

libgpod
2.0-devel

Generated by Doxygen 1.8.13

Contents

1	libgpod public API	1
2	Module Index	3
2.1	Modules	3
3	Data Structure Index	5
3.1	Data Structures	5
4	File Index	7
4.1	File List	7
5	Module Documentation	9
5.1	Common helper macros	9
5.1.1	Detailed Description	9
5.1.2	Macro Definition Documentation	9
5.1.2.1	GPIO_BIT	9
5.2	GPIO chip operations	11
5.2.1	Detailed Description	11
5.2.2	Function Documentation	11
5.2.2.1	gpod_chip_close()	11
5.2.2.2	gpod_chip_find_line()	12
5.2.2.3	gpod_chip_find_line_unique()	12
5.2.2.4	gpod_chip_get_all_lines()	12
5.2.2.5	gpod_chip_get_line()	13
5.2.2.6	gpod_chip_get_lines()	13

5.2.2.7	<code>gpiod_chip_label()</code>	14
5.2.2.8	<code>gpiod_chip_name()</code>	14
5.2.2.9	<code>gpiod_chip_num_lines()</code>	14
5.2.2.10	<code>gpiod_chip_open()</code>	15
5.2.2.11	<code>gpiod_is_gpiochip_device()</code>	15
5.3	GPIO line operations	16
5.3.1	Detailed Description	16
5.4	Operating on multiple lines	17
5.4.1	Detailed Description	17
5.4.2	Typedef Documentation	18
5.4.2.1	<code>gpiod_line_bulk_foreach_cb</code>	18
5.4.3	Enumeration Type Documentation	18
5.4.3.1	anonymous enum	18
5.4.4	Function Documentation	18
5.4.4.1	<code>gpiod_line_bulk_add_line()</code>	18
5.4.4.2	<code>gpiod_line_bulk_foreach_line()</code>	19
5.4.4.3	<code>gpiod_line_bulk_free()</code>	19
5.4.4.4	<code>gpiod_line_bulk_get_line()</code>	19
5.4.4.5	<code>gpiod_line_bulk_new()</code>	20
5.4.4.6	<code>gpiod_line_bulk_num_lines()</code>	20
5.4.4.7	<code>gpiod_line_bulk_reset()</code>	21
5.5	Line info	22
5.5.1	Detailed Description	23
5.5.2	Enumeration Type Documentation	23
5.5.2.1	anonymous enum	23
5.5.2.2	anonymous enum	23
5.5.2.3	anonymous enum	23
5.5.3	Function Documentation	24
5.5.3.1	<code>gpiod_line_bias()</code>	24
5.5.3.2	<code>gpiod_line_consumer()</code>	24

5.5.3.3	gpiod_line_direction()	25
5.5.3.4	gpiod_line_drive()	25
5.5.3.5	gpiod_line_get_chip()	25
5.5.3.6	gpiod_line_is_active_low()	26
5.5.3.7	gpiod_line_is_used()	26
5.5.3.8	gpiod_line_name()	26
5.5.3.9	gpiod_line_offset()	27
5.5.3.10	gpiod_line_update()	27
5.6	Line requests	29
5.6.1	Detailed Description	31
5.6.2	Enumeration Type Documentation	31
5.6.2.1	anonymous enum	31
5.6.2.2	anonymous enum	31
5.6.3	Function Documentation	32
5.6.3.1	gpiod_line_is_free()	32
5.6.3.2	gpiod_line_is_requested()	32
5.6.3.3	gpiod_line_release()	32
5.6.3.4	gpiod_line_release_bulk()	33
5.6.3.5	gpiod_line_request()	33
5.6.3.6	gpiod_line_request_both_edges_events()	33
5.6.3.7	gpiod_line_request_both_edges_events_flags()	34
5.6.3.8	gpiod_line_request_bulk()	34
5.6.3.9	gpiod_line_request_bulk_both_edges_events()	35
5.6.3.10	gpiod_line_request_bulk_both_edges_events_flags()	35
5.6.3.11	gpiod_line_request_bulk_falling_edge_events()	35
5.6.3.12	gpiod_line_request_bulk_falling_edge_events_flags()	36
5.6.3.13	gpiod_line_request_bulk_input()	36
5.6.3.14	gpiod_line_request_bulk_input_flags()	37
5.6.3.15	gpiod_line_request_bulk_output()	37
5.6.3.16	gpiod_line_request_bulk_output_flags()	37

5.6.3.17	<code>gpiod_line_request_bulk_rising_edge_events()</code>	38
5.6.3.18	<code>gpiod_line_request_bulk_rising_edge_events_flags()</code>	38
5.6.3.19	<code>gpiod_line_request_falling_edge_events()</code>	39
5.6.3.20	<code>gpiod_line_request_falling_edge_events_flags()</code>	39
5.6.3.21	<code>gpiod_line_request_input()</code>	39
5.6.3.22	<code>gpiod_line_request_input_flags()</code>	40
5.6.3.23	<code>gpiod_line_request_output()</code>	40
5.6.3.24	<code>gpiod_line_request_output_flags()</code>	41
5.6.3.25	<code>gpiod_line_request_rising_edge_events()</code>	41
5.6.3.26	<code>gpiod_line_request_rising_edge_events_flags()</code>	42
5.7	Reading & setting line values	43
5.7.1	Detailed Description	43
5.7.2	Function Documentation	43
5.7.2.1	<code>gpiod_line_get_value()</code>	43
5.7.2.2	<code>gpiod_line_get_value_bulk()</code>	44
5.7.2.3	<code>gpiod_line_set_value()</code>	44
5.7.2.4	<code>gpiod_line_set_value_bulk()</code>	44
5.8	Setting line configuration	47
5.8.1	Detailed Description	47
5.8.2	Function Documentation	47
5.8.2.1	<code>gpiod_line_set_config()</code>	47
5.8.2.2	<code>gpiod_line_set_config_bulk()</code>	48
5.8.2.3	<code>gpiod_line_set_direction_input()</code>	48
5.8.2.4	<code>gpiod_line_set_direction_input_bulk()</code>	49
5.8.2.5	<code>gpiod_line_set_direction_output()</code>	49
5.8.2.6	<code>gpiod_line_set_direction_output_bulk()</code>	50
5.8.2.7	<code>gpiod_line_set_flags()</code>	50
5.8.2.8	<code>gpiod_line_set_flags_bulk()</code>	50
5.9	Line events handling	52
5.9.1	Detailed Description	52

5.9.2	Enumeration Type Documentation	53
5.9.2.1	anonymous enum	53
5.9.3	Function Documentation	53
5.9.3.1	gpiod_line_event_get_fd()	53
5.9.3.2	gpiod_line_event_read()	53
5.9.3.3	gpiod_line_event_read_fd()	54
5.9.3.4	gpiod_line_event_read_fd_multiple()	54
5.9.3.5	gpiod_line_event_read_multiple()	55
5.9.3.6	gpiod_line_event_wait()	55
5.9.3.7	gpiod_line_event_wait_bulk()	56
5.10	Stuff that didn't fit anywhere else	57
5.10.1	Detailed Description	57
5.10.2	Function Documentation	57
5.10.2.1	gpiod_version_string()	57
5.11	C++ bindings	58
5.11.1	Detailed Description	58
5.11.2	Function Documentation	58
5.11.2.1	begin()	58
5.11.2.2	end()	59
5.11.2.3	is_gpiochip_device()	59

6	Data Structure Documentation	61
6.1	gpiod::chip Class Reference	61
6.1.1	Detailed Description	62
6.1.2	Constructor & Destructor Documentation	62
6.1.2.1	chip() [1/4]	62
6.1.2.2	chip() [2/4]	62
6.1.2.3	chip() [3/4]	63
6.1.2.4	chip() [4/4]	63
6.1.2.5	~chip()	63
6.1.3	Member Function Documentation	63
6.1.3.1	find_line()	64
6.1.3.2	get_all_lines()	64
6.1.3.3	get_line()	64
6.1.3.4	get_lines()	65
6.1.3.5	label()	65
6.1.3.6	name()	65
6.1.3.7	num_lines()	65
6.1.3.8	open()	66
6.1.3.9	operator bool()	67
6.1.3.10	operator"!()	67
6.1.3.11	operator"!=()	67
6.1.3.12	operator=() [1/2]	68
6.1.3.13	operator=() [2/2]	68
6.1.3.14	operator==()	68
6.2	gpiod_line_event Struct Reference	69
6.2.1	Detailed Description	69
6.2.2	Field Documentation	69
6.2.2.1	event_type	69
6.2.2.2	offset	70
6.2.2.3	ts	70

6.3	gpiod_line_request_config Struct Reference	70
6.3.1	Detailed Description	70
6.3.2	Field Documentation	70
6.3.2.1	consumer	71
6.3.2.2	flags	71
6.3.2.3	request_type	71
6.4	gpiod::line_bulk::iterator Class Reference	71
6.4.1	Detailed Description	72
6.4.2	Constructor & Destructor Documentation	72
6.4.2.1	iterator() [1/3]	72
6.4.2.2	iterator() [2/3]	72
6.4.2.3	iterator() [3/3]	73
6.4.3	Member Function Documentation	73
6.4.3.1	operator!=(())	73
6.4.3.2	operator*()	73
6.4.3.3	operator++()	74
6.4.3.4	operator->()	74
6.4.3.5	operator=() [1/2]	74
6.4.3.6	operator=() [2/2]	74
6.4.3.7	operator==(())	75
6.5	gpiod::line Class Reference	75
6.5.1	Detailed Description	77
6.5.2	Member Enumeration Documentation	77
6.5.2.1	anonymous enum	77
6.5.2.2	anonymous enum	78
6.5.2.3	anonymous enum	78
6.5.3	Constructor & Destructor Documentation	78
6.5.3.1	line() [1/3]	78
6.5.3.2	line() [2/3]	78
6.5.3.3	line() [3/3]	79

6.5.4	Member Function Documentation	79
6.5.4.1	bias()	79
6.5.4.2	consumer()	79
6.5.4.3	direction()	80
6.5.4.4	drive()	80
6.5.4.5	event_get_fd()	80
6.5.4.6	event_read()	80
6.5.4.7	event_read_multiple()	81
6.5.4.8	event_wait()	81
6.5.4.9	get_chip()	81
6.5.4.10	get_value()	81
6.5.4.11	is_active_low()	82
6.5.4.12	is_requested()	82
6.5.4.13	is_used()	82
6.5.4.14	name()	82
6.5.4.15	offset()	83
6.5.4.16	operator bool()	83
6.5.4.17	operator"!()	83
6.5.4.18	operator"!=(83
6.5.4.19	operator)=([1/2]	84
6.5.4.20	operator)=([2/2]	84
6.5.4.21	operator==(84
6.5.4.22	request()	85
6.5.4.23	reset()	85
6.5.4.24	set_config()	85
6.5.4.25	set_direction_output()	86
6.5.4.26	set_flags()	86
6.5.4.27	set_value()	86
6.6	gpiod::line_bulk Class Reference	86
6.6.1	Detailed Description	88

6.6.2	Constructor & Destructor Documentation	88
6.6.2.1	line_bulk() [1/4]	88
6.6.2.2	line_bulk() [2/4]	88
6.6.2.3	line_bulk() [3/4]	89
6.6.2.4	line_bulk() [4/4]	89
6.6.3	Member Function Documentation	89
6.6.3.1	append()	89
6.6.3.2	begin()	90
6.6.3.3	empty()	90
6.6.3.4	end()	90
6.6.3.5	event_wait()	90
6.6.3.6	get()	91
6.6.3.7	get_values()	91
6.6.3.8	operator bool()	92
6.6.3.9	operator"!()	92
6.6.3.10	operator=() [1/2]	92
6.6.3.11	operator=() [2/2]	92
6.6.3.12	operator[]()	93
6.6.3.13	request()	93
6.6.3.14	set_config()	94
6.6.3.15	set_direction_output()	94
6.6.3.16	set_flags()	94
6.6.3.17	set_values()	94
6.6.3.18	size()	96
6.7	gpiod::line_event Struct Reference	96
6.7.1	Detailed Description	97
6.7.2	Member Enumeration Documentation	97
6.7.2.1	anonymous enum	97
6.7.3	Field Documentation	97
6.7.3.1	event_type	97

6.7.3.2	source	98
6.7.3.3	timestamp	98
6.8	gpiod::line_iter Class Reference	98
6.8.1	Detailed Description	99
6.8.2	Constructor & Destructor Documentation	99
6.8.2.1	line_iter() [1/4]	99
6.8.2.2	line_iter() [2/4]	99
6.8.2.3	line_iter() [3/4]	99
6.8.2.4	line_iter() [4/4]	100
6.8.3	Member Function Documentation	100
6.8.3.1	operator!=(())	100
6.8.3.2	operator*()	100
6.8.3.3	operator++()	101
6.8.3.4	operator->()	101
6.8.3.5	operator=() [1/2]	101
6.8.3.6	operator=() [2/2]	101
6.8.3.7	operator==(())	102
6.9	gpiod::line_request Struct Reference	102
6.9.1	Detailed Description	103
6.9.2	Member Enumeration Documentation	103
6.9.2.1	anonymous enum	103
6.9.3	Field Documentation	103
6.9.3.1	consumer	104
6.9.3.2	FLAG_ACTIVE_LOW	104
6.9.3.3	FLAG_BIAS_DISABLED	104
6.9.3.4	FLAG_BIAS_PULL_DOWN	104
6.9.3.5	FLAG_BIAS_PULL_UP	104
6.9.3.6	FLAG_OPEN_DRAIN	105
6.9.3.7	FLAG_OPEN_SOURCE	105
6.9.3.8	flags	105
6.9.3.9	request_type	105
7	File Documentation	107
7.1	gpiod.h File Reference	107
7.2	gpiod.hpp File Reference	112
	Index	115

Chapter 1

libgpiod public API

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

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

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

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

- Common helper macros 9
- GPIO chip operations 11
- GPIO line operations 16
 - Operating on multiple lines 17
 - Line info 22
 - Line requests 29
 - Reading & setting line values 43
 - Setting line configuration 47
 - Line events handling 52
- Stuff that didn't fit anywhere else 57
- C++ bindings 58

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

gpiod::chip	Represents a GPIO chip	61
gpiod_line_event	Structure holding event info	69
gpiod_line_request_config	Structure holding configuration of a line request	70
gpiod::line_bulk::iterator	Iterator for iterating over lines held by line_bulk	71
gpiod::line	Represents a single GPIO line	75
gpiod::line_bulk	Represents a set of GPIO lines	86
gpiod::line_event	Describes a single GPIO line event	96
gpiod::line_iter	Allows to iterate over all lines owned by a GPIO chip	98
gpiod::line_request	Stores the configuration for line requests	102

Chapter 4

File Index

4.1 File List

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

gpiod.h	107
gpiod.hpp	112

Chapter 5

Module Documentation

5.1 Common helper macros

Commonly used utility macros.

Macros

- `#define GPIOD_API __attribute__((visibility("default")))`
Makes symbol visible.
- `#define GPIOD_BIT(nr) (1UL << (nr))`
Shift 1 by given offset.

5.1.1 Detailed Description

Commonly used utility macros.

5.1.2 Macro Definition Documentation

5.1.2.1 GPIOD_BIT

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

Shift 1 by given offset.

Parameters

<i>nr</i>	Bit position.
-----------	---------------

Returns

1 shifted by nr.

Definition at line 57 of file gpiod.h.

5.2 GPIO chip operations

Functions and data structures dealing with GPIO chips.

Functions

- bool [gpiod_is_gpiochip_device](#) (const char *path) [GPIO_API](#)
Check if the file pointed to by path is a GPIO chip character device.
- struct gpiod_chip * [gpiod_chip_open](#) (const char *path) [GPIO_API](#)
Open a gpiochip by path.
- void [gpiod_chip_close](#) (struct gpiod_chip *chip) [GPIO_API](#)
Close a GPIO chip handle and release all allocated resources.
- const char * [gpiod_chip_name](#) (struct gpiod_chip *chip) [GPIO_API](#)
Get the GPIO chip name as represented in the kernel.
- const char * [gpiod_chip_label](#) (struct gpiod_chip *chip) [GPIO_API](#)
Get the GPIO chip label as represented in the kernel.
- unsigned int [gpiod_chip_num_lines](#) (struct gpiod_chip *chip) [GPIO_API](#)
Get the number of GPIO lines exposed by this chip.
- struct gpiod_line * [gpiod_chip_get_line](#) (struct gpiod_chip *chip, unsigned int offset) [GPIO_API](#)
Get the handle to the GPIO line at given offset.
- struct gpiod_line_bulk * [gpiod_chip_get_lines](#) (struct gpiod_chip *chip, unsigned int *offsets, unsigned int num_offsets) [GPIO_API](#)
Retrieve a set of lines and store them in a line bulk object.
- struct gpiod_line_bulk * [gpiod_chip_get_all_lines](#) (struct gpiod_chip *chip) [GPIO_API](#)
Retrieve all lines exposed by a chip and store them in a bulk object.
- struct gpiod_line_bulk * [gpiod_chip_find_line](#) (struct gpiod_chip *chip, const char *name) [GPIO_API](#)
Find all GPIO lines by name among lines exposed by this GPIO chip.
- struct gpiod_line * [gpiod_chip_find_line_unique](#) (struct gpiod_chip *chip, const char *name) [GPIO_API](#)
Find a unique line by name among lines exposed by this GPIO chip.

