_free (  
    struct gpiod_edge_event * event)
```

Free the edge event object.

## Parameters

<i>event</i>	Edge event object to free.
--------------	----------------------------

#### 4.10.3.8 `gpiod_edge_event_get_event_type()`

```
enum gpiod\_edge\_event\_type gpiod_edge_event_get_event_type (  
    struct gpiod\_edge\_event * event)
```

Get the event type.

## Parameters

<i>event</i>	GPIO edge event.
--------------	------------------

## Returns

The event type ([GPIOD\\_EDGE\\_EVENT\\_RISING\\_EDGE](#) or [GPIOD\\_EDGE\\_EVENT\\_FALLING\\_EDGE](#)).

#### 4.10.3.9 `gpiod_edge_event_get_global_seqno()`

```
unsigned long gpiod_edge_event_get_global_seqno (  
    struct gpiod\_edge\_event * event)
```

Get the global sequence number of the event.

## Parameters

<i>event</i>	GPIO edge event.
--------------	------------------

## Returns

Sequence number of the event in the series of events for all lines in the associated line request.

#### 4.10.3.10 `gpiod_edge_event_get_line_offset()`

```
unsigned int gpiod_edge_event_get_line_offset (  
    struct gpiod\_edge\_event * event)
```

Get the offset of the line which triggered the event.

## Parameters

<i>event</i>	GPIO edge event.
--------------	------------------

## Returns

Line offset.

#### 4.10.3.11 `gpiod_edge_event_get_line_seqno()`

```
unsigned long gpiod_edge_event_get_line_seqno (  
    struct gpiod\_edge\_event * event)
```

Get the event sequence number specific to the line.

**Parameters**

<i>event</i>	GPIO edge event.
--------------	------------------

**Returns**

Sequence number of the event in the series of events only for this line within the lifetime of the associated line request.

**4.10.3.12 `gpiod_edge_event_get_timestamp_ns()`**

```
uint64_t gpiod_edge_event_get_timestamp_ns (  
    struct gpiod_edge_event * event)
```

Get the timestamp of the event.

**Parameters**

<i>event</i>	GPIO edge event.
--------------	------------------

**Returns**

Timestamp in nanoseconds.

**Note**

The source clock for the timestamp depends on the `event_clock` setting for the line.

**4.11 Stuff that didn't fit anywhere else****Functions**

- bool `gpiod_is_gpiochip_device` (const char \*path)  
*Check if the file pointed to by path is a GPIO chip character device.*
- const char \* `gpiod_api_version` (void)  
*Get the API version of the library as a human-readable string.*

**4.11.1 Detailed Description**

Various libgpiod-related functions.



## 4.11.2 Function Documentation

### 4.11.2.1 `gpiod_api_version()`

```
const char * gpiod_api_version (  
    void )
```

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

#### Returns

A valid pointer to a human-readable string containing the library version. The pointer is valid for the lifetime of the program and must not be freed by the caller.

### 4.11.2.2 `gpiod_is_gpiochip_device()`

```
bool gpiod_is_gpiochip_device (  
    const char * path)
```

Check if the file pointed to by *path* is a GPIO chip character device.

#### Parameters

<i>path</i>	Path to check.
-------------	----------------

#### Returns

True if the file exists and is either a GPIO chip character device or a symbolic link to one.



## Chapter 5

# Class Documentation

### 5.1 `gpiod_chip` Struct Reference

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

#### 5.1.1 Detailed Description

Refer to [GPIO chips](#) for functions that operate on `gpiod_chip`.

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

- [gpiod.h](#)

### 5.2 `gpiod_chip_info` Struct Reference

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

#### 5.2.1 Detailed Description

Refer to [Chip info](#) for functions that operate on `gpiod_chip_info`.

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

- [gpiod.h](#)

### 5.3 `gpiod_edge_event` Struct Reference

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

### 5.3.1 Detailed Description

Refer to [Line edge events handling](#) for functions that operate on [gpod\\_edge\\_event](#).

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

- [gpod.h](#)

## 5.4 gpod\_edge\_event\_buffer Struct Reference

```
#include <gpod.h>
```

### 5.4.1 Detailed Description

Refer to [Line edge events handling](#) for functions that operate on [gpod\\_edge\\_event\\_buffer](#).

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

- [gpod.h](#)

## 5.5 gpod\_info\_event Struct Reference

```
#include <gpod.h>
```

### 5.5.1 Detailed Description

Refer to [Line status watch events](#) for functions that operate on [gpod\\_info\\_event](#).

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

- [gpod.h](#)

## 5.6 gpod\_line\_config Struct Reference

```
#include <gpod.h>
```

### 5.6.1 Detailed Description

Refer to [Line configuration objects](#) for functions that operate on [gpod\\_line\\_config](#).

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

- [gpod.h](#)

## 5.7 gpiod\_line\_info Struct Reference

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

### 5.7.1 Detailed Description

Refer to [Line info](#) for functions that operate on [gpiod\\_line\\_info](#).

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

- [gpiod.h](#)

## 5.8 gpiod\_line\_request Struct Reference

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

### 5.8.1 Detailed Description

Refer to [Line request operations](#) for functions that operate on [gpiod\\_line\\_request](#).

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

- [gpiod.h](#)

## 5.9 gpiod\_line\_settings Struct Reference

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

### 5.9.1 Detailed Description

Refer to [Line settings objects](#) for functions that operate on [gpiod\\_line\\_settings](#).

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

- [gpiod.h](#)

## 5.10 gpiod\_request\_config Struct Reference

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

### 5.10.1 Detailed Description

Refer to [Request configuration objects](#) for functions that operate on [gpiod\\_request\\_config](#).

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

- [gpiod.h](#)



## Chapter 6

# File Documentation

### 6.1 gpiod.h File Reference