5.2.1 Detailed Description

Functions and data structures dealing with GPIO chips.

5.2.2 Function Documentation

5.2.2.1 gpiod_chip_close()

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

Close a GPIO chip handle and release all allocated resources.

Parameters

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

5.2.2.2 `gpiod_chip_find_line()`

```
struct gpiod_line_bulk* gpiod_chip_find_line (
    struct gpiod_chip * chip,
    const char * name )
```

Find all GPIO lines by name among lines exposed by this GPIO chip.

Parameters

<i>chip</i>	The GPIO chip object.
<i>name</i>	GPIO line name to look for.

Returns

New line bulk object containing all matching lines or NULL on error.

If no line with given name is associated with this chip, the function sets `errno` to `ENOENT`.

5.2.2.3 `gpiod_chip_find_line_unique()`

```
struct gpiod_line* gpiod_chip_find_line_unique (
    struct gpiod_chip * chip,
    const char * name )
```

Find a unique line by name among lines exposed by this GPIO chip.

Parameters

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

Returns

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

If no line with given name is associated with this chip, the function sets `errno` to `ENOENT`. If more than one line with given name is associated with this chip, the function sets `errno` to `ERANGE`.

5.2.2.4 `gpiod_chip_get_all_lines()`

```
struct gpiod_line_bulk* gpiod_chip_get_all_lines (
    struct gpiod_chip * chip )
```


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

Parameters

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

Returns

New line bulk object or NULL on error.

5.2.2.5 `gpiod_chip_get_line()`

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

Get the handle to the GPIO line at given offset.

Parameters

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

Returns

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

5.2.2.6 `gpiod_chip_get_lines()`

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

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

Parameters

<i>chip</i>	The GPIO chip object.
<i>offsets</i>	Array of offsets of lines to retrieve.
<i>num_offsets</i>	Number of lines to retrieve.

Returns

New line bulk object or NULL on error.

5.2.2.7 gpiod_chip_label()

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

Get the GPIO chip label as represented in the kernel.

Parameters

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

Returns

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

5.2.2.8 gpiod_chip_name()

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

Get the GPIO chip name as represented in the kernel.

Parameters

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

Returns

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

5.2.2.9 gpiod_chip_num_lines()

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

Get the number of GPIO lines exposed by this chip.

Parameters

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

Returns

Number of GPIO lines.

5.2.2.10 gpiod_chip_open()

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

Open a gpiochip by path.

Parameters

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

Returns

GPIO chip handle or NULL if an error occurred.

5.2.2.11 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.
-------------	----------------

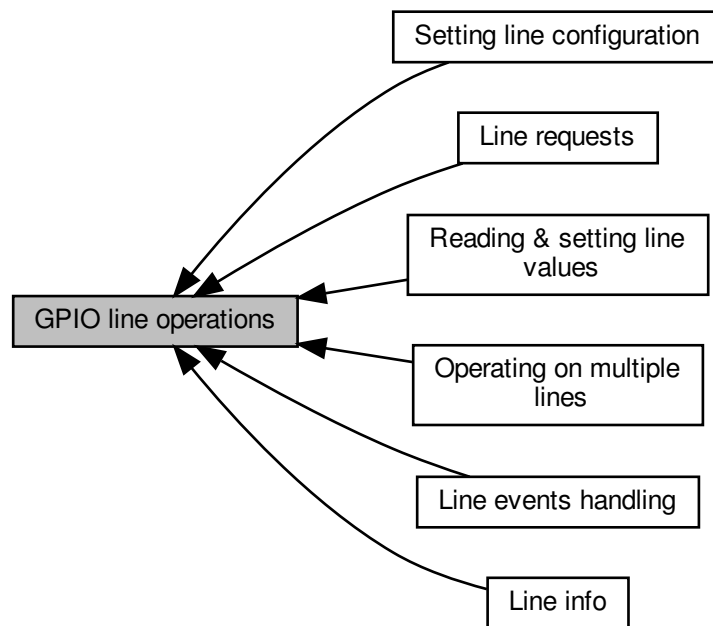
Returns

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

5.3 GPIO line operations

Functions and data structures dealing with GPIO lines.

Collaboration diagram for GPIO line operations:



Modules

- [Operating on multiple lines](#)
Convenience data structures and helper functions for storing and operating on multiple lines at once.
- [Line info](#)
Definitions and functions for retrieving kernel information about both requested and free lines.
- [Line requests](#)
Interface for requesting GPIO lines from userspace for both values and events.
- [Reading & setting line values](#)
Functions allowing to read and set GPIO line values for single lines and in bulk.
- [Setting line configuration](#)
Functions allowing modification of config options of GPIO lines requested from user-space.
- [Line events handling](#)
Structures and functions allowing to poll lines for events and read them, both for individual lines as well as in bulk.

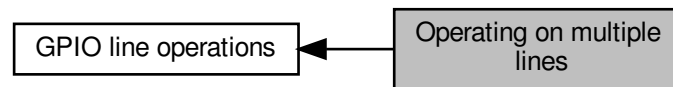
5.3.1 Detailed Description

Functions and data structures dealing with GPIO lines.

5.4 Operating on multiple lines

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

Collaboration diagram for Operating on multiple lines:



Typedefs

- typedef int(* [gpiod_line_bulk_foreach_cb](#)) (struct [gpiod_line](#) *, void *)
Signature of the callback passed to [gpiod_line_bulk_foreach_line](#).

Enumerations

- enum { [GPIOD_LINE_BULK_CB_NEXT](#) = 0, [GPIOD_LINE_BULK_CB_STOP](#) }
Values returned by the callback passed to [gpiod_line_bulk_foreach_line](#).

Functions

- struct [gpiod_line_bulk](#) * [gpiod_line_bulk_new](#) (unsigned int max_lines) [GPIOD_API](#)
Allocate and initialize a new line bulk object.
- void [gpiod_line_bulk_reset](#) (struct [gpiod_line_bulk](#) *bulk) [GPIOD_API](#)
Reset a bulk object.
- void [gpiod_line_bulk_free](#) (struct [gpiod_line_bulk](#) *bulk) [GPIOD_API](#)
Release all resources allocated for this bulk object.
- int [gpiod_line_bulk_add_line](#) (struct [gpiod_line_bulk](#) *bulk, struct [gpiod_line](#) *line) [GPIOD_API](#)
Add a single line to a GPIO bulk object.
- struct [gpiod_line](#) * [gpiod_line_bulk_get_line](#) (struct [gpiod_line_bulk](#) *bulk, unsigned int index) [GPIOD_API](#)
Retrieve the line handle from a line bulk object at given index.
- unsigned int [gpiod_line_bulk_num_lines](#) (struct [gpiod_line_bulk](#) *bulk) [GPIOD_API](#)
Retrieve the number of GPIO lines held by this line bulk object.
- void [gpiod_line_bulk_foreach_line](#) (struct [gpiod_line_bulk](#) *bulk, [gpiod_line_bulk_foreach_cb](#) func, void *data) [GPIOD_API](#)
Iterate over all lines held by this bulk object.

5.4.1 Detailed Description

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

5.4.2 Typedef Documentation

5.4.2.1 `gpiod_line_bulk_foreach_cb`

```
typedef int(* gpiod_line_bulk_foreach_cb) (struct gpiod_line *, void *)
```

Signature of the callback passed to [gpiod_line_bulk_foreach_line](#).

Takes the current line and additional user data as arguments.

Definition at line 247 of file `gpiod.h`.

5.4.3 Enumeration Type Documentation

5.4.3.1 anonymous enum

```
anonymous enum
```

Values returned by the callback passed to [gpiod_line_bulk_foreach_line](#).

Enumerator

<code>GPIO_LINE_BULK_CB_NEXT</code>	Continue the loop. Stop the loop.
-------------------------------------	-----------------------------------

Definition at line 235 of file `gpiod.h`.

5.4.4 Function Documentation

5.4.4.1 `gpiod_line_bulk_add_line()`

```
int gpiod_line_bulk_add_line (
    struct gpiod_line_bulk * bulk,
    struct gpiod_line * line )
```

Add a single line to a GPIO bulk object.

Parameters

<i>bulk</i>	Line bulk object.
<i>line</i>	Line to add.

Returns

0 on success, -1 on error.

Note

The line is added at the next free bulk index.

The function can fail if this bulk already holds its maximum amount of lines or if the added line is associated with a different chip than all the other lines already held by this object.

5.4.4.2 gpiod_line_bulk_foreach_line()

```
void gpiod_line_bulk_foreach_line (
    struct gpiod_line_bulk * bulk,
    gpiod_line_bulk_foreach_cb func,
    void * data )
```

Iterate over all lines held by this bulk object.

Parameters

<i>bulk</i>	Bulk object to iterate over.
<i>func</i>	Callback to be called for each line.
<i>data</i>	User data pointer that is passed to the callback.

5.4.4.3 gpiod_line_bulk_free()

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

Release all resources allocated for this bulk object.

Parameters

<i>bulk</i>	Bulk object to free.
-------------	----------------------

5.4.4.4 gpiod_line_bulk_get_line()

```
struct gpiod_line* gpiod_line_bulk_get_line (
    struct gpiod_line_bulk * bulk,
    unsigned int index )
```

Retrieve the line handle from a line bulk object at given index.

Parameters

<i>bulk</i>	Line bulk object.
<i>index</i>	Index of the line to retrieve.

Returns

Line handle at given index or NULL if index is greater or equal to the number of lines this bulk can hold.

5.4.4.5 gpiod_line_bulk_new()

```
struct gpiod_line_bulk* gpiod_line_bulk_new (  
    unsigned int max_lines )
```

Allocate and initialize a new line bulk object.

Parameters

<i>max_lines</i>	Maximum number of lines this object can hold.
------------------	---

Returns

New line bulk object or NULL on error.

5.4.4.6 gpiod_line_bulk_num_lines()

```
unsigned int gpiod_line_bulk_num_lines (  
    struct gpiod_line_bulk * bulk )
```

Retrieve the number of GPIO lines held by this line bulk object.

Parameters

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

Returns

Number of lines held by this line bulk.

5.4.4.7 `gpiod_line_bulk_reset()`

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

Reset a bulk object.

Remove all lines and set size to 0.

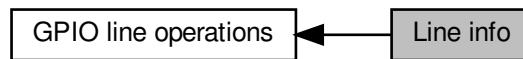
Parameters

<i>bulk</i>	Bulk object to reset.
-------------	-----------------------

5.5 Line info

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

Collaboration diagram for Line info:



Enumerations

- enum { [GPIOD_LINE_DIRECTION_INPUT](#) = 1, [GPIOD_LINE_DIRECTION_OUTPUT](#) }
Possible direction settings.
- enum { [GPIOD_LINE_DRIVE_PUSH_PULL](#) = 1, [GPIOD_LINE_DRIVE_OPEN_DRAIN](#), [GPIOD_LINE_DRIVE_OPEN_SOURCE](#) }
Possible drive settings.
- enum { [GPIOD_LINE_BIAS_UNKNOWN](#) = 1, [GPIOD_LINE_BIAS_DISABLED](#), [GPIOD_LINE_BIAS_PULL_UP](#), [GPIOD_LINE_BIAS_PULL_DOWN](#) }
Possible internal bias settings.

Functions

- unsigned int [gpiod_line_offset](#) (struct [gpiod_line](#) *line) [GPIOD_API](#)
Read the GPIO line offset.
- const char * [gpiod_line_name](#) (struct [gpiod_line](#) *line) [GPIOD_API](#)
Read the GPIO line name.
- const char * [gpiod_line_consumer](#) (struct [gpiod_line](#) *line) [GPIOD_API](#)
Read the GPIO line consumer name.
- int [gpiod_line_direction](#) (struct [gpiod_line](#) *line) [GPIOD_API](#)
Read the GPIO line direction setting.
- bool [gpiod_line_is_active_low](#) (struct [gpiod_line](#) *line) [GPIOD_API](#)
Check if the signal of this line is inverted.
- int [gpiod_line_bias](#) (struct [gpiod_line](#) *line) [GPIOD_API](#)
Read the GPIO line bias setting.
- bool [gpiod_line_is_used](#) (struct [gpiod_line](#) *line) [GPIOD_API](#)
Check if the line is currently in use.
- int [gpiod_line_drive](#) (struct [gpiod_line](#) *line) [GPIOD_API](#)
Read the GPIO line drive setting.
- int [gpiod_line_update](#) (struct [gpiod_line](#) *line) [GPIOD_API](#)
Re-read the line info.
- struct [gpiod_chip](#) * [gpiod_line_get_chip](#) (struct [gpiod_line](#) *line) [GPIOD_API](#)
Get the handle to the GPIO chip controlling this line.

5.5.1 Detailed Description

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

5.5.2 Enumeration Type Documentation

5.5.2.1 anonymous enum

`anonymous enum`

Possible direction settings.

Enumerator

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

Definition at line 272 of file `gpiod.h`.

5.5.2.2 anonymous enum

`anonymous enum`

Possible drive settings.

Enumerator

<code>GPIOD_LINE_DRIVE_PUSH_PULL</code>	Drive setting is push-pull.
<code>GPIOD_LINE_DRIVE_OPEN_DRAIN</code>	Line output is open-drain.
<code>GPIOD_LINE_DRIVE_OPEN_SOURCE</code>	Line output is open-source.

Definition at line 282 of file `gpiod.h`.

5.5.2.3 anonymous enum

`anonymous enum`

Possible internal bias settings.

Enumerator

GPIOD_LINE_BIAS_UNKNOWN	The internal bias state is unknown.
GPIOD_LINE_BIAS_DISABLED	The internal bias is disabled.
GPIOD_LINE_BIAS_PULL_UP	The internal pull-up bias is enabled.
GPIOD_LINE_BIAS_PULL_DOWN	The internal pull-down bias is enabled.

Definition at line 294 of file gpio.h.

5.5.3 Function Documentation

5.5.3.1 `gpio_line_bias()`

```
int gpio_line_bias (
    struct gpio_line * line )
```

Read the GPIO line bias setting.

Parameters

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

Returns

Returns GPIOD_LINE_BIAS_PULL_UP, GPIOD_LINE_BIAS_PULL_DOWN, GPIOD_LINE_BIAS_DISABLE or GPIOD_LINE_BIAS_UNKNOWN.

5.5.3.2 `gpio_line_consumer()`

```
const char* gpio_line_consumer (
    struct gpio_line * line )
```

Read the GPIO line consumer name.

Parameters

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

Returns

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

5.5.3.3 `gpiod_line_direction()`

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

Read the GPIO line direction setting.

Parameters

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

Returns

Returns `GPIO_LINE_DIRECTION_INPUT` or `GPIO_LINE_DIRECTION_OUTPUT`.

5.5.3.4 `gpiod_line_drive()`

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

Read the GPIO line drive setting.

Parameters

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

Returns

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

5.5.3.5 `gpiod_line_get_chip()`

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

Get the handle to the GPIO chip controlling this line.

Parameters

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

Returns

Pointer to the GPIO chip handle controlling this line.

5.5.3.6 gpiod_line_is_active_low()

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

Check if the signal of this line is inverted.

Parameters

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

Returns

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

5.5.3.7 gpiod_line_is_used()

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

Check if the line is currently in use.

Parameters

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

Returns

True if the line is in use, false otherwise.

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

5.5.3.8 gpiod_line_name()

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

Read the GPIO line name.

Parameters

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

Returns

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

5.5.3.9 gpiod_line_offset()

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

Read the GPIO line offset.

Parameters

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

Returns

Line offset.

5.5.3.10 gpiod_line_update()

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

Re-read the line info.

Parameters

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

Returns

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

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

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

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

5.6 Line requests

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

Collaboration diagram for Line requests:



Data Structures

- struct [gpiod_line_request_config](#)
Structure holding configuration of a line request.