```
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
```

#### Enumerations

- enum `gpiod_line_value` { `GPIO_LINE_VALUE_ERROR` = -1 , `GPIO_LINE_VALUE_INACTIVE` = 0 , `GPIO_LINE_VALUE_ACTIVE` = 1 }
- Logical line state.*
- enum `gpiod_line_direction` { `GPIO_LINE_DIRECTION_AS_IS` = 1 , `GPIO_LINE_DIRECTION_INPUT` , `GPIO_LINE_DIRECTION_OUTPUT` }
- Direction settings.*
- enum `gpiod_line_edge` { `GPIO_LINE_EDGE_NONE` = 1 , `GPIO_LINE_EDGE_RISING` , `GPIO_LINE_EDGE_FALLING` , `GPIO_LINE_EDGE_BOTH` }
- Edge detection settings.*
- enum `gpiod_line_bias` { `GPIO_LINE_BIAS_AS_IS` = 1 , `GPIO_LINE_BIAS_UNKNOWN` , `GPIO_LINE_BIAS_DISABLED` , `GPIO_LINE_BIAS_PULL_UP` , `GPIO_LINE_BIAS_PULL_DOWN` }
- Internal bias settings.*
- enum `gpiod_line_drive` { `GPIO_LINE_DRIVE_PUSH_PULL` = 1 , `GPIO_LINE_DRIVE_OPEN_DRAIN` , `GPIO_LINE_DRIVE_OPEN_SOURCE` }
- Drive settings.*
- enum `gpiod_line_clock` { `GPIO_LINE_CLOCK_MONOTONIC` = 1 , `GPIO_LINE_CLOCK_REALTIME` , `GPIO_LINE_CLOCK_HTE` }
- Clock settings.*
- enum `gpiod_info_event_type` { `GPIO_INFO_EVENT_LINE_REQUESTED` = 1 , `GPIO_INFO_EVENT_LINE_RELEASED` , `GPIO_INFO_EVENT_LINE_CONFIG_CHANGED` }
- Line status change event types.*
- enum `gpiod_edge_event_type` { `GPIO_EDGE_EVENT_RISING_EDGE` = 1 , `GPIO_EDGE_EVENT_FALLING_EDGE` }
- Event types.*

## Functions

- struct [gpiod\\_chip](#) \* [gpiod\\_chip\\_open](#) (const char \*path)  
*Open a chip by path.*
- void [gpiod\\_chip\\_close](#) (struct [gpiod\\_chip](#) \*chip)  
*Close the chip and release all associated resources.*
- struct [gpiod\\_chip\\_info](#) \* [gpiod\\_chip\\_get\\_info](#) (struct [gpiod\\_chip](#) \*chip)  
*Get information about the chip.*
- const char \* [gpiod\\_chip\\_get\\_path](#) (struct [gpiod\\_chip](#) \*chip)  
*Get the path used to open the chip.*
- struct [gpiod\\_line\\_info](#) \* [gpiod\\_chip\\_get\\_line\\_info](#) (struct [gpiod\\_chip](#) \*chip, unsigned int offset)  
*Get a snapshot of information about a line.*
- struct [gpiod\\_line\\_info](#) \* [gpiod\\_chip\\_watch\\_line\\_info](#) (struct [gpiod\\_chip](#) \*chip, unsigned int offset)  
*Get a snapshot of the status of a line and start watching it for future changes.*
- int [gpiod\\_chip\\_unwatch\\_line\\_info](#) (struct [gpiod\\_chip](#) \*chip, unsigned int offset)  
*Stop watching a line for status changes.*
- int [gpiod\\_chip\\_get\\_fd](#) (struct [gpiod\\_chip](#) \*chip)  
*Get the file descriptor associated with the chip.*
- int [gpiod\\_chip\\_wait\\_info\\_event](#) (struct [gpiod\\_chip](#) \*chip, int64\_t timeout\_ns)  
*Wait for line status change events on any of the watched lines on the chip.*
- struct [gpiod\\_info\\_event](#) \* [gpiod\\_chip\\_read\\_info\\_event](#) (struct [gpiod\\_chip](#) \*chip)  
*Read a single line status change event from the chip.*
- int [gpiod\\_chip\\_get\\_line\\_offset\\_from\\_name](#) (struct [gpiod\\_chip](#) \*chip, const char \*name)  
*Map a line's name to its offset within the chip.*
- struct [gpiod\\_line\\_request](#) \* [gpiod\\_chip\\_request\\_lines](#) (struct [gpiod\\_chip](#) \*chip, struct [gpiod\\_request\\_config](#) \*req\_cfg, struct [gpiod\\_line\\_config](#) \*line\_cfg)  
*Request a set of lines for exclusive usage.*
- void [gpiod\\_chip\\_info\\_free](#) (struct [gpiod\\_chip\\_info](#) \*info)  
*Free a chip info object and release all associated resources.*
- const char \* [gpiod\\_chip\\_info\\_get\\_name](#) (struct [gpiod\\_chip\\_info](#) \*info)  
*Get the name of the chip as represented in the kernel.*
- const char \* [gpiod\\_chip\\_info\\_get\\_label](#) (struct [gpiod\\_chip\\_info](#) \*info)  
*Get the label of the chip as represented in the kernel.*
- size\_t [gpiod\\_chip\\_info\\_get\\_num\\_lines](#) (struct [gpiod\\_chip\\_info](#) \*info)  
*Get the number of lines exposed by the chip.*
- void [gpiod\\_line\\_info\\_free](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Free a line info object and release all associated resources.*
- struct [gpiod\\_line\\_info](#) \* [gpiod\\_line\\_info\\_copy](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Copy a line info object.*
- unsigned int [gpiod\\_line\\_info\\_get\\_offset](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Get the offset of the line.*
- const char \* [gpiod\\_line\\_info\\_get\\_name](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Get the name of the line.*
- bool [gpiod\\_line\\_info\\_is\\_used](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Check if the line is in use.*
- const char \* [gpiod\\_line\\_info\\_get\\_consumer](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Get the name of the consumer of the line.*
- enum [gpiod\\_line\\_direction](#) [gpiod\\_line\\_info\\_get\\_direction](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Get the direction setting of the line.*
- enum [gpiod\\_line\\_edge](#) [gpiod\\_line\\_info\\_get\\_edge\\_detection](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Get the edge detection setting of the line.*



- enum [gpiod\\_line\\_bias](#) [gpiod\\_line\\_info\\_get\\_bias](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Get the bias setting of the line.*
- enum [gpiod\\_line\\_drive](#) [gpiod\\_line\\_info\\_get\\_drive](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Get the drive setting of the line.*
- bool [gpiod\\_line\\_info\\_is\\_active\\_low](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Check if the logical value of the line is inverted compared to the physical.*
- bool [gpiod\\_line\\_info\\_is\\_debounced](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Check if the line is debounced (either by hardware or by the kernel software debouncer).*
- unsigned long [gpiod\\_line\\_info\\_get\\_debounce\\_period\\_us](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Get the debounce period of the line, in microseconds.*
- enum [gpiod\\_line\\_clock](#) [gpiod\\_line\\_info\\_get\\_event\\_clock](#) (struct [gpiod\\_line\\_info](#) \*info)  
*Get the event clock setting used for edge event timestamps for the line.*
- void [gpiod\\_info\\_event\\_free](#) (struct [gpiod\\_info\\_event](#) \*event)  
*Free the info event object and release all associated resources.*
- enum [gpiod\\_info\\_event\\_type](#) [gpiod\\_info\\_event\\_get\\_event\\_type](#) (struct [gpiod\\_info\\_event](#) \*event)  
*Get the event type of the status change event.*
- uint64\_t [gpiod\\_info\\_event\\_get\\_timestamp\\_ns](#) (struct [gpiod\\_info\\_event](#) \*event)  
*Get the timestamp of the event.*
- struct [gpiod\\_line\\_info](#) \* [gpiod\\_info\\_event\\_get\\_line\\_info](#) (struct [gpiod\\_info\\_event](#) \*event)  
*Get the snapshot of line-info associated with the event.*
- struct [gpiod\\_line\\_settings](#) \* [gpiod\\_line\\_settings\\_new](#) (void)  
*Create a new line settings object.*
- void [gpiod\\_line\\_settings\\_free](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Free the line settings object and release all associated resources.*
- void [gpiod\\_line\\_settings\\_reset](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Reset the line settings object to its default values.*
- struct [gpiod\\_line\\_settings](#) \* [gpiod\\_line\\_settings\\_copy](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Copy the line settings object.*
- int [gpiod\\_line\\_settings\\_set\\_direction](#) (struct [gpiod\\_line\\_settings](#) \*settings, enum [gpiod\\_line\\_direction](#) direction)  
*Set direction.*
- enum [gpiod\\_line\\_direction](#) [gpiod\\_line\\_settings\\_get\\_direction](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Get direction.*
- int [gpiod\\_line\\_settings\\_set\\_edge\\_detection](#) (struct [gpiod\\_line\\_settings](#) \*settings, enum [gpiod\\_line\\_edge](#) edge)  
*Set edge detection.*
- enum [gpiod\\_line\\_edge](#) [gpiod\\_line\\_settings\\_get\\_edge\\_detection](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Get edge detection.*
- int [gpiod\\_line\\_settings\\_set\\_bias](#) (struct [gpiod\\_line\\_settings](#) \*settings, enum [gpiod\\_line\\_bias](#) bias)  
*Set bias.*
- enum [gpiod\\_line\\_bias](#) [gpiod\\_line\\_settings\\_get\\_bias](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Get bias.*
- int [gpiod\\_line\\_settings\\_set\\_drive](#) (struct [gpiod\\_line\\_settings](#) \*settings, enum [gpiod\\_line\\_drive](#) drive)  
*Set drive.*
- enum [gpiod\\_line\\_drive](#) [gpiod\\_line\\_settings\\_get\\_drive](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Get drive.*
- void [gpiod\\_line\\_settings\\_set\\_active\\_low](#) (struct [gpiod\\_line\\_settings](#) \*settings, bool active\_low)  
*Set active-low setting.*
- bool [gpiod\\_line\\_settings\\_get\\_active\\_low](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Get active-low setting.*

- void [gpiod\\_line\\_settings\\_set\\_debounce\\_period\\_us](#) (struct [gpiod\\_line\\_settings](#) \*settings, unsigned long period)  
*Set debounce period.*
- unsigned long [gpiod\\_line\\_settings\\_get\\_debounce\\_period\\_us](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Get debounce period.*
- int [gpiod\\_line\\_settings\\_set\\_event\\_clock](#) (struct [gpiod\\_line\\_settings](#) \*settings, enum [gpiod\\_line\\_clock](#) event↵\_clock)  
*Set event clock.*
- enum [gpiod\\_line\\_clock](#) [gpiod\\_line\\_settings\\_get\\_event\\_clock](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Get event clock setting.*
- int [gpiod\\_line\\_settings\\_set\\_output\\_value](#) (struct [gpiod\\_line\\_settings](#) \*settings, enum [gpiod\\_line\\_value](#) value)  
*Set the output value.*
- enum [gpiod\\_line\\_value](#) [gpiod\\_line\\_settings\\_get\\_output\\_value](#) (struct [gpiod\\_line\\_settings](#) \*settings)  
*Get the output value.*
- struct [gpiod\\_line\\_config](#) \* [gpiod\\_line\\_config\\_new](#) (void)  
*Create a new line config object.*
- void [gpiod\\_line\\_config\\_free](#) (struct [gpiod\\_line\\_config](#) \*config)  
*Free the line config object and release all associated resources.*
- void [gpiod\\_line\\_config\\_reset](#) (struct [gpiod\\_line\\_config](#) \*config)  
*Reset the line config object.*
- int [gpiod\\_line\\_config\\_add\\_line\\_settings](#) (struct [gpiod\\_line\\_config](#) \*config, const unsigned int \*offsets, size\_t num\_offsets, struct [gpiod\\_line\\_settings](#) \*settings)  
*Add line settings for a set of offsets.*
- struct [gpiod\\_line\\_settings](#) \* [gpiod\\_line\\_config\\_get\\_line\\_settings](#) (struct [gpiod\\_line\\_config](#) \*config, unsigned int offset)  
*Get line settings for offset.*
- int [gpiod\\_line\\_config\\_set\\_output\\_values](#) (struct [gpiod\\_line\\_config](#) \*config, const enum [gpiod\\_line\\_value](#) \*values, size\_t num\_values)  
*Set output values for a number of lines.*
- size\_t [gpiod\\_line\\_config\\_get\\_num\\_configured\\_offsets](#) (struct [gpiod\\_line\\_config](#) \*config)  
*Get the number of configured line offsets.*
- size\_t [gpiod\\_line\\_config\\_get\\_configured\\_offsets](#) (struct [gpiod\\_line\\_config](#) \*config, unsigned int \*offsets, size\_t max\_offsets)  
*Get configured offsets.*
- struct [gpiod\\_request\\_config](#) \* [gpiod\\_request\\_config\\_new](#) (void)  
*Create a new request config object.*
- void [gpiod\\_request\\_config\\_free](#) (struct [gpiod\\_request\\_config](#) \*config)  
*Free the request config object and release all associated resources.*
- void [gpiod\\_request\\_config\\_set\\_consumer](#) (struct [gpiod\\_request\\_config](#) \*config, const char \*consumer)  
*Set the consumer name for the request.*
- const char \* [gpiod\\_request\\_config\\_get\\_consumer](#) (struct [gpiod\\_request\\_config](#) \*config)  
*Get the consumer name configured in the request config.*
- void [gpiod\\_request\\_config\\_set\\_event\\_buffer\\_size](#) (struct [gpiod\\_request\\_config](#) \*config, size\_t event\_buffer↵\_size)  
*Set the size of the kernel event buffer for the request.*
- size\_t [gpiod\\_request\\_config\\_get\\_event\\_buffer\\_size](#) (struct [gpiod\\_request\\_config](#) \*config)  
*Get the edge event buffer size for the request config.*
- void [gpiod\\_line\\_request\\_release](#) (struct [gpiod\\_line\\_request](#) \*request)  
*Release the requested lines and free all associated resources.*
- const char \* [gpiod\\_line\\_request\\_get\\_chip\\_name](#) (struct [gpiod\\_line\\_request](#) \*request)  
*Get the name of the chip this request was made on.*
- size\_t [gpiod\\_line\\_request\\_get\\_num\\_requested\\_lines](#) (struct [gpiod\\_line\\_request](#) \*request)

- Get the number of lines in the request.*
- `size_t gpiod_line_request_get_requested_offsets` (struct `gpiod_line_request` \*request, unsigned int \*offsets, size\_t max\_offsets)
- Get the offsets of the lines in the request.*
- `enum gpiod_line_value gpiod_line_request_get_value` (struct `gpiod_line_request` \*request, unsigned int offset)
- Get the value of a single requested line.*
- `int gpiod_line_request_get_values_subset` (struct `gpiod_line_request` \*request, size\_t num\_values, const unsigned int \*offsets, enum `gpiod_line_value` \*values)
- Get the values of a subset of requested lines.*
- `int gpiod_line_request_get_values` (struct `gpiod_line_request` \*request, enum `gpiod_line_value` \*values)
- Get the values of all requested lines.*
- `int gpiod_line_request_set_value` (struct `gpiod_line_request` \*request, unsigned int offset, enum `gpiod_line_value` value)
- Set the value of a single requested line.*
- `int gpiod_line_request_set_values_subset` (struct `gpiod_line_request` \*request, size\_t num\_values, const unsigned int \*offsets, const enum `gpiod_line_value` \*values)
- Set the values of a subset of requested lines.*
- `int gpiod_line_request_set_values` (struct `gpiod_line_request` \*request, const enum `gpiod_line_value` \*values)
- Set the values of all lines associated with a request.*
- `int gpiod_line_request_reconfigure_lines` (struct `gpiod_line_request` \*request, struct `gpiod_line_config` \*config)
- Update the configuration of lines associated with a line request.*
- `int gpiod_line_request_get_fd` (struct `gpiod_line_request` \*request)
- Get the file descriptor associated with a line request.*
- `int gpiod_line_request_wait_edge_events` (struct `gpiod_line_request` \*request, int64\_t timeout\_ns)
- Wait for edge events on any of the requested lines.*
- `int gpiod_line_request_read_edge_events` (struct `gpiod_line_request` \*request, struct `gpiod_edge_event_buffer` \*buffer, size\_t max\_events)
- Read a number of edge events from a line request.*
- `void gpiod_edge_event_free` (struct `gpiod_edge_event` \*event)
- Free the edge event object.*
- `struct gpiod_edge_event * gpiod_edge_event_copy` (struct `gpiod_edge_event` \*event)
- Copy the edge event object.*
- `enum gpiod_edge_event_type gpiod_edge_event_get_event_type` (struct `gpiod_edge_event` \*event)
- Get the event type.*
- `uint64_t gpiod_edge_event_get_timestamp_ns` (struct `gpiod_edge_event` \*event)
- Get the timestamp of the event.*
- `unsigned int gpiod_edge_event_get_line_offset` (struct `gpiod_edge_event` \*event)
- Get the offset of the line which triggered the event.*
- `unsigned long gpiod_edge_event_get_global_seqno` (struct `gpiod_edge_event` \*event)
- Get the global sequence number of the event.*
- `unsigned long gpiod_edge_event_get_line_seqno` (struct `gpiod_edge_event` \*event)
- Get the event sequence number specific to the line.*
- `struct gpiod_edge_event_buffer * gpiod_edge_event_buffer_new` (size\_t capacity)
- Create a new edge event buffer.*
- `size_t gpiod_edge_event_buffer_get_capacity` (struct `gpiod_edge_event_buffer` \*buffer)
- Get the capacity (the max number of events that can be stored) of the event buffer.*
- `void gpiod_edge_event_buffer_free` (struct `gpiod_edge_event_buffer` \*buffer)
- Free the edge event buffer and release all associated resources.*

- `struct gpiod_edge_event * gpiod_edge_event_buffer_get_event` (`struct gpiod_edge_event_buffer *buffer`, unsigned long index)  
*Get an event stored in the buffer.*
- `size_t gpiod_edge_event_buffer_get_num_events` (`struct gpiod_edge_event_buffer *buffer`)  
*Get the number of events a buffer has stored.*
- `bool gpiod_is_gpiochip_device` (`const char *path`)  
*Check if the file pointed to by path is a GPIO chip character device.*
- `const char * gpiod_api_version` (`void`)  
*Get the API version of the library as a human-readable string.*

## 6.2 gpiod.h

[Go to the documentation of this file.](#)