Enumerations

- enum {
[GPIOD_LINE_REQUEST_DIRECTION_AS_IS](#) = 1, [GPIOD_LINE_REQUEST_DIRECTION_INPUT](#), [GPIOD_LINE_REQUEST_DIRECTION_OUTPUT](#), [GPIOD_LINE_REQUEST_EVENT_FALLING_EDGE](#), [GPIOD_LINE_REQUEST_EVENT_RISING_EDGE](#), [GPIOD_LINE_REQUEST_EVENT_BOTH_EDGES](#) }
 Available types of requests.
- enum {
[GPIOD_LINE_REQUEST_FLAG_OPEN_DRAIN](#) = GPIOD_BIT(0), [GPIOD_LINE_REQUEST_FLAG_OPEN_SOURCE](#) = GPIOD_BIT(1), [GPIOD_LINE_REQUEST_FLAG_ACTIVE_LOW](#) = GPIOD_BIT(2), [GPIOD_LINE_REQUEST_FLAG_BIAS_DISABLED](#) = GPIOD_BIT(3), [GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_DOWN](#) = GPIOD_BIT(4), [GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_UP](#) = GPIOD_BIT(5) }
 Miscellaneous GPIO request flags.

Functions

- int [gpiod_line_request](#) (struct [gpiod_line](#) *line, const struct [gpiod_line_request_config](#) *config, int default_val) [GPIO_API](#)
Reserve a single line.
- int [gpiod_line_request_input](#) (struct [gpiod_line](#) *line, const char *consumer) [GPIO_API](#)
Reserve a single line, set the direction to input.
- int [gpiod_line_request_output](#) (struct [gpiod_line](#) *line, const char *consumer, int default_val) [GPIO_API](#)
Reserve a single line, set the direction to output.
- int [gpiod_line_request_rising_edge_events](#) (struct [gpiod_line](#) *line, const char *consumer) [GPIO_API](#)
Request rising edge event notifications on a single line.
- int [gpiod_line_request_falling_edge_events](#) (struct [gpiod_line](#) *line, const char *consumer) [GPIO_API](#)
Request falling edge event notifications on a single line.
- int [gpiod_line_request_both_edges_events](#) (struct [gpiod_line](#) *line, const char *consumer) [GPIO_API](#)

- Request all event type notifications on a single line.*

 - int [gpiod_line_request_input_flags](#) (struct `gpiod_line` *line, const char *consumer, int flags) [GPIO_API](#)

Reserve a single line, set the direction to input.
- int [gpiod_line_request_output_flags](#) (struct `gpiod_line` *line, const char *consumer, int flags, int default_val) [GPIO_API](#)

Reserve a single line, set the direction to output.
- int [gpiod_line_request_rising_edge_events_flags](#) (struct `gpiod_line` *line, const char *consumer, int flags) [GPIO_API](#)

Request rising edge event notifications on a single line.
- int [gpiod_line_request_falling_edge_events_flags](#) (struct `gpiod_line` *line, const char *consumer, int flags) [GPIO_API](#)

Request falling edge event notifications on a single line.
- int [gpiod_line_request_both_edges_events_flags](#) (struct `gpiod_line` *line, const char *consumer, int flags) [GPIO_API](#)

Request all event type notifications on a single line.
- int [gpiod_line_request_bulk](#) (struct `gpiod_line_bulk` *bulk, const struct [gpiod_line_request_config](#) *config, const int *default_vals) [GPIO_API](#)

Reserve a set of GPIO lines.
- int [gpiod_line_request_bulk_input](#) (struct `gpiod_line_bulk` *bulk, const char *consumer) [GPIO_API](#)

Reserve a set of GPIO lines, set the direction to input.
- int [gpiod_line_request_bulk_output](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, const int *default_vals) [GPIO_API](#)

Reserve a set of GPIO lines, set the direction to output.
- int [gpiod_line_request_bulk_rising_edge_events](#) (struct `gpiod_line_bulk` *bulk, const char *consumer) [GPIO_API](#)

Request rising edge event notifications on a set of lines.
- int [gpiod_line_request_bulk_falling_edge_events](#) (struct `gpiod_line_bulk` *bulk, const char *consumer) [GPIO_API](#)

Request falling edge event notifications on a set of lines.
- int [gpiod_line_request_bulk_both_edges_events](#) (struct `gpiod_line_bulk` *bulk, const char *consumer) [GPIO_API](#)

Request all event type notifications on a set of lines.
- int [gpiod_line_request_bulk_input_flags](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, int flags) [GPIO_API](#)

Reserve a set of GPIO lines, set the direction to input.
- int [gpiod_line_request_bulk_output_flags](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, int flags, const int *default_vals) [GPIO_API](#)

Reserve a set of GPIO lines, set the direction to output.
- int [gpiod_line_request_bulk_rising_edge_events_flags](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, int flags) [GPIO_API](#)

Request rising edge event notifications on a set of lines.
- int [gpiod_line_request_bulk_falling_edge_events_flags](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, int flags) [GPIO_API](#)

Request falling edge event notifications on a set of lines.
- int [gpiod_line_request_bulk_both_edges_events_flags](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, int flags) [GPIO_API](#)

Request all event type notifications on a set of lines.
- void [gpiod_line_release](#) (struct `gpiod_line` *line) [GPIO_API](#)

Release a previously reserved line.
- void [gpiod_line_release_bulk](#) (struct `gpiod_line_bulk` *bulk) [GPIO_API](#)

Release a set of previously reserved lines.
- bool [gpiod_line_is_requested](#) (struct `gpiod_line` *line) [GPIO_API](#)

Check if the calling user has ownership of this line.
- bool [gpiod_line_is_free](#) (struct `gpiod_line` *line) [GPIO_API](#)

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

5.6.1 Detailed Description

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

5.6.2 Enumeration Type Documentation

5.6.2.1 anonymous enum

anonymous enum

Available types of requests.

Enumerator

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

Definition at line 413 of file gpiod.h.

5.6.2.2 anonymous enum

anonymous enum

Miscellaneous GPIO request flags.

Enumerator

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

Definition at line 431 of file gpiod.h.

5.6.3 Function Documentation

5.6.3.1 `gpiod_line_is_free()`

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

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

Parameters

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

Returns

True if given line is free, false otherwise.

5.6.3.2 `gpiod_line_is_requested()`

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

Check if the calling user has ownership of this line.

Parameters

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

Returns

True if given line was requested, false otherwise.

5.6.3.3 `gpiod_line_release()`

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

Release a previously reserved line.

Parameters

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

5.6.3.4 `gpiod_line_release_bulk()`

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

Release a set of previously reserved lines.

Parameters

<i>bulk</i>	Set of GPIO lines to release.
-------------	-------------------------------

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

5.6.3.5 `gpiod_line_request()`

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

Reserve a single line.

Parameters

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

Returns

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

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

5.6.3.6 `gpiod_line_request_both_edges_events()`

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

Request all event type notifications on a single line.

Parameters

<i>line</i>	GPIO line object.
<i>consumer</i>	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.7 gpiod_line_request_both_edges_events_flags()

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

Request all event type notifications on a single line.

Parameters

<i>line</i>	GPIO line object.
<i>consumer</i>	Name of the consumer.
<i>flags</i>	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.8 gpiod_line_request_bulk()

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

Reserve a set of GPIO lines.

Parameters

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

Returns

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

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

5.6.3.9 `gpiod_line_request_bulk_both_edges_events()`

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

Request all event type notifications on a set of lines.

Parameters

<i>bulk</i>	Set of GPIO lines to request.
<i>consumer</i>	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.10 `gpiod_line_request_bulk_both_edges_events_flags()`

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

Request all event type notifications on a set of lines.

Parameters

<i>bulk</i>	Set of GPIO lines to request.
<i>consumer</i>	Name of the consumer.
<i>flags</i>	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.11 `gpiod_line_request_bulk_falling_edge_events()`

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

Request falling edge event notifications on a set of lines.

Parameters

<i>bulk</i>	Set of GPIO lines to request.
<i>consumer</i>	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.12 gpiod_line_request_bulk_falling_edge_events_flags()

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

Request falling edge event notifications on a set of lines.

Parameters

<i>bulk</i>	Set of GPIO lines to request.
<i>consumer</i>	Name of the consumer.
<i>flags</i>	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.13 gpiod_line_request_bulk_input()

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

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

Parameters

<i>bulk</i>	Set of GPIO lines to reserve.
<i>consumer</i>	Name of the consumer.

Returns

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

5.6.3.14 `gpiod_line_request_bulk_input_flags()`

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

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

Parameters

<i>bulk</i>	Set of GPIO lines to reserve.
<i>consumer</i>	Name of the consumer.
<i>flags</i>	Additional request flags.

Returns

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

5.6.3.15 `gpiod_line_request_bulk_output()`

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

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

Parameters

<i>bulk</i>	Set of GPIO lines to reserve.
<i>consumer</i>	Name of the consumer.
<i>default_vals</i>	Initial line values.

Returns

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

5.6.3.16 `gpiod_line_request_bulk_output_flags()`

```
int gpiod_line_request_bulk_output_flags (
    struct gpiod_line_bulk * bulk,
```

```

    const char * consumer,
    int flags,
    const int * default_vals )

```

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

Parameters

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

Returns

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

5.6.3.17 `gpiod_line_request_bulk_rising_edge_events()`

```

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

```

Request rising edge event notifications on a set of lines.

Parameters

<i>bulk</i>	Set of GPIO lines to request.
<i>consumer</i>	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.18 `gpiod_line_request_bulk_rising_edge_events_flags()`

```

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

```

Request rising edge event notifications on a set of lines.

Parameters

<i>bulk</i>	Set of GPIO lines to request.
<i>consumer</i>	Name of the consumer.
<i>flags</i>	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.19 gpiod_line_request_falling_edge_events()

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

Request falling edge event notifications on a single line.

Parameters

<i>line</i>	GPIO line object.
<i>consumer</i>	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.20 gpiod_line_request_falling_edge_events_flags()

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

Request falling edge event notifications on a single line.

Parameters

<i>line</i>	GPIO line object.
<i>consumer</i>	Name of the consumer.
<i>flags</i>	Additional request flags.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.21 gpiod_line_request_input()

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

Reserve a single line, set the direction to input.

Parameters

<i>line</i>	GPIO line object.
<i>consumer</i>	Name of the consumer.

Returns

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

5.6.3.22 `gpiod_line_request_input_flags()`

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

Reserve a single line, set the direction to input.

Parameters

<i>line</i>	GPIO line object.
<i>consumer</i>	Name of the consumer.
<i>flags</i>	Additional request flags.

Returns

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

5.6.3.23 `gpiod_line_request_output()`

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

Reserve a single line, set the direction to output.

Parameters

<i>line</i>	GPIO line object.
<i>consumer</i>	Name of the consumer.
<i>default_val</i>	Initial line value.

Returns

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

5.6.3.24 gpiod_line_request_output_flags()

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

Reserve a single line, set the direction to output.

Parameters

<i>line</i>	GPIO line object.
<i>consumer</i>	Name of the consumer.
<i>flags</i>	Additional request flags.
<i>default_val</i>	Initial line value.

Returns

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

5.6.3.25 gpiod_line_request_rising_edge_events()

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

Request rising edge event notifications on a single line.

Parameters

<i>line</i>	GPIO line object.
<i>consumer</i>	Name of the consumer.

Returns

0 if the operation succeeds, -1 on failure.

5.6.3.26 `gpiod_line_request_rising_edge_events_flags()`

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

Request rising edge event notifications on a single line.

Parameters

<i>line</i>	GPIO line object.
<i>consumer</i>	Name of the consumer.
<i>flags</i>	Additional request flags.

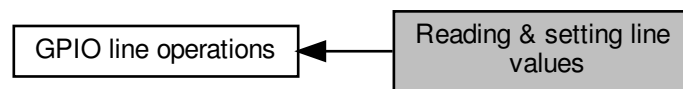
Returns

0 if the operation succeeds, -1 on failure.

5.7 Reading & setting line values

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

Collaboration diagram for Reading & setting line values:



Functions

- int [gpiod_line_get_value](#) (struct `gpiod_line` *line) [GPIOD_API](#)
Read current value of a single GPIO line.
- int [gpiod_line_get_value_bulk](#) (struct `gpiod_line_bulk` *bulk, int *values) [GPIOD_API](#)
Read current values of a set of GPIO lines.
- int [gpiod_line_set_value](#) (struct `gpiod_line` *line, int value) [GPIOD_API](#)
Set the value of a single GPIO line.
- int [gpiod_line_set_value_bulk](#) (struct `gpiod_line_bulk` *bulk, const int *values) [GPIOD_API](#)
Set the values of a set of GPIO lines.

5.7.1 Detailed Description

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

5.7.2 Function Documentation

5.7.2.1 `gpiod_line_get_value()`

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

Read current value of a single GPIO line.

Parameters

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

Returns

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

5.7.2.2 gpiod_line_get_value_bulk()

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

Read current values of a set of GPIO lines.

Parameters

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

Returns

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

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

5.7.2.3 gpiod_line_set_value()

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

Set the value of a single GPIO line.

Parameters

<i>line</i>	GPIO line object.
<i>value</i>	New value.

Returns

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

5.7.2.4 gpiod_line_set_value_bulk()

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


Set the values of a set of GPIO lines.

Parameters

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

Returns

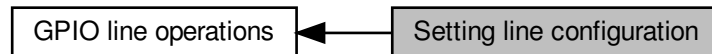
0 is the operation succeeds. In case of an error this routine returns -1 and sets the last error number.

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

5.8 Setting line configuration

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

Collaboration diagram for Setting line configuration:



Functions

- int [gpiod_line_set_config](#) (struct [gpiod_line](#) *line, int direction, int flags, int value) [GPIO_API](#)
Update the configuration of a single GPIO line.
- int [gpiod_line_set_config_bulk](#) (struct [gpiod_line_bulk](#) *bulk, int direction, int flags, const int *values) [GPIO_API](#)
Update the configuration of a set of GPIO lines.
- int [gpiod_line_set_flags](#) (struct [gpiod_line](#) *line, int flags) [GPIO_API](#)
Update the configuration flags of a single GPIO line.
- int [gpiod_line_set_flags_bulk](#) (struct [gpiod_line_bulk](#) *bulk, int flags) [GPIO_API](#)
Update the configuration flags of a set of GPIO lines.
- int [gpiod_line_set_direction_input](#) (struct [gpiod_line](#) *line) [GPIO_API](#)
Set the direction of a single GPIO line to input.
- int [gpiod_line_set_direction_input_bulk](#) (struct [gpiod_line_bulk](#) *bulk) [GPIO_API](#)
Set the direction of a set of GPIO lines to input.
- int [gpiod_line_set_direction_output](#) (struct [gpiod_line](#) *line, int value) [GPIO_API](#)
Set the direction of a single GPIO line to output.
- int [gpiod_line_set_direction_output_bulk](#) (struct [gpiod_line_bulk](#) *bulk, const int *values) [GPIO_API](#)
Set the direction of a set of GPIO lines to output.