```

00001 /* SPDX-License-Identifier: LGPL-2.1-or-later */
00002 /* SPDX-FileCopyrightText: 2017-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
00003
00008 #ifndef __LIBGPIOD_GPIOD_H__
00009 #define __LIBGPIOD_GPIOD_H__
00010
00011 #include <stdbool.h>
00012 #include <stddef.h>
00013 #include <stdint.h>
00014
00015 #ifdef __cplusplus
00016 extern "C" {
00017 #endif
00018
00027 struct gpiod_chip;
00028
00037 struct gpiod_chip_info;
00038
00047 struct gpiod_line_info;
00048
00058 struct gpiod_line_settings;
00059
00068 struct gpiod_line_config;
00069
00079 struct gpiod_request_config;
00080
00090 struct gpiod_line_request;
00091
00100 struct gpiod_info_event;
00101
00110 struct gpiod_edge_event;
00111
00121 struct gpiod_edge_event_buffer;
00122
00141 struct gpiod_chip *gpiod_chip_open(const char *path);
00142
00147 void gpiod_chip_close(struct gpiod_chip *chip);
00148
00155 struct gpiod_chip_info *gpiod_chip_get_info(struct gpiod_chip *chip);
00156
00164 const char *gpiod_chip_get_path(struct gpiod_chip *chip);
00165
00173 struct gpiod_line_info *gpiod_chip_get_line_info(struct gpiod_chip *chip,
00174 unsigned int offset);
00175
00186 struct gpiod_line_info *gpiod_chip_watch_line_info(struct gpiod_chip *chip,
00187 unsigned int offset);
00188
00195 int gpiod_chip_unwatch_line_info(struct gpiod_chip *chip, unsigned int offset);
00196
00206 int gpiod_chip_get_fd(struct gpiod_chip *chip);
00207
00219 int gpiod_chip_wait_info_event(struct gpiod_chip *chip, int64_t timeout_ns);
00220
00228 struct gpiod_info_event *gpiod_chip_read_info_event(struct gpiod_chip *chip);
00229
00238 int gpiod_chip_get_line_offset_from_name(struct gpiod_chip *chip,
00239 const char *name);
00240
00249 struct gpiod_line_request *
00250 gpiod_chip_request_lines(struct gpiod_chip *chip,
00251 struct gpiod_request_config *req_cfg,

```

```
00252         struct gpiod_line_config *line_cfg);
00253
00278 void gpiod_chip_info_free(struct gpiod_chip_info *info);
00279
00287 const char *gpiod_chip_info_get_name(struct gpiod_chip_info *info);
00288
00296 const char *gpiod_chip_info_get_label(struct gpiod_chip_info *info);
00297
00303 size_t gpiod_chip_info_get_num_lines(struct gpiod_chip_info *info);
00304
00317 enum gpiod_line_value {
00318     GPIOD_LINE_VALUE_ERROR = -1,
00320     GPIOD_LINE_VALUE_INACTIVE = 0,
00322     GPIOD_LINE_VALUE_ACTIVE = 1,
00324 };
00325
00329 enum gpiod_line_direction {
00330     GPIOD_LINE_DIRECTION_AS_IS = 1,
00332     GPIOD_LINE_DIRECTION_INPUT,
00335     GPIOD_LINE_DIRECTION_OUTPUT,
00337 };
00338
00342 enum gpiod_line_edge {
00343     GPIOD_LINE_EDGE_NONE = 1,
00345     GPIOD_LINE_EDGE_RISING,
00347     GPIOD_LINE_EDGE_FALLING,
00349     GPIOD_LINE_EDGE_BOTH,
00351 };
00352
00356 enum gpiod_line_bias {
00357     GPIOD_LINE_BIAS_AS_IS = 1,
00359     GPIOD_LINE_BIAS_UNKNOWN,
00361     GPIOD_LINE_BIAS_DISABLED,
00363     GPIOD_LINE_BIAS_PULL_UP,
00365     GPIOD_LINE_BIAS_PULL_DOWN,
00367 };
00368
00372 enum gpiod_line_drive {
00373     GPIOD_LINE_DRIVE_PUSH_PULL = 1,
00375     GPIOD_LINE_DRIVE_OPEN_DRAIN,
00377     GPIOD_LINE_DRIVE_OPEN_SOURCE,
00379 };
00380
00384 enum gpiod_line_clock {
00385     GPIOD_LINE_CLOCK_MONOTONIC = 1,
00387     GPIOD_LINE_CLOCK_REALTIME,
00389     GPIOD_LINE_CLOCK_HTE,
00391 };
00392
00419 void gpiod_line_info_free(struct gpiod_line_info *info);
00420
00427 struct gpiod_line_info *gpiod_line_info_copy(struct gpiod_line_info *info);
00428
00437 unsigned int gpiod_line_info_get_offset(struct gpiod_line_info *info);
00438
00447 const char *gpiod_line_info_get_name(struct gpiod_line_info *info);
00448
00459 bool gpiod_line_info_is_used(struct gpiod_line_info *info);
00460
00470 const char *gpiod_line_info_get_consumer(struct gpiod_line_info *info);
00471
00478 enum gpiod_line_direction
00479 gpiod_line_info_get_direction(struct gpiod_line_info *info);
00480
00487 enum gpiod_line_edge
00488 gpiod_line_info_get_edge_detection(struct gpiod_line_info *info);
00489
00496 enum gpiod_line_bias
00497 gpiod_line_info_get_bias(struct gpiod_line_info *info);
00498
00505 enum gpiod_line_drive
00506 gpiod_line_info_get_drive(struct gpiod_line_info *info);
00507
00514 bool gpiod_line_info_is_active_low(struct gpiod_line_info *info);
00515
00522 bool gpiod_line_info_is_debounced(struct gpiod_line_info *info);
00523
00530 unsigned long
00531 gpiod_line_info_get_debounce_period_us(struct gpiod_line_info *info);
00532
00540 enum gpiod_line_clock
00541 gpiod_line_info_get_event_clock(struct gpiod_line_info *info);
00542
00561 enum gpiod_info_event_type {
00562     GPIOD_INFO_EVENT_LINE_REQUESTED = 1,
00564     GPIOD_INFO_EVENT_LINE_RELEASED,
00566     GPIOD_INFO_EVENT_LINE_CONFIG_CHANGED,
```

```
00568 };
00569
00574 void gpio_info_event_free(struct gpio_info_event *event);
00575
00583 enum gpio_info_event_type
00584 gpio_info_event_get_event_type(struct gpio_info_event *event);
00585
00591 uint64_t gpio_info_event_get_timestamp_ns(struct gpio_info_event *event);
00592
00606 struct gpio_line_info *
00607 gpio_info_event_get_line_info(struct gpio_info_event *event);
00608
00631 struct gpio_line_settings *gpio_line_settings_new(void);
00632
00637 void gpio_line_settings_free(struct gpio_line_settings *settings);
00638
00643 void gpio_line_settings_reset(struct gpio_line_settings *settings);
00644
00651 struct gpio_line_settings *
00652 gpio_line_settings_copy(struct gpio_line_settings *settings);
00653
00660 int gpio_line_settings_set_direction(struct gpio_line_settings *settings,
00661                                     enum gpio_line_direction direction);
00662
00668 enum gpio_line_direction
00669 gpio_line_settings_get_direction(struct gpio_line_settings *settings);
00670
00677 int gpio_line_settings_set_edge_detection(struct gpio_line_settings *settings,
00678                                           enum gpio_line_edge edge);
00679
00685 enum gpio_line_edge
00686 gpio_line_settings_get_edge_detection(struct gpio_line_settings *settings);
00687
00694 int gpio_line_settings_set_bias(struct gpio_line_settings *settings,
00695                                enum gpio_line_bias bias);
00696
00702 enum gpio_line_bias
00703 gpio_line_settings_get_bias(struct gpio_line_settings *settings);
00704
00711 int gpio_line_settings_set_drive(struct gpio_line_settings *settings,
00712                                 enum gpio_line_drive drive);
00713
00719 enum gpio_line_drive
00720 gpio_line_settings_get_drive(struct gpio_line_settings *settings);
00721
00727 void gpio_line_settings_set_active_low(struct gpio_line_settings *settings,
00728                                       bool active_low);
00729
00735 bool gpio_line_settings_get_active_low(struct gpio_line_settings *settings);
00736
00742 void
00743 gpio_line_settings_set_debounce_period_us(struct gpio_line_settings *settings,
00744                                           unsigned long period);
00745
00751 unsigned long
00752 gpio_line_settings_get_debounce_period_us(
00753     struct gpio_line_settings *settings);
00754
00761 int gpio_line_settings_set_event_clock(struct gpio_line_settings *settings,
00762                                       enum gpio_line_clock event_clock);
00763
00769 enum gpio_line_clock
00770 gpio_line_settings_get_event_clock(struct gpio_line_settings *settings);
00771
00778 int gpio_line_settings_set_output_value(struct gpio_line_settings *settings,
00779                                         enum gpio_line_value value);
00780
00786 enum gpio_line_value
00787 gpio_line_settings_get_output_value(struct gpio_line_settings *settings);
00788
00817 struct gpio_line_config *gpio_line_config_new(void);
00818
00823 void gpio_line_config_free(struct gpio_line_config *config);
00824
00832 void gpio_line_config_reset(struct gpio_line_config *config);
00833
00842 int gpio_line_config_add_line_settings(struct gpio_line_config *config,
00843                                       const unsigned int *offsets,
00844                                       size_t num_offsets,
00845                                       struct gpio_line_settings *settings);
00846
00854 struct gpio_line_settings *
00855 gpio_line_config_get_line_settings(struct gpio_line_config *config,
00856                                   unsigned int offset);
00857
00879 int gpio_line_config_set_output_values(struct gpio_line_config *config,
00880                                       const enum gpio_line_value *values,
```

```

00881         size_t num_values);
00882
00888     size_t
00889     gpiod_line_config_get_num_configured_offsets(struct gpiod_line_config *config);
00890
00902     size_t
00903     gpiod_line_config_get_configured_offsets(struct gpiod_line_config *config,
00904         unsigned int *offsets,
00905         size_t max_offsets);
00906
00926     struct gpiod_request_config *gpiod_request_config_new(void);
00927
00932     void gpiod_request_config_free(struct gpiod_request_config *config);
00933
00941     void gpiod_request_config_set_consumer(struct gpiod_request_config *config,
00942         const char *consumer);
00943
00949     const char *
00950     gpiod_request_config_get_consumer(struct gpiod_request_config *config);
00951
00961     void
00962     gpiod_request_config_set_event_buffer_size(struct gpiod_request_config *config,
00963         size_t event_buffer_size);
00964
00970     size_t
00971     gpiod_request_config_get_event_buffer_size(struct gpiod_request_config *config);
00972
00986     void gpiod_line_request_release(struct gpiod_line_request *request);
00987
00994     const char *
00995     gpiod_line_request_get_chip_name(struct gpiod_line_request *request);
00996
01002     size_t
01003     gpiod_line_request_get_num_requested_lines(struct gpiod_line_request *request);
01004
01016     size_t
01017     gpiod_line_request_get_requested_offsets(struct gpiod_line_request *request,
01018         unsigned int *offsets,
01019         size_t max_offsets);
01020
01027     enum gpiod_line_value
01028     gpiod_line_request_get_value(struct gpiod_line_request *request,
01029         unsigned int offset);
01030
01042     int gpiod_line_request_get_values_subset(struct gpiod_line_request *request,
01043         size_t num_values,
01044         const unsigned int *offsets,
01045         enum gpiod_line_value *values);
01046
01058     int gpiod_line_request_get_values(struct gpiod_line_request *request,
01059         enum gpiod_line_value *values);
01060
01068     int gpiod_line_request_set_value(struct gpiod_line_request *request,
01069         unsigned int offset,
01070         enum gpiod_line_value value);
01071
01084     int gpiod_line_request_set_values_subset(struct gpiod_line_request *request,
01085         size_t num_values,
01086         const unsigned int *offsets,
01087         const enum gpiod_line_value *values);
01088
01100     int gpiod_line_request_set_values(struct gpiod_line_request *request,
01101         const enum gpiod_line_value *values);
01102
01114     int gpiod_line_request_reconfigure_lines(struct gpiod_line_request *request,
01115         struct gpiod_line_config *config);
01116
01125     int gpiod_line_request_get_fd(struct gpiod_line_request *request);
01126
01140     int gpiod_line_request_wait_edge_events(struct gpiod_line_request *request,
01141         int64_t timeout_ns);
01142
01154     int gpiod_line_request_read_edge_events(struct gpiod_line_request *request,
01155         struct gpiod_edge_event_buffer *buffer,
01156         size_t max_events);
01157
01179     enum gpiod_edge_event_type {
01180         GPIOD_EDGE_EVENT_RISING_EDGE = 1,
01182         GPIOD_EDGE_EVENT_FALLING_EDGE,
01184     };
01185
01190     void gpiod_edge_event_free(struct gpiod_edge_event *event);
01191
01198     struct gpiod_edge_event *gpiod_edge_event_copy(struct gpiod_edge_event *event);
01199
01206     enum gpiod_edge_event_type
01207     gpiod_edge_event_get_event_type(struct gpiod_edge_event *event);

```