5.8.1 Detailed Description

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

5.8.2 Function Documentation

5.8.2.1 [gpiod_line_set_config\(\)](#)

```

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

Update the configuration of a single GPIO line.

Parameters

<i>line</i>	GPIO line object.
<i>direction</i>	Updated direction which may be one of <code>GPIO_LINE_REQUEST_DIRECTION_AS_IS</code> , <code>GPIO_LINE_REQUEST_DIRECTION_INPUT</code> , or <code>GPIO_LINE_REQUEST_DIRECTION_OUTPUT</code> .
<i>flags</i>	Replacement flags.
<i>value</i>	The new output value for the line when direction is <code>GPIO_LINE_REQUEST_DIRECTION_OUTPUT</code> .

Returns

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

5.8.2.2 `gpiod_line_set_config_bulk()`

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

Update the configuration of a set of GPIO lines.

Parameters

<i>bulk</i>	Set of GPIO lines.
<i>direction</i>	Updated direction which may be one of <code>GPIO_LINE_REQUEST_DIRECTION_AS_IS</code> , <code>GPIO_LINE_REQUEST_DIRECTION_INPUT</code> , or <code>GPIO_LINE_REQUEST_DIRECTION_OUTPUT</code> .
<i>flags</i>	Replacement flags.
<i>values</i>	An array holding <code>line_bulk->num_lines</code> new logical values for lines when direction is <code>GPIO_LINE_REQUEST_DIRECTION_OUTPUT</code> . A NULL pointer is interpreted as a logical low for all lines.

Returns

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

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

5.8.2.3 `gpiod_line_set_direction_input()`

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

Set the direction of a single GPIO line to input.

Parameters

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

Returns

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

5.8.2.4 gpiod_line_set_direction_input_bulk()

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

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

Parameters

<i>bulk</i>	Set of GPIO lines.
-------------	--------------------

Returns

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

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

5.8.2.5 gpiod_line_set_direction_output()

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

Set the direction of a single GPIO line to output.

Parameters

<i>line</i>	GPIO line object.
<i>value</i>	The logical value output on the line.

Returns

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

5.8.2.6 `gpiod_line_set_direction_output_bulk()`

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

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

Parameters

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

Returns

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

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

5.8.2.7 `gpiod_line_set_flags()`

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

Update the configuration flags of a single GPIO line.

Parameters

<i>line</i>	GPIO line object.
<i>flags</i>	Replacement flags.

Returns

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

5.8.2.8 `gpiod_line_set_flags_bulk()`

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

Update the configuration flags of a set of GPIO lines.

Parameters

<i>bulk</i>	Set of GPIO lines.
<i>flags</i>	Replacement flags.

Returns

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

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

5.9 Line events handling

Structures and functions allowing to poll lines for events and read them, both for individual lines as well as in bulk.

Collaboration diagram for Line events handling:



Data Structures

- struct [gpiod_line_event](#)
Structure holding event info.

Enumerations

- enum { [GPIO_LINE_EVENT_RISING_EDGE](#) = 1, [GPIO_LINE_EVENT_FALLING_EDGE](#) }
Event types.

Functions

- int [gpiod_line_event_wait](#) (struct [gpiod_line](#) *line, const struct timespec *timeout) [GPIO_API](#)
Wait for an event on a single line.
- int [gpiod_line_event_wait_bulk](#) (struct [gpiod_line_bulk](#) *bulk, const struct timespec *timeout, struct [gpiod_line_bulk](#) *event_bulk) [GPIO_API](#)
Wait for events on a set of lines.
- int [gpiod_line_event_read](#) (struct [gpiod_line](#) *line, struct [gpiod_line_event](#) *event) [GPIO_API](#)
Read next pending event from the GPIO line.
- int [gpiod_line_event_read_multiple](#) (struct [gpiod_line](#) *line, struct [gpiod_line_event](#) *events, unsigned int num_events) [GPIO_API](#)
Read up to a certain number of events from the GPIO line.
- int [gpiod_line_event_get_fd](#) (struct [gpiod_line](#) *line) [GPIO_API](#)
Get the event file descriptor.
- int [gpiod_line_event_read_fd](#) (int fd, struct [gpiod_line_event](#) *event) [GPIO_API](#)
Read the last GPIO event directly from a file descriptor.
- int [gpiod_line_event_read_fd_multiple](#) (int fd, struct [gpiod_line_event](#) *events, unsigned int num_events) [GPIO_API](#)
Read up to a certain number of events directly from a file descriptor.

5.9.1 Detailed Description

Structures and functions allowing to poll lines for events and read them, both for individual lines as well as in bulk.

Also contains functions for retrieving the associated file descriptors and operate on them for easy integration with standard unix interfaces.

5.9.2 Enumeration Type Documentation

5.9.2.1 anonymous enum

anonymous enum

Event types.

Enumerator

<code>GPIOD_LINE_EVENT_RISING_EDGE</code>	Rising edge event.
<code>GPIOD_LINE_EVENT_FALLING_EDGE</code>	Falling edge event.

Definition at line 914 of file `gpiod.h`.

5.9.3 Function Documentation

5.9.3.1 `gpiod_line_event_get_fd()`

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

Get the event file descriptor.

Parameters

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

Returns

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

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

5.9.3.2 `gpiod_line_event_read()`

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

Read next pending event from the GPIO line.

Parameters

<i>line</i>	GPIO line object.
<i>event</i>	Buffer to which the event data will be copied.

Returns

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

Note

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

5.9.3.3 gpiod_line_event_read_fd()

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

Read the last GPIO event directly from a file descriptor.

Parameters

<i>fd</i>	File descriptor.
<i>event</i>	Buffer in which the event data will be stored.

Returns

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

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

5.9.3.4 gpiod_line_event_read_fd_multiple()

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

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

Parameters

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

Returns

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

5.9.3.5 gpiod_line_event_read_multiple()

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

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

Parameters

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

Returns

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

5.9.3.6 gpiod_line_event_wait()

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

Wait for an event on a single line.

Parameters

<i>line</i>	GPIO line object.
<i>timeout</i>	Wait time limit.

Returns

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

5.9.3.7 `gpiod_line_event_wait_bulk()`

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

Wait for events on a set of lines.

Parameters

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

Returns

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

5.10 Stuff that didn't fit anywhere else

Various libgpiod-related functions.

Functions

- `const char * gpiod_version_string (void) GPIOD_API`
- Get the API version of the library as a human-readable string.*

5.10.1 Detailed Description

Various libgpiod-related functions.

5.10.2 Function Documentation

5.10.2.1 `gpiod_version_string()`

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

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

Returns

Human-readable string containing the library version.

5.11 C++ bindings

Data Structures

- class `gpiod::chip`
Represents a GPIO chip.
- struct `gpiod::line_request`
Stores the configuration for line requests.
- class `gpiod::line`
Represents a single GPIO line.
- struct `gpiod::line_event`
Describes a single GPIO line event.
- class `gpiod::line_bulk`
Represents a set of GPIO lines.
- class `gpiod::line_iter`
Allows to iterate over all lines owned by a GPIO chip.

Functions

- `bool gpiod::is_gpiochip_device (const ::std::string &path) GPIOD_API`
Check if the file pointed to by path is a GPIO chip character device.
- `GPIOD_API line_iter gpiod::begin (line_iter iter) noexcept`
Support for range-based loops for line iterators.
- `GPIOD_API line_iter gpiod::end (const line_iter &iter) noexcept`
Support for range-based loops for line iterators.

5.11.1 Detailed Description

5.11.2 Function Documentation

5.11.2.1 `begin()`

```
GPIOD_API line_iter gpiod::begin (
    line_iter iter ) [noexcept]
```

Support for range-based loops for line iterators.

Parameters

<i>iter</i>	A line iterator.
-------------	------------------

Returns

Iterator unchanged.

5.11.2.2 end()

```
GPIOD_API line_iter gpiod::end (
    const line_iter & iter ) [noexcept]
```

Support for range-based loops for line iterators.

Parameters

<i>iter</i>	A line iterator.
-------------	------------------

Returns

New end iterator.

5.11.2.3 is_gpiochip_device()

```
bool gpiod::is_gpiochip_device (
    const ::std::string & 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 a GPIO chip character device or a symbolic link to it.

Chapter 6

Data Structure Documentation

6.1 `gpiod::chip` Class Reference

Represents a GPIO chip.

```
#include <gpiod.hpp>
```

Public Member Functions

- `GPIOD_API chip` (void)=default
Default constructor.
- `GPIOD_API chip` (const ::std::string &path)
Constructor.
- `GPIOD_API chip` (const `chip` &other)=default
Copy constructor.
- `GPIOD_API chip` (`chip` &&other)=default
Move constructor.
- `GPIOD_API chip` & `operator=` (const `chip` &other)=default
Assignment operator.
- `GPIOD_API chip` & `operator=` (`chip` &&other)=default
Move assignment operator.
- `GPIOD_API ~chip` (void)=default
Destructor.
- `GPIOD_API` void `open` (const ::std::string &path)
Open a GPIO chip.
- `GPIOD_API` void `reset` (void) noexcept
Reset the internal smart pointer owned by this object.
- `GPIOD_API` ::std::string `name` (void) const
Return the name of the chip held by this object.
- `GPIOD_API` ::std::string `label` (void) const
Return the label of the chip held by this object.
- `GPIOD_API` unsigned int `num_lines` (void) const
Return the number of lines exposed by this chip.
- `GPIOD_API` line `get_line` (unsigned int offset) const
Get the line exposed by this chip at given offset.

- `GPIO_API` `::std::vector< line > find_line` (const `::std::string &name`, bool unique=false) const
Find all GPIO lines by name among lines exposed by this GPIO chip.
- `GPIO_API` `line_bulk get_lines` (const `::std::vector< unsigned int > &offsets`) const
Get a set of lines exposed by this chip at given offsets.
- `GPIO_API` `line_bulk get_all_lines` (void) const
Get all lines exposed by this chip.
- `GPIO_API` `bool operator==` (const `chip` &rhs) const noexcept
Equality operator.
- `GPIO_API` `bool operator!=` (const `chip` &rhs) const noexcept
Inequality operator.
- `GPIO_API` `operator bool` (void) const noexcept
Check if this object holds a reference to a GPIO chip.
- `GPIO_API` `bool operator!` (void) const noexcept
Check if this object doesn't hold a reference to a GPIO chip.

6.1.1 Detailed Description

Represents a GPIO chip.

Internally this class holds a smart pointer to an open GPIO chip descriptor. Multiple objects of this class can reference the same chip. The chip is closed and all resources freed when the last reference is dropped.

Definition at line 46 of file `gpiod.hpp`.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `chip()` [1/4]

```
GPIO_API gpiod::chip::chip (
    void ) [default]
```

Default constructor.

Creates an empty GPIO chip object.

6.1.2.2 `chip()` [2/4]

```
GPIO_API gpiod::chip::chip (
    const ::std::string & path )
```

Constructor.

Opens the chip using `chip::open`.

Parameters

<i>path</i>	Path to the GPIO chip device.
-------------	-------------------------------

6.1.2.3 chip() [3/4]

```
GPIOD_API gpiod::chip::chip (  
    const chip & other ) [default]
```

Copy constructor.

References the object held by other.

Parameters

<i>other</i>	Other chip object.
--------------	--------------------

6.1.2.4 chip() [4/4]

```
GPIOD_API gpiod::chip::chip (  
    chip && other ) [default]
```

Move constructor.

References the object held by other.

Parameters

<i>other</i>	Other chip object.
--------------	--------------------

6.1.2.5 ~chip()

```
GPIOD_API gpiod::chip::~~chip (  
    void ) [default]
```

Destructor.

Unreferences the internal chip object.

6.1.3 Member Function Documentation

6.1.3.1 find_line()

```
GPIOD_API ::std::vector<line> gpiod::chip::find_line (
    const ::std::string & name,
    bool unique = false ) const
```

Find all GPIO lines by name among lines exposed by this GPIO chip.

Parameters

<i>name</i>	Line name.
<i>unique</i>	If set to true: throw an error if multiple lines match the name.

Returns

Vector of all matching lines.

6.1.3.2 get_all_lines()

```
GPIOD_API line_bulk gpiod::chip::get_all_lines (
    void ) const
```

Get all lines exposed by this chip.

Returns

All lines exposed by this chip held by a [line_bulk](#) object.

6.1.3.3 get_line()

```
GPIOD_API line gpiod::chip::get_line (
    unsigned int offset ) const
```

Get the line exposed by this chip at given offset.

Parameters

<i>offset</i>	Offset of the line.
---------------	---------------------

Returns

Line object.

6.1.3.4 get_lines()

```
GPIOD_API line_bulk gpiod::chip::get_lines (
    const ::std::vector< unsigned int > & offsets ) const
```

Get a set of lines exposed by this chip at given offsets.

Parameters

<i>offsets</i>	Vector of line offsets.
----------------	-------------------------

Returns

Set of lines held by a [line_bulk](#) object.

6.1.3.5 label()

```
GPIOD_API ::std::string gpiod::chip::label (
    void ) const
```

Return the label of the chip held by this object.

Returns

Label of the GPIO chip.

6.1.3.6 name()

```
GPIOD_API ::std::string gpiod::chip::name (
    void ) const
```

Return the name of the chip held by this object.

Returns

Name of the GPIO chip.

6.1.3.7 num_lines()

```
GPIOD_API unsigned int gpiod::chip::num_lines (
    void ) const
```

Return the number of lines exposed by this chip.

Returns

Number of lines.

6.1.3.8 open()

```
GPIOD_API void gpiod::chip::open (
    const ::std::string & path )
```

Open a GPIO chip.

Parameters

<i>path</i>	Path to the GPIO chip device.
-------------	-------------------------------

If the object already holds a reference to an open chip, it will be closed and the reference reset.

6.1.3.9 operator bool()

```
GPIOD_API gpiod::chip::operator bool (
    void ) const [explicit], [noexcept]
```

Check if this object holds a reference to a GPIO chip.

Returns

True if this object references a GPIO chip, false otherwise.

6.1.3.10 operator!()

```
GPIOD_API bool gpiod::chip::operator! (
    void ) const [noexcept]
```

Check if this object doesn't hold a reference to a GPIO chip.

Returns

False if this object references a GPIO chip, true otherwise.

6.1.3.11 operator!==(())

```
GPIOD_API bool gpiod::chip::operator!=(
    const chip & rhs ) const [noexcept]
```

Inequality operator.

Parameters

<i>rhs</i>	Right-hand side of the equation.
------------	----------------------------------

Returns

False if rhs references the same chip. True otherwise.

6.1.3.12 `operator=()` [1/2]

```
GPIOD_API chip& gpiod::chip::operator= (
    const chip & other ) [default]
```

Assignment operator.

References the object held by other.

Parameters

<i>other</i>	Other chip object.
--------------	--------------------

Returns

Reference to this object.

6.1.3.13 `operator=()` [2/2]

```
GPIOD_API chip& gpiod::chip::operator= (
    chip && other ) [default]
```

Move assignment operator.

References the object held by other.

Parameters

<i>other</i>	Other chip object.
--------------	--------------------

Returns

Reference to this object.

6.1.3.14 `operator==()`

```
GPIOD_API bool gpiod::chip::operator== (
    const chip & rhs ) const [noexcept]
```

Equality operator.

Parameters

<i>rhs</i>	Right-hand side of the equation.
------------	----------------------------------

Returns

True if rhs references the same chip. False otherwise.

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

- [gpiod.hpp](#)

6.2 gpiod_line_event Struct Reference

Structure holding event info.

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

Data Fields

- struct timespec [ts](#)
Best estimate of time of event occurrence.
- int [event_type](#)
Type of the event that occurred.
- int [offset](#)
Offset of line on which the event occurred.

6.2.1 Detailed Description

Structure holding event info.

Definition at line 924 of file gpiod.h.

6.2.2 Field Documentation

6.2.2.1 event_type

```
int gpiod_line_event::event_type
```

Type of the event that occurred.

Definition at line 927 of file gpiod.h.

6.2.2.2 offset

```
int gpiod_line_event::offset
```

Offset of line on which the event occurred.

Definition at line 929 of file gpiod.h.

6.2.2.3 ts

```
struct timespec gpiod_line_event::ts
```

Best estimate of time of event occurrence.

Definition at line 925 of file gpiod.h.

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

- [gpiod.h](#)

6.3 gpiod_line_request_config Struct Reference

Structure holding configuration of a line request.

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

Data Fields

- const char * [consumer](#)
Name of the consumer.
- int [request_type](#)
Request type.
- int [flags](#)
Other configuration flags.

6.3.1 Detailed Description

Structure holding configuration of a line request.

Definition at line 449 of file gpiod.h.

6.3.2 Field Documentation

6.3.2.1 consumer

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

Name of the consumer.

Definition at line 450 of file gpiod.h.

6.3.2.2 flags

```
int gpiod_line_request_config::flags
```

Other configuration flags.

Definition at line 454 of file gpiod.h.

6.3.2.3 request_type

```
int gpiod_line_request_config::request_type
```

Request type.

Definition at line 452 of file gpiod.h.

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

- [gpiod.h](#)

6.4 gpiod::line_bulk::iterator Class Reference

Iterator for iterating over lines held by [line_bulk](#).

```
#include <gpiod.hpp>
```

Public Member Functions

- [GPIOD_API iterator](#) (void)=default
Default constructor.
- [GPIOD_API iterator](#) (const [iterator](#) &other)=default
Copy constructor.
- [GPIOD_API iterator](#) ([iterator](#) &&other)=default
Move constructor.
- [GPIOD_API iterator](#) & [operator=](#) (const [iterator](#) &other)=default
Assignment operator.
- [GPIOD_API iterator](#) & [operator=](#) ([iterator](#) &&other)=default
Move assignment operator.
- [GPIOD_API ~iterator](#) (void)=default
Destructor.
- [GPIOD_API iterator](#) & [operator++](#) (void)
Advance the iterator by one element.
- [GPIOD_API](#) const [line](#) & [operator*](#) (void) const
Dereference current element.
- [GPIOD_API](#) const [line](#) * [operator->](#) (void) const
Member access operator.
- [GPIOD_API](#) bool [operator==](#) (const [iterator](#) &rhs) const noexcept
Check if this operator points to the same element.
- [GPIOD_API](#) bool [operator!=](#) (const [iterator](#) &rhs) const noexcept
Check if this operator doesn't point to the same element.

6.4.1 Detailed Description

Iterator for iterating over lines held by [line_bulk](#).

Definition at line 735 of file [gpiod.hpp](#).

6.4.2 Constructor & Destructor Documentation

6.4.2.1 [iterator\(\)](#) [1/3]

```
GPIOD_API gpiod::line_bulk::iterator::iterator (
    void ) [default]
```

Default constructor.

Builds an empty iterator object.

6.4.2.2 [iterator\(\)](#) [2/3]

```
GPIOD_API gpiod::line_bulk::iterator::iterator (
    const iterator & other ) [default]
```

Copy constructor.