```
01208
01216 uint64_t gpiod_edge_event_get_timestamp_ns(struct gpiod_edge_event *event);
01217
01223 unsigned int gpiod_edge_event_get_line_offset(struct gpiod_edge_event *event);
01224
01231 unsigned long gpiod_edge_event_get_global_seqno(struct gpiod_edge_event *event);
01232
01239 unsigned long gpiod_edge_event_get_line_seqno(struct gpiod_edge_event *event);
01240
01253 struct gpiod_edge_event_buffer *
01254 gpiod_edge_event_buffer_new(size_t capacity);
01255
01262 size_t
01263 gpiod_edge_event_buffer_get_capacity(struct gpiod_edge_event_buffer *buffer);
01264
01269 void gpiod_edge_event_buffer_free(struct gpiod_edge_event_buffer *buffer);
01270
01285 struct gpiod_edge_event *
01286 gpiod_edge_event_buffer_get_event(struct gpiod_edge_event_buffer *buffer,
01287                                 unsigned long index);
01288
01294 size_t
01295 gpiod_edge_event_buffer_get_num_events(struct gpiod_edge_event_buffer *buffer);
01296
01312 bool gpiod_is_gpiochip_device(const char *path);
01313
01320 const char *gpiod_api_version(void);
01321
01326 #ifdef __cplusplus
01327 } /* extern "C" */
01328 #endif
01329
01330 #endif /* __LIBGPIOD_GPIOD_H__ */
```



# Index

- Chip info, [12](#)
  - [gpiod\\_chip\\_info\\_free](#), [12](#)
  - [gpiod\\_chip\\_info\\_get\\_label](#), [12](#)
  - [gpiod\\_chip\\_info\\_get\\_name](#), [12](#)
  - [gpiod\\_chip\\_info\\_get\\_num\\_lines](#), [13](#)
- GPIO chips, [7](#)
  - [gpiod\\_chip\\_close](#), [8](#)
  - [gpiod\\_chip\\_get\\_fd](#), [8](#)
  - [gpiod\\_chip\\_get\\_info](#), [8](#)
  - [gpiod\\_chip\\_get\\_line\\_info](#), [8](#)
  - [gpiod\\_chip\\_get\\_line\\_offset\\_from\\_name](#), [9](#)
  - [gpiod\\_chip\\_get\\_path](#), [9](#)
  - [gpiod\\_chip\\_open](#), [9](#)
  - [gpiod\\_chip\\_read\\_info\\_event](#), [10](#)
  - [gpiod\\_chip\\_request\\_lines](#), [10](#)
  - [gpiod\\_chip\\_unwatch\\_line\\_info](#), [10](#)
  - [gpiod\\_chip\\_wait\\_info\\_event](#), [11](#)
  - [gpiod\\_chip\\_watch\\_line\\_info](#), [11](#)
- [gpiod.h](#), [55](#)
- [gpiod\\_api\\_version](#)
  - Stuff that didn't fit anywhere else, [49](#)
- [gpiod\\_chip](#), [51](#)
- [gpiod\\_chip\\_close](#)
  - GPIO chips, [8](#)
- [gpiod\\_chip\\_get\\_fd](#)
  - GPIO chips, [8](#)
- [gpiod\\_chip\\_get\\_info](#)
  - GPIO chips, [8](#)
- [gpiod\\_chip\\_get\\_line\\_info](#)
  - GPIO chips, [8](#)
- [gpiod\\_chip\\_get\\_line\\_offset\\_from\\_name](#)
  - GPIO chips, [9](#)
- [gpiod\\_chip\\_get\\_path](#)
  - GPIO chips, [9](#)
- [gpiod\\_chip\\_info](#), [51](#)
- [gpiod\\_chip\\_info\\_free](#)
  - Chip info, [12](#)
- [gpiod\\_chip\\_info\\_get\\_label](#)
  - Chip info, [12](#)
- [gpiod\\_chip\\_info\\_get\\_name](#)
  - Chip info, [12](#)
- [gpiod\\_chip\\_info\\_get\\_num\\_lines](#)
  - Chip info, [13](#)
- [gpiod\\_chip\\_open](#)
  - GPIO chips, [9](#)
- [gpiod\\_chip\\_read\\_info\\_event](#)
  - GPIO chips, [10](#)
- [gpiod\\_chip\\_request\\_lines](#)
  - GPIO chips, [10](#)
- [gpiod\\_chip\\_unwatch\\_line\\_info](#)
  - GPIO chips, [10](#)
- [gpiod\\_chip\\_wait\\_info\\_event](#)
  - GPIO chips, [11](#)
- [gpiod\\_chip\\_watch\\_line\\_info](#)
  - GPIO chips, [11](#)
- [gpiod\\_edge\\_event](#), [51](#)
- [gpiod\\_edge\\_event\\_buffer](#), [52](#)
- [gpiod\\_edge\\_event\\_buffer\\_free](#)
  - Line edge events handling, [44](#)
- [gpiod\\_edge\\_event\\_buffer\\_get\\_capacity](#)
  - Line edge events handling, [44](#)
- [gpiod\\_edge\\_event\\_buffer\\_get\\_event](#)
  - Line edge events handling, [44](#)
- [gpiod\\_edge\\_event\\_buffer\\_get\\_num\\_events](#)
  - Line edge events handling, [44](#)
- [gpiod\\_edge\\_event\\_buffer\\_new](#)
  - Line edge events handling, [45](#)
- [gpiod\\_edge\\_event\\_copy](#)
  - Line edge events handling, [45](#)
- [GPIOD\\_EDGE\\_EVENT\\_FALLING\\_EDGE](#)
  - Line edge events handling, [44](#)
- [gpiod\\_edge\\_event\\_free](#)
  - Line edge events handling, [45](#)
- [gpiod\\_edge\\_event\\_get\\_event\\_type](#)
  - Line edge events handling, [47](#)
- [gpiod\\_edge\\_event\\_get\\_global\\_seqno](#)
  - Line edge events handling, [47](#)
- [gpiod\\_edge\\_event\\_get\\_line\\_offset](#)
  - Line edge events handling, [47](#)
- [gpiod\\_edge\\_event\\_get\\_line\\_seqno](#)
  - Line edge events handling, [47](#)
- [gpiod\\_edge\\_event\\_get\\_timestamp\\_ns](#)
  - Line edge events handling, [48](#)
- [GPIOD\\_EDGE\\_EVENT\\_RISING\\_EDGE](#)
  - Line edge events handling, [44](#)
- [gpiod\\_edge\\_event\\_type](#)
  - Line edge events handling, [43](#)
- [gpiod\\_info\\_event](#), [52](#)
- [gpiod\\_info\\_event\\_free](#)
  - Line status watch events, [22](#)
- [gpiod\\_info\\_event\\_get\\_event\\_type](#)
  - Line status watch events, [22](#)
- [gpiod\\_info\\_event\\_get\\_line\\_info](#)
  - Line status watch events, [23](#)
- [gpiod\\_info\\_event\\_get\\_timestamp\\_ns](#)
  - Line status watch events, [23](#)
- [GPIOD\\_INFO\\_EVENT\\_LINE\\_CONFIG\\_CHANGED](#)
  - Line status watch events, [22](#)

- GPIOD\_INFO\_EVENT\_LINE\_RELEASED
  - Line status watch events, [22](#)
- GPIOD\_INFO\_EVENT\_LINE\_REQUESTED
  - Line status watch events, [22](#)
- gpiod\_info\_event\_type
  - Line status watch events, [22](#)
- gpiod\_is\_gpiochip\_device
  - Stuff that didn't fit anywhere else, [49](#)
- gpiod\_line\_bias
  - Line definitions, [14](#)
- GPIOD\_LINE\_BIAS\_AS\_IS
  - Line definitions, [14](#)
- GPIOD\_LINE\_BIAS\_DISABLED
  - Line definitions, [14](#)
- GPIOD\_LINE\_BIAS\_PULL\_DOWN
  - Line definitions, [14](#)