Parameters

<i>other</i>	Other line_bulk iterator.
--------------	---

6.4.2.3 iterator() [3/3]

```
GPIO_API gpiod::line_bulk::iterator::iterator (
    iterator && other ) [default]
```

Move constructor.

Parameters

<i>other</i>	Other line_bulk iterator.
--------------	---

6.4.3 Member Function Documentation

6.4.3.1 operator!=(())

```
GPIO_API bool gpiod::line_bulk::iterator::operator!=(
    const iterator & rhs ) const [noexcept]
```

Check if this operator doesn't point to the same element.

Parameters

<i>rhs</i>	Right-hand side of the equation.
------------	----------------------------------

Returns

True if this iterator doesn't point to the same GPIO line, false otherwise.

6.4.3.2 operator*()

```
GPIO_API const line& gpiod::line_bulk::iterator::operator* (
    void ) const
```

Dereference current element.

Returns

Current GPIO line by reference.

6.4.3.3 operator++()

```
GPIO_API iterator& gpiod::line_bulk::iterator::operator++ (
    void )
```

Advance the iterator by one element.

Returns

Reference to this iterator.

6.4.3.4 operator->()

```
GPIO_API const line* gpiod::line_bulk::iterator::operator-> (
    void ) const
```

Member access operator.

Returns

Current GPIO line by pointer.

6.4.3.5 operator=() [1/2]

```
GPIO_API iterator& gpiod::line_bulk::iterator::operator= (
    const iterator & other ) [default]
```

Assignment operator.

Parameters

<i>other</i>	Other <code>line_bulk</code> iterator.
--------------	--

Returns

Reference to this iterator.

6.4.3.6 operator=() [2/2]

```
GPIO_API iterator& gpiod::line_bulk::iterator::operator= (
    iterator && other ) [default]
```

Move assignment operator.

Parameters

<i>other</i>	Other line_bulk iterator.
--------------	---

Returns

Reference to this iterator.

6.4.3.7 operator==()

```
GPIOD_API bool gpiod::line_bulk::iterator::operator== (
    const iterator & rhs ) const [noexcept]
```

Check if this operator points to the same element.

Parameters

<i>rhs</i>	Right-hand side of the equation.
------------	----------------------------------

Returns

True if this iterator points to the same GPIO line, false otherwise.

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

- [gpiod.hpp](#)

6.5 gpiod::line Class Reference

Represents a single GPIO line.

```
#include <gpiod.hpp>
```

Public Types

- enum : int { [DIRECTION_INPUT](#) = 1, [DIRECTION_OUTPUT](#) }
Possible direction settings.
- enum : int { [DRIVE_PUSH_PULL](#) = 1, [DRIVE_OPEN_DRAIN](#), [DRIVE_OPEN_SOURCE](#) }
Possible drive settings.
- enum : int { [BIAS_UNKNOWN](#) = 1, [BIAS_DISABLED](#), [BIAS_PULL_UP](#), [BIAS_PULL_DOWN](#) }
Possible bias settings.

Public Member Functions

- [GPIOD_API line](#) (void)
Default constructor.
- [GPIOD_API line](#) (const [line](#) &other)=default
Copy constructor.
- [GPIOD_API line](#) ([line](#) &&other)=default
Move constructor.
- [GPIOD_API line](#) & [operator=](#) (const [line](#) &other)=default
Assignment operator.
- [GPIOD_API line](#) & [operator=](#) ([line](#) &&other)=default
Move assignment operator.
- [GPIOD_API ~line](#) (void)=default
Destructor.
- [GPIOD_API](#) unsigned int [offset](#) (void) const
Get the offset of this line.
- [GPIOD_API](#) ::std::string [name](#) (void) const
Get the name of this line (if any).
- [GPIOD_API](#) ::std::string [consumer](#) (void) const
Get the consumer of this line (if any).
- [GPIOD_API](#) int [direction](#) (void) const
Get current direction of this line.
- [GPIOD_API](#) bool [is_active_low](#) (void) const
Check if this line's signal is inverted.
- [GPIOD_API](#) int [bias](#) (void) const
Get current bias of this line.
- [GPIOD_API](#) bool [is_used](#) (void) const
Check if this line is used by the kernel or other user space process.
- [GPIOD_API](#) int [drive](#) (void) const
Get current drive setting of this line.
- [GPIOD_API](#) void [request](#) (const [line_request](#) &config, int default_val=0) const
Request this line.
- [GPIOD_API](#) void [release](#) (void) const
Release the line if it was previously requested.
- [GPIOD_API](#) bool [is_requested](#) (void) const
Check if this user has ownership of this line.
- [GPIOD_API](#) int [get_value](#) (void) const
Read the line value.
- [GPIOD_API](#) void [set_value](#) (int val) const
Set the value of this line.
- [GPIOD_API](#) void [set_config](#) (int [direction](#), ::std::bitset< 32 > flags, int value=0) const
Set configuration of this line.
- [GPIOD_API](#) void [set_flags](#) (::std::bitset< 32 > flags) const
Set configuration flags of this line.
- [GPIOD_API](#) void [set_direction_input](#) () const
Change the direction this line to input.
- [GPIOD_API](#) void [set_direction_output](#) (int value=0) const
Change the direction this lines to output.
- [GPIOD_API](#) bool [event_wait](#) (const ::std::chrono::nanoseconds &timeout) const
Wait for an event on this line.
- [GPIOD_API](#) [line_event](#) [event_read](#) (void) const

- Read a line event.*

 - `GPIOD_API` `::std::vector< line_event > event_read_multiple` (void) const

Read multiple line events.

 - `GPIOD_API` `int event_get_fd` (void) const

Get the event file descriptor associated with this line.

 - `GPIOD_API` `const chip get_chip` (void) const

Get the parent chip.

 - `GPIOD_API` `void update` (void) const

Re-read the line info from the kernel.

 - `GPIOD_API` `void reset` (void)

Reset the state of this object.

 - `GPIOD_API` `bool operator==` (const `line` &rhs) const noexcept

Check if two line objects reference the same GPIO line.

 - `GPIOD_API` `bool operator!=` (const `line` &rhs) const noexcept

Check if two line objects reference different GPIO lines.

 - `GPIOD_API` `operator bool` (void) const noexcept

Check if this object holds a reference to any GPIO line.

 - `GPIOD_API` `bool operator!` (void) const noexcept

Check if this object doesn't reference any GPIO line.

6.5.1 Detailed Description

Represents a single GPIO line.

Internally this class holds a raw pointer to a GPIO line descriptor and a reference to the parent chip. All line resources are freed when the last reference to the parent chip is dropped.

Definition at line 246 of file `gpiod.hpp`.

6.5.2 Member Enumeration Documentation

6.5.2.1 anonymous enum

```
anonymous enum : int
```

Possible bias settings.

Enumerator

<code>BIAS_UNKNOWN</code>	Line's bias state is unknown.
<code>BIAS_DISABLED</code>	Line's internal bias is disabled.
<code>BIAS_PULL_UP</code>	Line's internal pull-up bias is enabled.
<code>BIAS_PULL_DOWN</code>	Line's internal pull-down bias is enabled.

Definition at line 490 of file `gpiod.hpp`.

6.5.2.2 anonymous enum

```
anonymous enum : int
```

Possible direction settings.

Enumerator

DIRECTION_INPUT	Line's direction setting is input.
DIRECTION_OUTPUT	Line's direction setting is output.

Definition at line 468 of file gpiod.hpp.

6.5.2.3 anonymous enum

```
anonymous enum : int
```

Possible drive settings.

Enumerator

DRIVE_PUSH_PULL	Drive setting is unknown.
DRIVE_OPEN_DRAIN	Line output is open-drain.
DRIVE_OPEN_SOURCE	Line output is open-source.

Definition at line 478 of file gpiod.hpp.

6.5.3 Constructor & Destructor Documentation

6.5.3.1 line() [1/3]

```
GPIOD_API gpiod::line::line (
    void )
```

Default constructor.

Creates an empty line object.

6.5.3.2 line() [2/3]

```
GPIOD_API gpiod::line::line (
    const line & other ) [default]
```

Copy constructor.

Parameters

<i>other</i>	Other line object.
--------------	--------------------

6.5.3.3 line() [3/3]

```
GPIOD_API gpiod::line::line (  
    line && other ) [default]
```

Move constructor.

Parameters

<i>other</i>	Other line object.
--------------	--------------------

6.5.4 Member Function Documentation**6.5.4.1 bias()**

```
GPIOD_API int gpiod::line::bias (  
    void ) const
```

Get current bias of this line.

Returns

Current bias setting.

6.5.4.2 consumer()

```
GPIOD_API ::std::string gpiod::line::consumer (  
    void ) const
```

Get the consumer of this line (if any).

Returns

Name of the consumer of this line or an empty string if it is unused.

6.5.4.3 direction()

```
GPIOD_API int gpiod::line::direction (
    void ) const
```

Get current direction of this line.

Returns

Current direction setting.

6.5.4.4 drive()

```
GPIOD_API int gpiod::line::drive (
    void ) const
```

Get current drive setting of this line.

Returns

Current drive setting.

6.5.4.5 event_get_fd()

```
GPIOD_API int gpiod::line::event_get_fd (
    void ) const
```

Get the event file descriptor associated with this line.

Returns

File descriptor number.

6.5.4.6 event_read()

```
GPIOD_API line_event gpiod::line::event_read (
    void ) const
```

Read a line event.

Returns

Line event object.

6.5.4.7 event_read_multiple()

```
GPIO_API ::std::vector<line_event> gpiod::line::event_read_multiple (
    void ) const
```

Read multiple line events.

Returns

Vector of line event objects.

6.5.4.8 event_wait()

```
GPIO_API bool gpiod::line::event_wait (
    const ::std::chrono::nanoseconds & timeout ) const
```

Wait for an event on this line.

Parameters

<i>timeout</i>	Time to wait before returning if no event occurred.
----------------	---

Returns

True if an event occurred and can be read, false if the wait timed out.

6.5.4.9 get_chip()

```
GPIO_API const chip gpiod::line::get_chip (
    void ) const
```

Get the parent chip.

Returns

Parent chip of this line.

6.5.4.10 get_value()

```
GPIO_API int gpiod::line::get_value (
    void ) const
```

Read the line value.

Returns

Current value (0 or 1).

6.5.4.11 is_active_low()

```
GPIOD_API bool gpiod::line::is_active_low (
    void ) const
```

Check if this line's signal is inverted.

Returns

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

6.5.4.12 is_requested()

```
GPIOD_API bool gpiod::line::is_requested (
    void ) const
```

Check if this user has ownership of this line.

Returns

True if the user has ownership of this line, false otherwise.

6.5.4.13 is_used()

```
GPIOD_API bool gpiod::line::is_used (
    void ) const
```

Check if this line is used by the kernel or other user space process.

Returns

True if this line is in use, false otherwise.

6.5.4.14 name()

```
GPIOD_API ::std::string gpiod::line::name (
    void ) const
```

Get the name of this line (if any).

Returns

Name of this line or an empty string if it is unnamed.

6.5.4.15 offset()

```
GPIOD_API unsigned int gpiod::line::offset (
    void ) const
```

Get the offset of this line.

Returns

Offset of this line.

6.5.4.16 operator bool()

```
GPIOD_API gpiod::line::operator bool (
    void ) const [explicit], [noexcept]
```

Check if this object holds a reference to any GPIO line.

Returns

True if this object references a GPIO line, false otherwise.

6.5.4.17 operator!()

```
GPIOD_API bool gpiod::line::operator! (
    void ) const [noexcept]
```

Check if this object doesn't reference any GPIO line.

Returns

True if this object doesn't reference any GPIO line, true otherwise.

6.5.4.18 operator!==(())

```
GPIOD_API bool gpiod::line::operator!= (
    const line & rhs ) const [noexcept]
```

Check if two line objects reference different GPIO lines.

Parameters

<i>rhs</i>	Right-hand side of the equation.
------------	----------------------------------

Returns

False if both objects reference the same line, true otherwise.

6.5.4.19 operator=() [1/2]

```
GPIOD_API line& gpiod::line::operator= (
    const line & other ) [default]
```

Assignment operator.

Parameters

<i>other</i>	Other line object.
--------------	--------------------

Returns

Reference to this object.

6.5.4.20 operator=() [2/2]

```
GPIOD_API line& gpiod::line::operator= (
    line && other ) [default]
```

Move assignment operator.

Parameters

<i>other</i>	Other line object.
--------------	--------------------

Returns

Reference to this object.

6.5.4.21 operator==()

```
GPIOD_API bool gpiod::line::operator== (
    const line & rhs ) const [noexcept]
```

Check if two line objects reference the same GPIO line.

Parameters

<i>rhs</i>	Right-hand side of the equation.
------------	----------------------------------

Returns

True if both objects reference the same line, false otherwise.

6.5.4.22 request()

```
GPIO_API void gpiod::line::request (
    const line_request & config,
    int default_val = 0 ) const
```

Request this line.

Parameters

<i>config</i>	Request config (see gpiod::line_request).
<i>default_val</i>	Default value - only matters for OUTPUT direction.

6.5.4.23 reset()

```
GPIO_API void gpiod::line::reset (
    void )
```

Reset the state of this object.

This is useful when the user needs to e.g. keep the [line_event](#) object but wants to drop the reference to the GPIO chip indirectly held by the line being the source of the event.

6.5.4.24 set_config()

```
GPIO_API void gpiod::line::set_config (
    int direction,
    ::std::bitset< 32 > flags,
    int value = 0 ) const
```

Set configuration of this line.

Parameters

<i>direction</i>	New direction.
<i>flags</i>	Replacement flags.
<i>value</i>	New value (0 or 1) - only matters for OUTPUT direction.

6.5.4.25 set_direction_output()

```
GPIO_API void gpio::line::set_direction_output (
    int value = 0 ) const
```

Change the direction this lines to output.

Parameters

<i>value</i>	New value (0 or 1).
--------------	---------------------

6.5.4.26 set_flags()

```
GPIO_API void gpio::line::set_flags (
    ::std::bitset< 32 > flags ) const
```

Set configuration flags of this line.

Parameters

<i>flags</i>	Replacement flags.
--------------	--------------------

6.5.4.27 set_value()

```
GPIO_API void gpio::line::set_value (
    int val ) const
```

Set the value of this line.

Parameters

<i>val</i>	New value (0 or 1).
------------	---------------------

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

- [gpio.hpp](#)

6.6 gpio::line_bulk Class Reference

Represents a set of GPIO lines.

```
#include <gpio.hpp>
```

Data Structures

- class [iterator](#)
Iterator for iterating over lines held by [line_bulk](#).

Public Member Functions

- [GPIOD_API line_bulk](#) (void)=default
Default constructor.
- [GPIOD_API line_bulk](#) (const ::std::vector< [line](#) > &lines)
Construct a [line_bulk](#) from a vector of lines.
- [GPIOD_API line_bulk](#) (const [line_bulk](#) &other)=default
Copy constructor.
- [GPIOD_API line_bulk](#) ([line_bulk](#) &&other)=default
Move constructor.
- [GPIOD_API line_bulk & operator=](#) (const [line_bulk](#) &other)=default
Assignment operator.
- [GPIOD_API line_bulk & operator=](#) ([line_bulk](#) &&other)=default
Move assignment operator.
- [GPIOD_API ~line_bulk](#) (void)=default
Destructor.
- [GPIOD_API void append](#) (const [line](#) &new_line)
Add a line to this [line_bulk](#) object.
- [GPIOD_API line & get](#) (unsigned int index)
Get the line at given offset.
- [GPIOD_API line & operator\[\]](#) (unsigned int index)
Get the line at given offset without bounds checking.
- [GPIOD_API unsigned int size](#) (void) const noexcept
Get the number of lines currently held by this object.
- [GPIOD_API bool empty](#) (void) const noexcept
Check if this [line_bulk](#) doesn't hold any lines.
- [GPIOD_API void clear](#) (void)
Remove all lines from this object.
- [GPIOD_API void request](#) (const [line_request](#) &config, const ::std::vector< int > default_vals=::std::vector< int >()) const
Request all lines held by this object.
- [GPIOD_API void release](#) (void) const
Release all lines held by this object.
- [GPIOD_API ::std::vector< int > get_values](#) (void) const
Read values from all lines held by this object.
- [GPIOD_API void set_values](#) (const ::std::vector< int > &values) const
Set values of all lines held by this object.
- [GPIOD_API void set_config](#) (int direction, ::std::bitset< 32 > flags, const ::std::vector< int > values=::std::vector< int >()) const
Set configuration of all lines held by this object.
- [GPIOD_API void set_flags](#) (::std::bitset< 32 > flags) const
Set configuration flags of all lines held by this object.
- [GPIOD_API void set_direction_input](#) () const
Change the direction all lines held by this object to input.
- [GPIOD_API void set_direction_output](#) (const ::std::vector< int > &values) const

- Change the direction all lines held by this object to output.*

 - [GPIOD_API line_bulk event_wait](#) (const ::std::chrono::nanoseconds &timeout) const
Poll the set of lines for line events.
 - [GPIOD_API operator bool](#) (void) const noexcept
Check if this object holds any lines.
 - [GPIOD_API bool operator!](#) (void) const noexcept
Check if this object doesn't hold any lines.
 - [GPIOD_API iterator begin](#) (void) noexcept
Returns an iterator to the first line.
 - [GPIOD_API iterator end](#) (void) noexcept
Returns an iterator to the element following the last line.

Static Public Attributes

- static [GPIOD_API](#) const unsigned int [MAX_LINES](#)
Max number of lines that this object can hold.

6.6.1 Detailed Description

Represents a set of GPIO lines.

Internally an object of this class stores an array of line objects owned by a single chip.

Definition at line 560 of file `gpiod.hpp`.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 `line_bulk()` [1/4]

```
GPIOD_API gpiod::line_bulk::line_bulk (
    void ) [default]
```

Default constructor.

Creates an empty [line_bulk](#) object.

6.6.2.2 `line_bulk()` [2/4]

```
GPIOD_API gpiod::line_bulk::line_bulk (
    const ::std::vector< line > & lines )
```

Construct a [line_bulk](#) from a vector of lines.

Parameters

<i>lines</i>	Vector of gpiod::line objects.
--------------	--

Note

All lines must be owned by the same GPIO chip.

6.6.2.3 `line_bulk()` [3/4]

```
GPIOD_API gpiod::line_bulk::line_bulk (  
    const line\_bulk & other ) [default]
```

Copy constructor.

Parameters

<i>other</i>	Other line_bulk object.
--------------	---

6.6.2.4 `line_bulk()` [4/4]

```
GPIOD_API gpiod::line_bulk::line_bulk (  
    line\_bulk && other ) [default]
```

Move constructor.

Parameters

<i>other</i>	Other line_bulk object.
--------------	---

6.6.3 Member Function Documentation

6.6.3.1 `append()`

```
GPIOD_API void gpiod::line_bulk::append (  
    const line & new_line )
```

Add a line to this [line_bulk](#) object.

Parameters

<code>new_line</code>	Line to add.
-----------------------	--------------

Note

The new line must be owned by the same chip as all the other lines already held by this [line_bulk](#) object.

6.6.3.2 begin()

```
GPIOD_API iterator gpiod::line_bulk::begin (
    void ) [noexcept]
```

Returns an iterator to the first line.

Returns

A [line_bulk](#) iterator.

6.6.3.3 empty()

```
GPIOD_API bool gpiod::line_bulk::empty (
    void ) const [noexcept]
```

Check if this [line_bulk](#) doesn't hold any lines.

Returns

True if this object is empty, false otherwise.

6.6.3.4 end()

```
GPIOD_API iterator gpiod::line_bulk::end (
    void ) [noexcept]
```

Returns an iterator to the element following the last line.

Returns

A [line_bulk](#) iterator.

6.6.3.5 event_wait()

```
GPIOD_API line_bulk gpiod::line_bulk::event_wait (
    const ::std::chrono::nanoseconds & timeout ) const
```

Poll the set of lines for line events.

Parameters

<i>timeout</i>	Number of nanoseconds to wait before returning an empty line_bulk .
----------------	---

Returns

Returns a [line_bulk](#) object containing lines on which events occurred.

6.6.3.6 get()

```
GPIOD_API line& gpiod::line_bulk::get (
    unsigned int index )
```

Get the line at given offset.

Parameters

<i>index</i>	Index of the line to get.
--------------	---------------------------

Returns

Reference to the line object.

Note

This method will throw if index is equal or greater than the number of lines currently held by this bulk.

6.6.3.7 get_values()

```
GPIOD_API ::std::vector<int> gpiod::line_bulk::get_values (
    void ) const
```

Read values from all lines held by this object.

Returns

Vector containing line values the order of which corresponds with the order of lines in the internal array.

6.6.3.8 operator bool()

```
GPIOD_API gpiod::line_bulk::operator bool (
    void ) const [explicit], [noexcept]
```

Check if this object holds any lines.

Returns

True if this [line_bulk](#) holds at least one line, false otherwise.

6.6.3.9 operator!()

```
GPIOD_API bool gpiod::line_bulk::operator! (
    void ) const [noexcept]
```

Check if this object doesn't hold any lines.

Returns

True if this [line_bulk](#) is empty, false otherwise.

6.6.3.10 operator=() [1/2]

```
GPIOD_API line_bulk& gpiod::line_bulk::operator= (
    const line_bulk & other ) [default]
```

Assignment operator.

Parameters

<i>other</i>	Other line_bulk object.
--------------	---

Returns

Reference to this object.

6.6.3.11 operator=() [2/2]

```
GPIOD_API line_bulk& gpiod::line_bulk::operator= (
    line_bulk && other ) [default]
```

Move assignment operator.

Parameters

<i>other</i>	Other line_bulk object.
--------------	---

Returns

Reference to this object.

6.6.3.12 operator[]()

```
GPIOD_API line& gpiod::line_bulk::operator[] (
    unsigned int index )
```

Get the line at given offset without bounds checking.

Parameters

<i>index</i>	Offset of the line to get.
--------------	----------------------------

Returns

Reference to the line object.

Note

No bounds checking is performed.

6.6.3.13 request()

```
GPIOD_API void gpiod::line_bulk::request (
    const line_request & config,
    const ::std::vector< int > default_vals = ::std::vector< int >() ) const
```

Request all lines held by this object.

Parameters

<i>config</i>	Request config (see gpiod::line_request).
<i>default_vals</i>	Vector of default values. Only relevant for output direction requests.

6.6.3.14 set_config()

```

GPIO_API void gpiod::line_bulk::set_config (
    int direction,
    ::std::bitset< 32 > flags,
    const ::std::vector< int > values = ::std::vector< int >() ) const

```

Set configuration of all lines held by this object.

Parameters

<i>direction</i>	New direction.
<i>flags</i>	Replacement flags.
<i>values</i>	Vector of values to set. Must be the same size as the number of lines held by this line_bulk . Only relevant for output direction requests.

6.6.3.15 set_direction_output()

```

GPIO_API void gpiod::line_bulk::set_direction_output (
    const ::std::vector< int > & values ) const

```

Change the direction all lines held by this object to output.

Parameters

<i>values</i>	Vector of values to set. Must be the same size as the number of lines held by this line_bulk .
---------------	--

6.6.3.16 set_flags()

```

GPIO_API void gpiod::line_bulk::set_flags (
    ::std::bitset< 32 > flags ) const

```

Set configuration flags of all lines held by this object.

Parameters

<i>flags</i>	Replacement flags.
--------------	--------------------

6.6.3.17 set_values()

```

GPIO_API void gpiod::line_bulk::set_values (
    const ::std::vector< int > & values ) const

```

Set values of all lines held by this object.

Parameters

<i>values</i>	Vector of values to set. Must be the same size as the number of lines held by this line_bulk .
---------------	--

6.6.3.18 size()

```
GPIOD_API unsigned int gpiod::line_bulk::size (
    void ) const [noexcept]
```

Get the number of lines currently held by this object.

Returns

Number of elements in this [line_bulk](#).

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

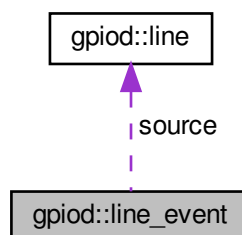
- [gpiod.hpp](#)

6.7 gpiod::line_event Struct Reference

Describes a single GPIO line event.

```
#include <gpiod.hpp>
```

Collaboration diagram for gpiod::line_event:



Public Types

- enum : int { [RISING_EDGE](#) = 1, [FALLING_EDGE](#) }
- Possible event types.*

Data Fields

- `::std::chrono::nanoseconds` [timestamp](#)
Best estimate of time of event occurrence in nanoseconds.
- `int` [event_type](#)
Type of the event that occurred.
- [line source](#)
Line object referencing the GPIO line on which the event occurred.

6.7.1 Detailed Description

Describes a single GPIO line event.

Definition at line 534 of file `gpiod.hpp`.

6.7.2 Member Enumeration Documentation

6.7.2.1 anonymous enum

```
anonymous enum : int
```

Possible event types.

Enumerator

<code>RISING_EDGE</code>	Rising edge event.
<code>FALLING_EDGE</code>	Falling edge event.

Definition at line 539 of file `gpiod.hpp`.

6.7.3 Field Documentation

6.7.3.1 event_type

```
int gpiod::line_event::event_type
```

Type of the event that occurred.

Definition at line 548 of file `gpiod.hpp`.

6.7.3.2 source

`line` `gpiod::line_event::source`

Line object referencing the GPIO line on which the event occurred.

Definition at line 550 of file `gpiod.hpp`.

6.7.3.3 timestamp

`::std::chrono::nanoseconds` `gpiod::line_event::timestamp`

Best estimate of time of event occurrence in nanoseconds.

Definition at line 546 of file `gpiod.hpp`.

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

- [gpiod.hpp](#)

6.8 gpiod::line_iter Class Reference

Allows to iterate over all lines owned by a GPIO chip.

```
#include <gpiod.hpp>
```

Public Member Functions

- [GPIOD_API line_iter](#) (void)=default
Default constructor.
- [GPIOD_API line_iter](#) (const [chip](#) &owner)
Constructor.
- [GPIOD_API line_iter](#) (const [line_iter](#) &other)=default
Copy constructor.
- [GPIOD_API line_iter](#) ([line_iter](#) &&other)=default
Move constructor.
- [GPIOD_API line_iter](#) & [operator=](#) (const [line_iter](#) &other)=default
Assignment operator.
- [GPIOD_API line_iter](#) & [operator=](#) ([line_iter](#) &&other)=default
Move assignment operator.
- [GPIOD_API ~line_iter](#) (void)=default
Destructor.
- [GPIOD_API line_iter](#) & [operator++](#) (void)
Advance the iterator by one element.
- [GPIOD_API](#) const [line](#) & [operator*](#) (void) const
Dereference current element.
- [GPIOD_API](#) const [line](#) * [operator->](#) (void) const
Member access operator.
- [GPIOD_API](#) bool [operator==](#) (const [line_iter](#) &rhs) const noexcept
Check if this operator points to the same element.
- [GPIOD_API](#) bool [operator!=](#) (const [line_iter](#) &rhs) const noexcept
Check if this operator doesn't point to the same element.

6.8.1 Detailed Description

Allows to iterate over all lines owned by a GPIO chip.

Definition at line 864 of file gpiod.hpp.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 line_iter() [1/4]

```
GPIOD_API gpiod::line_iter::line_iter (  
    void ) [default]
```

Default constructor.

Creates the end iterator.

6.8.2.2 line_iter() [2/4]

```
GPIOD_API gpiod::line_iter::line_iter (  
    const chip & owner )
```

Constructor.

Creates the begin iterator.

Parameters

<i>owner</i>	Chip owning the GPIO lines over which we want to iterate.
--------------	---

6.8.2.3 line_iter() [3/4]

```
GPIOD_API gpiod::line_iter::line_iter (  
    const line_iter & other ) [default]
```

Copy constructor.

Parameters

<i>other</i>	Other line iterator.
--------------	----------------------

6.8.2.4 `line_iter()` [4/4]

```
GPIOD_API gpiod::line_iter::line_iter (
    line_iter && other ) [default]
```

Move constructor.

Parameters

<i>other</i>	Other line iterator.
--------------	----------------------

6.8.3 Member Function Documentation

6.8.3.1 `operator!=()`

```
GPIOD_API bool gpiod::line_iter::operator!= (
    const line_iter & rhs ) const [noexcept]
```

Check if this operator doesn't point to the same element.

Parameters

<i>rhs</i>	Right-hand side of the equation.
------------	----------------------------------

Returns

True if this iterator doesn't point to the same `line_iter`, false otherwise.

6.8.3.2 `operator*()`

```
GPIOD_API const line& gpiod::line_iter::operator* (
    void ) const
```

Dereference current element.

Returns

Current GPIO line by reference.

6.8.3.3 operator++()

```
GPIOD_API line_iter& gpiod::line_iter::operator++ (
    void )
```

Advance the iterator by one element.

Returns

Reference to this iterator.

6.8.3.4 operator->()

```
GPIOD_API const line* gpiod::line_iter::operator-> (
    void ) const
```

Member access operator.

Returns

Current GPIO line by pointer.

6.8.3.5 operator=() [1/2]

```
GPIOD_API line_iter& gpiod::line_iter::operator= (
    const line_iter & other ) [default]
```

Assignment operator.

Parameters

<i>other</i>	Other line iterator.
--------------	----------------------

Returns

Reference to this [line_iter](#).

6.8.3.6 operator=() [2/2]

```
GPIOD_API line_iter& gpiod::line_iter::operator= (
    line_iter && other ) [default]
```

Move assignment operator.

Parameters

<i>other</i>	Other line iterator.
--------------	----------------------

Returns

Reference to this [line_iter](#).

6.8.3.7 operator==()

```
GPIOD_API bool gpiod::line_iter::operator==(
    const line_iter & rhs ) const [noexcept]
```

Check if this operator points to the same element.

Parameters

<i>rhs</i>	Right-hand side of the equation.
------------	----------------------------------

Returns

True if this iterator points to the same [line_iter](#), false otherwise.

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

- [gpiod.hpp](#)

6.9 gpiod::line_request Struct Reference

Stores the configuration for line requests.

```
#include <gpiod.hpp>
```

Public Types

- enum : int {
[DIRECTION_AS_IS](#) = 1, [DIRECTION_INPUT](#), [DIRECTION_OUTPUT](#), [EVENT_FALLING_EDGE](#),
[EVENT_RISING_EDGE](#), [EVENT_BOTH_EDGES](#) }
Request types.

Data Fields

- ::std::string [consumer](#)
Consumer name to pass to the request.
- int [request_type](#)
Type of the request.
- ::std::bitset< 32 > [flags](#)
Additional request flags.

Static Public Attributes

- static [GPIOD_API](#) const ::std::bitset< 32 > [FLAG_ACTIVE_LOW](#)
Set the active state to 'low' (high is the default).
- static [GPIOD_API](#) const ::std::bitset< 32 > [FLAG_OPEN_SOURCE](#)
The line is an open-source port.
- static [GPIOD_API](#) const ::std::bitset< 32 > [FLAG_OPEN_DRAIN](#)
The line is an open-drain port.
- static [GPIOD_API](#) const ::std::bitset< 32 > [FLAG_BIAS_DISABLED](#)
The line has neither pull-up nor pull-down resistor enabled.
- static [GPIOD_API](#) const ::std::bitset< 32 > [FLAG_BIAS_PULL_DOWN](#)
The line has a configurable pull-down resistor enabled.
- static [GPIOD_API](#) const ::std::bitset< 32 > [FLAG_BIAS_PULL_UP](#)
The line has a configurable pull-up resistor enabled.

6.9.1 Detailed Description

Stores the configuration for line requests.

Definition at line 198 of file gpiod.hpp.

6.9.2 Member Enumeration Documentation

6.9.2.1 anonymous enum

```
anonymous enum : int
```

Request types.

Enumerator

DIRECTION_AS_IS	Request for values, don't change the direction.
DIRECTION_INPUT	Request for reading line values.
DIRECTION_OUTPUT	Request for driving the GPIO lines.
EVENT_FALLING_EDGE	Listen for falling edge events.
EVENT_RISING_EDGE	Listen for rising edge events.
EVENT_BOTH_EDGES	Listen for all types of events.

Definition at line 203 of file gpiod.hpp.

6.9.3 Field Documentation

6.9.3.1 consumer

```
::std::string gpiod::line_request::consumer
```

Consumer name to pass to the request.

Definition at line 231 of file gpiod.hpp.

6.9.3.2 FLAG_ACTIVE_LOW

```
GPIOD_API const ::std::bitset<32> gpiod::line_request::FLAG_ACTIVE_LOW [static]
```

Set the active state to 'low' (high is the default).

Definition at line 218 of file gpiod.hpp.

6.9.3.3 FLAG_BIAS_DISABLED

```
GPIOD_API const ::std::bitset<32> gpiod::line_request::FLAG_BIAS_DISABLED [static]
```

The line has neither pull-up nor pull-down resistor enabled.

Definition at line 224 of file gpiod.hpp.

6.9.3.4 FLAG_BIAS_PULL_DOWN

```
GPIOD_API const ::std::bitset<32> gpiod::line_request::FLAG_BIAS_PULL_DOWN [static]
```

The line has a configurable pull-down resistor enabled.