- GPIOD\_LINE\_BIAS\_PULL\_UP
  - Line definitions, [14](#)
- GPIOD\_LINE\_BIAS\_UNKNOWN
  - Line definitions, [14](#)
- gpiod\_line\_clock
  - Line definitions, [14](#)
- GPIOD\_LINE\_CLOCK\_HTE
  - Line definitions, [14](#)
- GPIOD\_LINE\_CLOCK\_MONOTONIC
  - Line definitions, [14](#)
- GPIOD\_LINE\_CLOCK\_REALTIME
  - Line definitions, [14](#)
- gpiod\_line\_config, [52](#)
- gpiod\_line\_config\_add\_line\_settings
  - Line configuration objects, [32](#)
- gpiod\_line\_config\_free
  - Line configuration objects, [32](#)
- gpiod\_line\_config\_get\_configured\_offsets
  - Line configuration objects, [32](#)
- gpiod\_line\_config\_get\_line\_settings
  - Line configuration objects, [32](#)
- gpiod\_line\_config\_get\_num\_configured\_offsets
  - Line configuration objects, [33](#)
- gpiod\_line\_config\_new
  - Line configuration objects, [33](#)
- gpiod\_line\_config\_reset
  - Line configuration objects, [33](#)
- gpiod\_line\_config\_set\_output\_values
  - Line configuration objects, [34](#)
- gpiod\_line\_direction
  - Line definitions, [14](#)
- GPIOD\_LINE\_DIRECTION\_AS\_IS
  - Line definitions, [14](#)
- GPIOD\_LINE\_DIRECTION\_INPUT
  - Line definitions, [14](#)
- GPIOD\_LINE\_DIRECTION\_OUTPUT
  - Line definitions, [14](#)
- gpiod\_line\_drive
  - Line definitions, [14](#)
- GPIOD\_LINE\_DRIVE\_OPEN\_DRAIN
  - Line definitions, [15](#)
- GPIOD\_LINE\_DRIVE\_OPEN\_SOURCE
  - Line definitions, [15](#)
- GPIOD\_LINE\_DRIVE\_PUSH\_PULL
  - Line definitions, [15](#)
- gpiod\_line\_edge
  - Line definitions, [15](#)
- GPIOD\_LINE\_EDGE\_BOTH
  - Line definitions, [15](#)
- GPIOD\_LINE\_EDGE\_FALLING
  - Line definitions, [15](#)
- GPIOD\_LINE\_EDGE\_NONE
  - Line definitions, [15](#)
- GPIOD\_LINE\_EDGE\_RISING
  - Line definitions, [15](#)
- gpiod\_line\_info, [53](#)
- gpiod\_line\_info\_copy
  - Line info, [16](#)
- gpiod\_line\_info\_free
  - Line info, [16](#)
- gpiod\_line\_info\_get\_bias
  - Line info, [17](#)
- gpiod\_line\_info\_get\_consumer
  - Line info, [17](#)
- gpiod\_line\_info\_get\_debounce\_period\_us
  - Line info, [17](#)
- gpiod\_line\_info\_get\_direction
  - Line info, [17](#)
- gpiod\_line\_info\_get\_drive
  - Line info, [19](#)
- gpiod\_line\_info\_get\_edge\_detection
  - Line info, [19](#)
- gpiod\_line\_info\_get\_event\_clock
  - Line info, [19](#)
- gpiod\_line\_info\_get\_name
  - Line info, [20](#)
- gpiod\_line\_info\_get\_offset
  - Line info, [20](#)
- gpiod\_line\_info\_is\_active\_low
  - Line info, [20](#)
- gpiod\_line\_info\_is\_debounced
  - Line info, [21](#)
- gpiod\_line\_info\_is\_used
  - Line info, [21](#)
- gpiod\_line\_request, [53](#)
- gpiod\_line\_request\_get\_chip\_name
  - Line request operations, [37](#)
- gpiod\_line\_request\_get\_fd
  - Line request operations, [38](#)
- gpiod\_line\_request\_get\_num\_requested\_lines
  - Line request operations, [38](#)
- gpiod\_line\_request\_get\_requested\_offsets
  - Line request operations, [38](#)
- gpiod\_line\_request\_get\_value
  - Line request operations, [39](#)
- gpiod\_line\_request\_get\_values
  - Line request operations, [39](#)
- gpiod\_line\_request\_get\_values\_subset
  - Line request operations, [39](#)
- gpiod\_line\_request\_read\_edge\_events

- Line request operations, [40](#)
- `gpiod_line_request_reconfigure_lines`
  - Line request operations, [40](#)
- `gpiod_line_request_release`
  - Line request operations, [41](#)
- `gpiod_line_request_set_value`
  - Line request operations, [41](#)
- `gpiod_line_request_set_values`
  - Line request operations, [41](#)
- `gpiod_line_request_set_values_subset`
  - Line request operations, [42](#)
- `gpiod_line_request_wait_edge_events`
  - Line request operations, [42](#)
- `gpiod_line_settings`, [53](#)
- `gpiod_line_settings_copy`
  - Line settings objects, [25](#)
- `gpiod_line_settings_free`
  - Line settings objects, [25](#)
- `gpiod_line_settings_get_active_low`
  - Line settings objects, [25](#)
- `gpiod_line_settings_get_bias`
  - Line settings objects, [25](#)
- `gpiod_line_settings_get_debounce_period_us`
  - Line settings objects, [26](#)
- `gpiod_line_settings_get_direction`
  - Line settings objects, [26](#)
- `gpiod_line_settings_get_drive`
  - Line settings objects, [26](#)
- `gpiod_line_settings_get_edge_detection`
  - Line settings objects, [27](#)
- `gpiod_line_settings_get_event_clock`
  - Line settings objects, [27](#)
- `gpiod_line_settings_get_output_value`
  - Line settings objects, [27](#)
- `gpiod_line_settings_new`
  - Line settings objects, [28](#)
- `gpiod_line_settings_reset`
  - Line settings objects, [28](#)
- `gpiod_line_settings_set_active_low`
  - Line settings objects, [28](#)
- `gpiod_line_settings_set_bias`
  - Line settings objects, [28](#)
- `gpiod_line_settings_set_debounce_period_us`
  - Line settings objects, [29](#)
- `gpiod_line_settings_set_direction`
  - Line settings objects, [29](#)
- `gpiod_line_settings_set_drive`
  - Line settings objects, [29](#)
- `gpiod_line_settings_set_edge_detection`
  - Line settings objects, [30](#)
- `gpiod_line_settings_set_event_clock`
  - Line settings objects, [30](#)
- `gpiod_line_settings_set_output_value`
  - Line settings objects, [30](#)
- `gpiod_line_value`
  - Line definitions, [15](#)
- `GPIOD_LINE_VALUE_ACTIVE`
  - Line definitions, [15](#)
- `GPIOD_LINE_VALUE_ERROR`
  - Line definitions, [15](#)
- `GPIOD_LINE_VALUE_INACTIVE`
  - Line definitions, [15](#)
- `gpiod_request_config`, [53](#)
- `gpiod_request_config_free`
  - Request configuration objects, [35](#)