Definition at line 226 of file gpiod.hpp.

6.9.3.5 FLAG_BIAS_PULL_UP

```
GPIOD_API const ::std::bitset<32> gpiod::line_request::FLAG_BIAS_PULL_UP [static]
```

The line has a configurable pull-up resistor enabled.

Definition at line 228 of file gpiod.hpp.

6.9.3.6 FLAG_OPEN_DRAIN

```
GPIOD_API const ::std::bitset<32> gpiod::line_request::FLAG_OPEN_DRAIN [static]
```

The line is an open-drain port.

Definition at line 222 of file gpiod.hpp.

6.9.3.7 FLAG_OPEN_SOURCE

```
GPIOD_API const ::std::bitset<32> gpiod::line_request::FLAG_OPEN_SOURCE [static]
```

The line is an open-source port.

Definition at line 220 of file gpiod.hpp.

6.9.3.8 flags

```
::std::bitset<32> gpiod::line_request::flags
```

Additional request flags.

Definition at line 235 of file gpiod.hpp.

6.9.3.9 request_type

```
int gpiod::line_request::request_type
```

Type of the request.

Definition at line 233 of file gpiod.hpp.

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

- [gpiod.hpp](#)

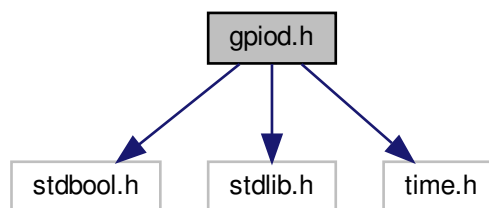
Chapter 7

File Documentation

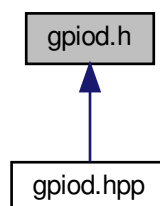
7.1 gpiod.h File Reference

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

Include dependency graph for gpiod.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gpiod_line_request_config](#)
Structure holding configuration of a line request.
- struct [gpiod_line_event](#)
Structure holding event info.

Macros

- #define [GPIOD_API __attribute__\(\(visibility\("default"\)\)\)](#)
Makes symbol visible.
- #define [GPIOD_BIT\(nr\)](#) (1UL << (nr))
Shift 1 by given offset.

Typedefs

- typedef int(* [gpiod_line_bulk_foreach_cb](#)) (struct [gpiod_line](#) *, void *)
Signature of the callback passed to [gpiod_line_bulk_foreach_line](#).

Enumerations

- enum { [GPIOD_LINE_BULK_CB_NEXT](#) = 0, [GPIOD_LINE_BULK_CB_STOP](#) }
Values returned by the callback passed to [gpiod_line_bulk_foreach_line](#).
- enum { [GPIOD_LINE_DIRECTION_INPUT](#) = 1, [GPIOD_LINE_DIRECTION_OUTPUT](#) }
Possible direction settings.
- enum { [GPIOD_LINE_DRIVE_PUSH_PULL](#) = 1, [GPIOD_LINE_DRIVE_OPEN_DRAIN](#), [GPIOD_LINE_DRIVE_OPEN_SOURCE](#) }
Possible drive settings.
- enum { [GPIOD_LINE_BIAS_UNKNOWN](#) = 1, [GPIOD_LINE_BIAS_DISABLED](#), [GPIOD_LINE_BIAS_PULL_UP](#), [GPIOD_LINE_BIAS_PULL_DOWN](#) }
Possible internal bias settings.
- enum { [GPIOD_LINE_REQUEST_DIRECTION_AS_IS](#) = 1, [GPIOD_LINE_REQUEST_DIRECTION_INPUT](#), [GPIOD_LINE_REQUEST_DIRECTION_OUTPUT](#), [GPIOD_LINE_REQUEST_EVENT_FALLING_EDGE](#), [GPIOD_LINE_REQUEST_EVENT_RISING_EDGE](#), [GPIOD_LINE_REQUEST_EVENT_BOTH_EDGES](#) }
Available types of requests.
- enum { [GPIOD_LINE_REQUEST_FLAG_OPEN_DRAIN](#) = [GPIOD_BIT](#)(0), [GPIOD_LINE_REQUEST_FLAG_OPEN_SOURCE](#) = [GPIOD_BIT](#)(1), [GPIOD_LINE_REQUEST_FLAG_ACTIVE_LOW](#) = [GPIOD_BIT](#)(2), [GPIOD_LINE_REQUEST_FLAG_BIAS_DISABLED](#) = [GPIOD_BIT](#)(3), [GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_DOWN](#) = [GPIOD_BIT](#)(4), [GPIOD_LINE_REQUEST_FLAG_BIAS_PULL_UP](#) = [GPIOD_BIT](#)(5) }
Miscellaneous GPIO request flags.
- enum { [GPIOD_LINE_EVENT_RISING_EDGE](#) = 1, [GPIOD_LINE_EVENT_FALLING_EDGE](#) }
Event types.

Functions

- bool [gpiod_is_gpiochip_device](#) (const char *path) [GPIOD_API](#)
Check if the file pointed to by path is a GPIO chip character device.
- struct gpiod_chip * [gpiod_chip_open](#) (const char *path) [GPIOD_API](#)
Open a gpiochip by path.
- void [gpiod_chip_close](#) (struct gpiod_chip *chip) [GPIOD_API](#)
Close a GPIO chip handle and release all allocated resources.
- const char * [gpiod_chip_name](#) (struct gpiod_chip *chip) [GPIOD_API](#)
Get the GPIO chip name as represented in the kernel.
- const char * [gpiod_chip_label](#) (struct gpiod_chip *chip) [GPIOD_API](#)
Get the GPIO chip label as represented in the kernel.
- unsigned int [gpiod_chip_num_lines](#) (struct gpiod_chip *chip) [GPIOD_API](#)
Get the number of GPIO lines exposed by this chip.
- struct gpiod_line * [gpiod_chip_get_line](#) (struct gpiod_chip *chip, unsigned int offset) [GPIOD_API](#)
Get the handle to the GPIO line at given offset.
- struct gpiod_line_bulk * [gpiod_chip_get_lines](#) (struct gpiod_chip *chip, unsigned int *offsets, unsigned int num_offsets) [GPIOD_API](#)
Retrieve a set of lines and store them in a line bulk object.
- struct gpiod_line_bulk * [gpiod_chip_get_all_lines](#) (struct gpiod_chip *chip) [GPIOD_API](#)
Retrieve all lines exposed by a chip and store them in a bulk object.
- struct gpiod_line_bulk * [gpiod_chip_find_line](#) (struct gpiod_chip *chip, const char *name) [GPIOD_API](#)
Find all GPIO lines by name among lines exposed by this GPIO chip.
- struct gpiod_line * [gpiod_chip_find_line_unique](#) (struct gpiod_chip *chip, const char *name) [GPIOD_API](#)
Find a unique line by name among lines exposed by this GPIO chip.
- struct gpiod_line_bulk * [gpiod_line_bulk_new](#) (unsigned int max_lines) [GPIOD_API](#)
Allocate and initialize a new line bulk object.
- void [gpiod_line_bulk_reset](#) (struct gpiod_line_bulk *bulk) [GPIOD_API](#)
Reset a bulk object.
- void [gpiod_line_bulk_free](#) (struct gpiod_line_bulk *bulk) [GPIOD_API](#)
Release all resources allocated for this bulk object.
- int [gpiod_line_bulk_add_line](#) (struct gpiod_line_bulk *bulk, struct gpiod_line *line) [GPIOD_API](#)
Add a single line to a GPIO bulk object.
- struct gpiod_line * [gpiod_line_bulk_get_line](#) (struct gpiod_line_bulk *bulk, unsigned int index) [GPIOD_API](#)
Retrieve the line handle from a line bulk object at given index.
- unsigned int [gpiod_line_bulk_num_lines](#) (struct gpiod_line_bulk *bulk) [GPIOD_API](#)
Retrieve the number of GPIO lines held by this line bulk object.
- void [gpiod_line_bulk_foreach_line](#) (struct gpiod_line_bulk *bulk, [gpiod_line_bulk_foreach_cb](#) func, void *data) [GPIOD_API](#)
Iterate over all lines held by this bulk object.
- unsigned int [gpiod_line_offset](#) (struct gpiod_line *line) [GPIOD_API](#)
Read the GPIO line offset.
- const char * [gpiod_line_name](#) (struct gpiod_line *line) [GPIOD_API](#)
Read the GPIO line name.
- const char * [gpiod_line_consumer](#) (struct gpiod_line *line) [GPIOD_API](#)
Read the GPIO line consumer name.
- int [gpiod_line_direction](#) (struct gpiod_line *line) [GPIOD_API](#)
Read the GPIO line direction setting.
- bool [gpiod_line_is_active_low](#) (struct gpiod_line *line) [GPIOD_API](#)
Check if the signal of this line is inverted.
- int [gpiod_line_bias](#) (struct gpiod_line *line) [GPIOD_API](#)

- Read the GPIO line bias setting.*

 - bool [gpiod_line_is_used](#) (struct [gpiod_line](#) *line) [GPIO_API](#)
- Check if the line is currently in use.*

 - int [gpiod_line_drive](#) (struct [gpiod_line](#) *line) [GPIO_API](#)
- Read the GPIO line drive setting.*

 - int [gpiod_line_update](#) (struct [gpiod_line](#) *line) [GPIO_API](#)
- Re-read the line info.*

 - struct [gpiod_chip](#) * [gpiod_line_get_chip](#) (struct [gpiod_line](#) *line) [GPIO_API](#)
- Get the handle to the GPIO chip controlling this line.*

 - int [gpiod_line_request](#) (struct [gpiod_line](#) *line, const struct [gpiod_line_request_config](#) *config, int default_val) [GPIO_API](#)
- Reserve a single line.*

 - int [gpiod_line_request_input](#) (struct [gpiod_line](#) *line, const char *consumer) [GPIO_API](#)
- Reserve a single line, set the direction to input.*

 - int [gpiod_line_request_output](#) (struct [gpiod_line](#) *line, const char *consumer, int default_val) [GPIO_API](#)
- Reserve a single line, set the direction to output.*

 - int [gpiod_line_request_rising_edge_events](#) (struct [gpiod_line](#) *line, const char *consumer) [GPIO_API](#)
- Request rising edge event notifications on a single line.*

 - int [gpiod_line_request_falling_edge_events](#) (struct [gpiod_line](#) *line, const char *consumer) [GPIO_API](#)
- Request falling edge event notifications on a single line.*

 - int [gpiod_line_request_both_edges_events](#) (struct [gpiod_line](#) *line, const char *consumer) [GPIO_API](#)
- Request all event type notifications on a single line.*

 - int [gpiod_line_request_input_flags](#) (struct [gpiod_line](#) *line, const char *consumer, int flags) [GPIO_API](#)
- Reserve a single line, set the direction to input.*

 - int [gpiod_line_request_output_flags](#) (struct [gpiod_line](#) *line, const char *consumer, int flags, int default_val) [GPIO_API](#)
- Reserve a single line, set the direction to output.*

 - int [gpiod_line_request_rising_edge_events_flags](#) (struct [gpiod_line](#) *line, const char *consumer, int flags) [GPIO_API](#)
- Request rising edge event notifications on a single line.*

 - int [gpiod_line_request_falling_edge_events_flags](#) (struct [gpiod_line](#) *line, const char *consumer, int flags) [GPIO_API](#)
- Request falling edge event notifications on a single line.*

 - int [gpiod_line_request_both_edges_events_flags](#) (struct [gpiod_line](#) *line, const char *consumer, int flags) [GPIO_API](#)
- Request all event type notifications on a single line.*

 - int [gpiod_line_request_bulk](#) (struct [gpiod_line_bulk](#) *bulk, const struct [gpiod_line_request_config](#) *config, const int *default_vals) [GPIO_API](#)
- Reserve a set of GPIO lines.*

 - int [gpiod_line_request_bulk_input](#) (struct [gpiod_line_bulk](#) *bulk, const char *consumer) [GPIO_API](#)
- Reserve a set of GPIO lines, set the direction to input.*

 - int [gpiod_line_request_bulk_output](#) (struct [gpiod_line_bulk](#) *bulk, const char *consumer, const int *default_vals) [GPIO_API](#)
- Reserve a set of GPIO lines, set the direction to output.*

 - int [gpiod_line_request_bulk_rising_edge_events](#) (struct [gpiod_line_bulk](#) *bulk, const char *consumer) [GPIO_API](#)
- Request rising edge event notifications on a set of lines.*

 - int [gpiod_line_request_bulk_falling_edge_events](#) (struct [gpiod_line_bulk](#) *bulk, const char *consumer) [GPIO_API](#)
- Request falling edge event notifications on a set of lines.*

 - int [gpiod_line_request_bulk_both_edges_events](#) (struct [gpiod_line_bulk](#) *bulk, const char *consumer) [GPIO_API](#)

- Request all event type notifications on a set of lines.*

 - int [gpiod_line_request_bulk_input_flags](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, int flags) [GPIO↔IOD_API](#)

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

 - int [gpiod_line_request_bulk_output_flags](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, int flags, const int *default_vals) [GPIO_API](#)

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

 - int [gpiod_line_request_bulk_rising_edge_events_flags](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, int flags) [GPIO_API](#)

Request rising edge event notifications on a set of lines.

 - int [gpiod_line_request_bulk_falling_edge_events_flags](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, int flags) [GPIO_API](#)

Request falling edge event notifications on a set of lines.

 - int [gpiod_line_request_bulk_both_edges_events_flags](#) (struct `gpiod_line_bulk` *bulk, const char *consumer, int flags) [GPIO_API](#)

Request all event type notifications on a set of lines.

 - void [gpiod_line_release](#) (struct `gpiod_line` *line) [GPIO_API](#)

Release a previously reserved line.

 - void [gpiod_line_release_bulk](#) (struct `gpiod_line_bulk` *bulk) [GPIO_API](#)

Release a set of previously reserved lines.

 - bool [gpiod_line_is_requested](#) (struct `gpiod_line` *line) [GPIO_API](#)

Check if the calling user has ownership of this line.

 - bool [gpiod_line_is_free](#) (struct `gpiod_line` *line) [GPIO_API](#)

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

 - int [gpiod_line_get_value](#) (struct `gpiod_line` *line) [GPIO_API](#)

Read current value of a single GPIO line.

 - int [gpiod_line_get_value_bulk](#) (struct `gpiod_line_bulk` *bulk, int *values) [GPIO_API](#)

Read current values of a set of GPIO lines.

 - int [gpiod_line_set_value](#) (struct `gpiod_line` *line, int value) [GPIO_API](#)

Set the value of a single GPIO line.

 - int [gpiod_line_set_value_bulk](#) (struct `gpiod_line_bulk` *bulk, const int *values) [GPIO_API](#)

Set the values of a set of GPIO lines.

 - int [gpiod_line_set_config](#) (struct `gpiod_line` *line, int direction, int flags, int value) [GPIO_API](#)

Update the configuration of a single GPIO line.

 - int [gpiod_line_set_config_bulk](#) (struct `gpiod_line_bulk` *bulk, int direction, int flags, const int *values) [GPIO↔OD_API](#)

Update the configuration of a set of GPIO lines.

 - int [gpiod_line_set_flags](#) (struct `gpiod_line` *line, int flags) [GPIO_API](#)

Update the configuration flags of a single GPIO line.

 - int [gpiod_line_set_flags_bulk](#) (struct `gpiod_line_bulk` *bulk, int flags) [GPIO_API](#)

Update the configuration flags of a set of GPIO lines.

 - int [gpiod_line_set_direction_input](#) (struct `gpiod_line` *line) [GPIO_API](#)

Set the direction of a single GPIO line to input.

 - int [gpiod_line_set_direction_input_bulk](#) (struct `gpiod_line_bulk` *bulk) [GPIO_API](#)

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

 - int [gpiod_line_set_direction_output](#) (struct `gpiod_line` *line, int value) [GPIO_API](#)

Set the direction of a single GPIO line to output.

 - int [gpiod_line_set_direction_output_bulk](#) (struct `gpiod_line_bulk` *bulk, const int *values) [GPIO_API](#)

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

 - int [gpiod_line_event_wait](#) (struct `gpiod_line` *line, const struct `timespec` *timeout) [GPIO_API](#)

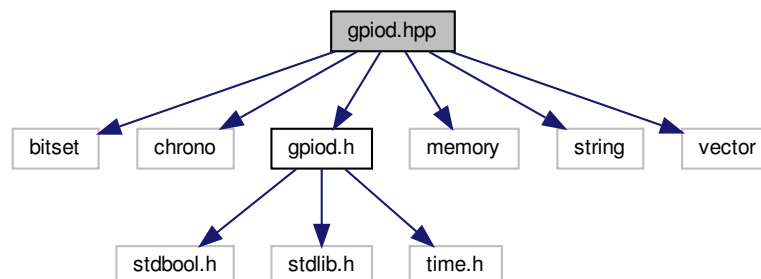
Wait for an event on a single line.