- `gpiod_request_config_get_consumer`
  - Request configuration objects, [35](#)
- `gpiod_request_config_get_event_buffer_size`
  - Request configuration objects, [35](#)
- `gpiod_request_config_new`
  - Request configuration objects, [35](#)
- `gpiod_request_config_set_consumer`
  - Request configuration objects, [36](#)
- `gpiod_request_config_set_event_buffer_size`
  - Request configuration objects, [36](#)
- Line configuration objects, [31](#)
  - `gpiod_line_config_add_line_settings`, [32](#)
  - `gpiod_line_config_free`, [32](#)
  - `gpiod_line_config_get_configured_offsets`, [32](#)
  - `gpiod_line_config_get_line_settings`, [32](#)
  - `gpiod_line_config_get_num_configured_offsets`, [33](#)
  - `gpiod_line_config_new`, [33](#)
  - `gpiod_line_config_reset`, [33](#)
  - `gpiod_line_config_set_output_values`, [34](#)
- Line definitions, [13](#)
  - `gpiod_line_bias`, [14](#)
  - `GPIOD_LINE_BIAS_AS_IS`, [14](#)
  - `GPIOD_LINE_BIAS_DISABLED`, [14](#)
  - `GPIOD_LINE_BIAS_PULL_DOWN`, [14](#)
  - `GPIOD_LINE_BIAS_PULL_UP`, [14](#)
  - `GPIOD_LINE_BIAS_UNKNOWN`, [14](#)
  - `gpiod_line_clock`, [14](#)
  - `GPIOD_LINE_CLOCK_HTE`, [14](#)
  - `GPIOD_LINE_CLOCK_MONOTONIC`, [14](#)
  - `GPIOD_LINE_CLOCK_REALTIME`, [14](#)
  - `gpiod_line_direction`, [14](#)
  - `GPIOD_LINE_DIRECTION_AS_IS`, [14](#)
  - `GPIOD_LINE_DIRECTION_INPUT`, [14](#)
  - `GPIOD_LINE_DIRECTION_OUTPUT`, [14](#)
  - `gpiod_line_drive`, [14](#)
  - `GPIOD_LINE_DRIVE_OPEN_DRAIN`, [15](#)
  - `GPIOD_LINE_DRIVE_OPEN_SOURCE`, [15](#)
  - `GPIOD_LINE_DRIVE_PUSH_PULL`, [15](#)
  - `gpiod_line_edge`, [15](#)
  - `GPIOD_LINE_EDGE_BOTH`, [15](#)
  - `GPIOD_LINE_EDGE_FALLING`, [15](#)
  - `GPIOD_LINE_EDGE_NONE`, [15](#)
  - `GPIOD_LINE_EDGE_RISING`, [15](#)
  - `gpiod_line_value`, [15](#)
  - `GPIOD_LINE_VALUE_ACTIVE`, [15](#)
  - `GPIOD_LINE_VALUE_ERROR`, [15](#)
  - `GPIOD_LINE_VALUE_INACTIVE`, [15](#)
- Line edge events handling, [43](#)
  - `gpiod_edge_event_buffer_free`, [44](#)
  - `gpiod_edge_event_buffer_get_capacity`, [44](#)

- gpiod\_edge\_event\_buffer\_get\_event, 44
- gpiod\_edge\_event\_buffer\_get\_num\_events, 44
- gpiod\_edge\_event\_buffer\_new, 45
- gpiod\_edge\_event\_copy, 45
- GPIOD\_EDGE\_EVENT\_FALLING\_EDGE, 44
- gpiod\_edge\_event\_free, 45
- gpiod\_edge\_event\_get\_event\_type, 47
- gpiod\_edge\_event\_get\_global\_seqno, 47
- gpiod\_edge\_event\_get\_line\_offset, 47
- gpiod\_edge\_event\_get\_line\_seqno, 47
- gpiod\_edge\_event\_get\_timestamp\_ns, 48
- GPIOD\_EDGE\_EVENT\_RISING\_EDGE, 44
- gpiod\_edge\_event\_type, 43
- Line info, 15
  - gpiod\_line\_info\_copy, 16
  - gpiod\_line\_info\_free, 16
  - gpiod\_line\_info\_get\_bias, 17
  - gpiod\_line\_info\_get\_consumer, 17
  - gpiod\_line\_info\_get\_debounce\_period\_us, 17
  - gpiod\_line\_info\_get\_direction, 17
  - gpiod\_line\_info\_get\_drive, 19
  - gpiod\_line\_info\_get\_edge\_detection, 19
  - gpiod\_line\_info\_get\_event\_clock, 19
  - gpiod\_line\_info\_get\_name, 20
  - gpiod\_line\_info\_get\_offset, 20
  - gpiod\_line\_info\_is\_active\_low, 20
  - gpiod\_line\_info\_is\_debounced, 21
  - gpiod\_line\_info\_is\_used, 21
- Line request operations, 37
  - gpiod\_line\_request\_get\_chip\_name, 37
  - gpiod\_line\_request\_get\_fd, 38
  - gpiod\_line\_request\_get\_num\_requested\_lines, 38
  - gpiod\_line\_request\_get\_requested\_offsets, 38
  - gpiod\_line\_request\_get\_value, 39
  - gpiod\_line\_request\_get\_values, 39
  - gpiod\_line\_request\_get\_values\_subset, 39
  - gpiod\_line\_request\_read\_edge\_events, 40
  - gpiod\_line\_request\_reconfigure\_lines, 40
  - gpiod\_line\_request\_release, 41
  - gpiod\_line\_request\_set\_value, 41
  - gpiod\_line\_request\_set\_values, 41
  - gpiod\_line\_request\_set\_values\_subset, 42
  - gpiod\_line\_request\_wait\_edge\_events, 42
- Line settings objects, 24
  - gpiod\_line\_settings\_copy, 25
  - gpiod\_line\_settings\_free, 25
  - gpiod\_line\_settings\_get\_active\_low, 25
  - gpiod\_line\_settings\_get\_bias, 25
  - gpiod\_line\_settings\_get\_debounce\_period\_us, 26
  - gpiod\_line\_settings\_get\_direction, 26
  - gpiod\_line\_settings\_get\_drive, 26
  - gpiod\_line\_settings\_get\_edge\_detection, 27
  - gpiod\_line\_settings\_get\_event\_clock, 27
  - gpiod\_line\_settings\_get\_output\_value, 27
  - gpiod\_line\_settings\_new, 28
  - gpiod\_line\_settings\_reset, 28
  - gpiod\_line\_settings\_set\_active\_low, 28
  - gpiod\_line\_settings\_set\_bias, 28
  - gpiod\_line\_settings\_set\_debounce\_period\_us, 29
  - gpiod\_line\_settings\_set\_direction, 29
  - gpiod\_line\_settings\_set\_drive, 29
  - gpiod\_line\_settings\_set\_edge\_detection, 30
  - gpiod\_line\_settings\_set\_event\_clock, 30
  - gpiod\_line\_settings\_set\_output\_value, 30
- Line status watch events, 21
  - gpiod\_info\_event\_free, 22
  - gpiod\_info\_event\_get\_event\_type, 22
  - gpiod\_info\_event\_get\_line\_info, 23
  - gpiod\_info\_event\_get\_timestamp\_ns, 23
  - GPIOD\_INFO\_EVENT\_LINE\_CONFIG\_CHANGED, 22
  - GPIOD\_INFO\_EVENT\_LINE\_RELEASED, 22
  - GPIOD\_INFO\_EVENT\_LINE\_REQUESTED, 22
  - gpiod\_info\_event\_type, 22
- Request configuration objects, 34
  - gpiod\_request\_config\_free, 35
  - gpiod\_request\_config\_get\_consumer, 35
  - gpiod\_request\_config\_get\_event\_buffer\_size, 35
  - gpiod\_request\_config\_new, 35
  - gpiod\_request\_config\_set\_consumer, 36
  - gpiod\_request\_config\_set\_event\_buffer\_size, 36
- Stuff that didn't fit anywhere else, 48
  - gpiod\_api\_version, 49
  - gpiod\_is\_gpiochip\_device, 49