- int [gpiod_line_event_wait_bulk](#) (struct [gpiod_line_bulk](#) *bulk, const struct [timespec](#) *timeout, struct [gpiod_line_bulk](#) *event_bulk) [GPIO_API](#)
Wait for events on a set of lines.
- int [gpiod_line_event_read](#) (struct [gpiod_line](#) *line, struct [gpiod_line_event](#) *event) [GPIO_API](#)
Read next pending event from the GPIO line.
- int [gpiod_line_event_read_multiple](#) (struct [gpiod_line](#) *line, struct [gpiod_line_event](#) *events, unsigned int num_events) [GPIO_API](#)
Read up to a certain number of events from the GPIO line.
- int [gpiod_line_event_get_fd](#) (struct [gpiod_line](#) *line) [GPIO_API](#)
Get the event file descriptor.
- int [gpiod_line_event_read_fd](#) (int fd, struct [gpiod_line_event](#) *event) [GPIO_API](#)
Read the last GPIO event directly from a file descriptor.
- int [gpiod_line_event_read_fd_multiple](#) (int fd, struct [gpiod_line_event](#) *events, unsigned int num_events) [GPIO_API](#)
Read up to a certain number of events directly from a file descriptor.
- const char * [gpiod_version_string](#) (void) [GPIO_API](#)
Get the API version of the library as a human-readable string.

7.2 gpiod.hpp File Reference

```
#include <bitset>
#include <chrono>
#include <gpiod.h>
#include <memory>
#include <string>
#include <vector>
```

Include dependency graph for gpiod.hpp:



Data Structures

- class [gpiod::chip](#)
Represents a GPIO chip.
- struct [gpiod::line_request](#)
Stores the configuration for line requests.
- class [gpiod::line](#)

- Represents a single GPIO line.*
- struct [gpiod::line_event](#)
 - Describes a single GPIO line event.*
- class [gpiod::line_bulk](#)
 - Represents a set of GPIO lines.*
- class [gpiod::line_bulk::iterator](#)
 - Iterator for iterating over lines held by [line_bulk](#).*
- class [gpiod::line_iter](#)
 - Allows to iterate over all lines owned by a GPIO chip.*

Functions

- bool [gpiod::is_gpiochip_device](#) (const ::std::string &path) **GPIOD_API**
 - Check if the file pointed to by path is a GPIO chip character device.*
- **GPIOD_API** line_iter [gpiod::begin](#) (line_iter iter) noexcept
 - Support for range-based loops for line iterators.*
- **GPIOD_API** line_iter [gpiod::end](#) (const line_iter &iter) noexcept
 - Support for range-based loops for line iterators.*

Index

- ~chip
 - gpiod::chip, 63
- append
 - gpiod::line_bulk, 89
- begin
 - C++ bindings, 58
 - gpiod::line_bulk, 90
- bias
 - gpiod::line, 79
- C++ bindings, 58
 - begin, 58
 - end, 58
 - is_gpiochip_device, 59
- chip
 - gpiod::chip, 62, 63
- Common helper macros, 9
 - GPIO_BIT, 9
- consumer
 - gpiod::line, 79
 - gpiod::line_request, 103
 - gpiod_line_request_config, 70
- direction
 - gpiod::line, 79
- drive
 - gpiod::line, 80
- empty
 - gpiod::line_bulk, 90
- end
 - C++ bindings, 58
 - gpiod::line_bulk, 90
- event_get_fd
 - gpiod::line, 80
- event_read
 - gpiod::line, 80
- event_read_multiple
 - gpiod::line, 80
- event_type
 - gpiod::line_event, 97
 - gpiod_line_event, 69
- event_wait
 - gpiod::line, 81
 - gpiod::line_bulk, 90
- FLAG_ACTIVE_LOW
 - gpiod::line_request, 104
- FLAG_BIAS_DISABLED
 - gpiod::line_request, 104
- FLAG_BIAS_PULL_DOWN
 - gpiod::line_request, 104
- FLAG_BIAS_PULL_UP
 - gpiod::line_request, 104
- FLAG_OPEN_DRAIN
 - gpiod::line_request, 104
- FLAG_OPEN_SOURCE
 - gpiod::line_request, 105
- find_line
 - gpiod::chip, 63
- flags
 - gpiod::line_request, 105
 - gpiod_line_request_config, 71
- GPIO chip operations, 11
 - gpiod_chip_close, 11
 - gpiod_chip_find_line, 12
 - gpiod_chip_find_line_unique, 12
 - gpiod_chip_get_all_lines, 12
 - gpiod_chip_get_line, 13
 - gpiod_chip_get_lines, 13
 - gpiod_chip_label, 14
 - gpiod_chip_name, 14
 - gpiod_chip_num_lines, 14
 - gpiod_chip_open, 15
 - gpiod_is_gpiochip_device, 15
- GPIO line operations, 16
 - GPIO_BIT
 - Common helper macros, 9
- get
 - gpiod::line_bulk, 91
- get_all_lines
 - gpiod::chip, 64
- get_chip
 - gpiod::line, 81
- get_line
 - gpiod::chip, 64
- get_lines
 - gpiod::chip, 64
- get_value
 - gpiod::line, 81
- get_values
 - gpiod::line_bulk, 91
- gpiod.h, 107
- gpiod.hpp, 112
- gpiod::chip, 61
 - ~chip, 63
 - chip, 62, 63
 - find_line, 63

- get_all_lines, 64
- get_line, 64
- get_lines, 64
- label, 65
- name, 65
- num_lines, 65
- open, 65
- operator bool, 67
- operator!, 67
- operator!=, 67
- operator=, 67, 68
- operator==, 68
- gpiod::line, 75
 - bias, 79
 - consumer, 79
 - direction, 79
 - drive, 80
 - event_get_fd, 80
 - event_read, 80
 - event_read_multiple, 80
 - event_wait, 81
 - get_chip, 81
 - get_value, 81
 - is_active_low, 81
 - is_requested, 82
 - is_used, 82
 - line, 78, 79
 - name, 82
 - offset, 82
 - operator bool, 83
 - operator!, 83
 - operator!=, 83
 - operator=, 84
 - operator==, 84
 - request, 85
 - reset, 85
 - set_config, 85
 - set_direction_output, 86
 - set_flags, 86
 - set_value, 86
- gpiod::line_bulk, 86
 - append, 89
 - begin, 90
 - empty, 90
 - end, 90
 - event_wait, 90
 - get, 91
 - get_values, 91
 - line_bulk, 88, 89
 - operator bool, 91
 - operator!, 92
 - operator=, 92
 - operator[], 93
 - request, 93
 - set_config, 93
 - set_direction_output, 94
 - set_flags, 94
 - set_values, 94
 - size, 96
- gpiod::line_bulk::iterator, 71
 - iterator, 72, 73
 - operator!=, 73
 - operator*, 73
 - operator++, 73
 - operator->, 74
 - operator=, 74
 - operator==, 75
- gpiod::line_event, 96
 - event_type, 97
 - source, 97
 - timestamp, 98
- gpiod::line_iter, 98
 - line_iter, 99
 - operator!=, 100
 - operator*, 100
 - operator++, 100
 - operator->, 101
 - operator=, 101
 - operator==, 102
- gpiod::line_request, 102
 - consumer, 103
 - FLAG_ACTIVE_LOW, 104
 - FLAG_BIAS_DISABLED, 104
 - FLAG_BIAS_PULL_DOWN, 104
 - FLAG_BIAS_PULL_UP, 104
 - FLAG_OPEN_DRAIN, 104
 - FLAG_OPEN_SOURCE, 105
 - flags, 105
 - request_type, 105
- gpiod_chip_close
 - GPIO chip operations, 11
- gpiod_chip_find_line
 - GPIO chip operations, 12
- gpiod_chip_find_line_unique
 - GPIO chip operations, 12
- gpiod_chip_get_all_lines
 - GPIO chip operations, 12
- gpiod_chip_get_line
 - GPIO chip operations, 13
- gpiod_chip_get_lines
 - GPIO chip operations, 13
- gpiod_chip_label
 - GPIO chip operations, 14
- gpiod_chip_name
 - GPIO chip operations, 14
- gpiod_chip_num_lines
 - GPIO chip operations, 14
- gpiod_chip_open
 - GPIO chip operations, 15
- gpiod_is_gpiochip_device
 - GPIO chip operations, 15
- gpiod_line_bias
 - Line info, 24
- gpiod_line_bulk_add_line
 - Operating on multiple lines, 18
- gpiod_line_bulk_foreach_cb

- Operating on multiple lines, [18](#)
- `gpiod_line_bulk_foreach_line`
 - Operating on multiple lines, [19](#)
- `gpiod_line_bulk_free`
 - Operating on multiple lines, [19](#)
- `gpiod_line_bulk_get_line`
 - Operating on multiple lines, [19](#)
- `gpiod_line_bulk_new`
 - Operating on multiple lines, [20](#)
- `gpiod_line_bulk_num_lines`
 - Operating on multiple lines, [20](#)
- `gpiod_line_bulk_reset`
 - Operating on multiple lines, [20](#)
- `gpiod_line_consumer`
 - Line info, [24](#)
- `gpiod_line_direction`
 - Line info, [24](#)
- `gpiod_line_drive`
 - Line info, [25](#)
- `gpiod_line_event`, [69](#)
 - event_type, [69](#)
 - offset, [69](#)
 - ts, [70](#)
- `gpiod_line_event_get_fd`
 - Line events handling, [53](#)
- `gpiod_line_event_read`
 - Line events handling, [53](#)
- `gpiod_line_event_read_fd`
 - Line events handling, [54](#)
- `gpiod_line_event_read_fd_multiple`
 - Line events handling, [54](#)
- `gpiod_line_event_read_multiple`
 - Line events handling, [55](#)
- `gpiod_line_event_wait`
 - Line events handling, [55](#)
- `gpiod_line_event_wait_bulk`
 - Line events handling, [55](#)
- `gpiod_line_get_chip`
 - Line info, [25](#)
- `gpiod_line_get_value`
 - Reading & setting line values, [43](#)
- `gpiod_line_get_value_bulk`
 - Reading & setting line values, [44](#)
- `gpiod_line_is_active_low`
 - Line info, [26](#)
- `gpiod_line_is_free`
 - Line requests, [32](#)
- `gpiod_line_is_requested`
 - Line requests, [32](#)
- `gpiod_line_is_used`
 - Line info, [26](#)
- `gpiod_line_name`
 - Line info, [26](#)
- `gpiod_line_offset`
 - Line info, [27](#)
- `gpiod_line_release`
 - Line requests, [32](#)
- `gpiod_line_release_bulk`
 - Line requests, [33](#)
- `gpiod_line_request`
 - Line requests, [33](#)
- `gpiod_line_request_both_edges_events`
 - Line requests, [33](#)
- `gpiod_line_request_both_edges_events_flags`
 - Line requests, [34](#)
- `gpiod_line_request_bulk`
 - Line requests, [34](#)
- `gpiod_line_request_bulk_both_edges_events`
 - Line requests, [34](#)
- `gpiod_line_request_bulk_both_edges_events_flags`
 - Line requests, [35](#)
- `gpiod_line_request_bulk_falling_edge_events`
 - Line requests, [35](#)
- `gpiod_line_request_bulk_falling_edge_events_flags`
 - Line requests, [36](#)
- `gpiod_line_request_bulk_input`
 - Line requests, [36](#)
- `gpiod_line_request_bulk_input_flags`
 - Line requests, [37](#)
- `gpiod_line_request_bulk_output`
 - Line requests, [37](#)
- `gpiod_line_request_bulk_output_flags`
 - Line requests, [37](#)
- `gpiod_line_request_bulk_rising_edge_events`
 - Line requests, [38](#)
- `gpiod_line_request_bulk_rising_edge_events_flags`
 - Line requests, [38](#)
- `gpiod_line_request_config`, [70](#)
 - consumer, [70](#)
 - flags, [71](#)
 - request_type, [71](#)
- `gpiod_line_request_falling_edge_events`
 - Line requests, [39](#)
- `gpiod_line_request_falling_edge_events_flags`
 - Line requests, [39](#)
- `gpiod_line_request_input`
 - Line requests, [39](#)
- `gpiod_line_request_input_flags`
 - Line requests, [40](#)
- `gpiod_line_request_output`
 - Line requests, [40](#)
- `gpiod_line_request_output_flags`
 - Line requests, [41](#)
- `gpiod_line_request_rising_edge_events`
 - Line requests, [41](#)
- `gpiod_line_request_rising_edge_events_flags`
 - Line requests, [41](#)
- `gpiod_line_set_config`
 - Setting line configuration, [47](#)
- `gpiod_line_set_config_bulk`
 - Setting line configuration, [48](#)
- `gpiod_line_set_direction_input`
 - Setting line configuration, [48](#)
- `gpiod_line_set_direction_input_bulk`
 - Setting line configuration, [49](#)
- `gpiod_line_set_direction_output`

- Setting line configuration, 49
- gpiod_line_set_direction_output_bulk
 - Setting line configuration, 49
- gpiod_line_set_flags
 - Setting line configuration, 50
- gpiod_line_set_flags_bulk
 - Setting line configuration, 50
- gpiod_line_set_value
 - Reading & setting line values, 44
- gpiod_line_set_value_bulk
 - Reading & setting line values, 44
- gpiod_line_update
 - Line info, 27
- gpiod_version_string
 - Stuff that didn't fit anywhere else, 57
- is_active_low
 - gpiod::line, 81
- is_gpiochip_device
 - C++ bindings, 59
- is_requested
 - gpiod::line, 82
- is_used
 - gpiod::line, 82
- iterator
 - gpiod::line_bulk::iterator, 72, 73
- label
 - gpiod::chip, 65
- line
 - gpiod::line, 78, 79
- Line events handling, 52
 - gpiod_line_event_get_fd, 53
 - gpiod_line_event_read, 53
 - gpiod_line_event_read_fd, 54
 - gpiod_line_event_read_fd_multiple, 54
 - gpiod_line_event_read_multiple, 55
 - gpiod_line_event_wait, 55
 - gpiod_line_event_wait_bulk, 55
- Line info, 22
 - gpiod_line_bias, 24
 - gpiod_line_consumer, 24
 - gpiod_line_direction, 24
 - gpiod_line_drive, 25
 - gpiod_line_get_chip, 25
 - gpiod_line_is_active_low, 26
 - gpiod_line_is_used, 26
 - gpiod_line_name, 26
 - gpiod_line_offset, 27
 - gpiod_line_update, 27
- Line requests, 29
 - gpiod_line_is_free, 32
 - gpiod_line_is_requested, 32
 - gpiod_line_release, 32
 - gpiod_line_release_bulk, 33
 - gpiod_line_request, 33
 - gpiod_line_request_both_edges_events, 33
 - gpiod_line_request_both_edges_events_flags, 34
 - gpiod_line_request_bulk, 34
 - gpiod_line_request_bulk_both_edges_events, 34
 - gpiod_line_request_bulk_both_edges_events_↔
 - flags, 35
 - gpiod_line_request_bulk_falling_edge_events, 35
 - gpiod_line_request_bulk_falling_edge_events_↔
 - flags, 36
 - gpiod_line_request_bulk_input, 36
 - gpiod_line_request_bulk_input_flags, 37
 - gpiod_line_request_bulk_output, 37
 - gpiod_line_request_bulk_output_flags, 37
 - gpiod_line_request_bulk_rising_edge_events, 38
 - gpiod_line_request_bulk_rising_edge_events_↔
 - flags, 38
 - gpiod_line_request_falling_edge_events, 39
 - gpiod_line_request_falling_edge_events_flags, 39
 - gpiod_line_request_input, 39
 - gpiod_line_request_input_flags, 40
 - gpiod_line_request_output, 40
 - gpiod_line_request_output_flags, 41
 - gpiod_line_request_rising_edge_events, 41
 - gpiod_line_request_rising_edge_events_flags, 41
- line_bulk
 - gpiod::line_bulk, 88, 89
- line_iter
 - gpiod::line_iter, 99
- name
 - gpiod::chip, 65
 - gpiod::line, 82
- num_lines
 - gpiod::chip, 65
- offset
 - gpiod::line, 82
 - gpiod_line_event, 69
- open
 - gpiod::chip, 65
- Operating on multiple lines, 17
 - gpiod_line_bulk_add_line, 18
 - gpiod_line_bulk_foreach_cb, 18
 - gpiod_line_bulk_foreach_line, 19
 - gpiod_line_bulk_free, 19
 - gpiod_line_bulk_get_line, 19
 - gpiod_line_bulk_new, 20
 - gpiod_line_bulk_num_lines, 20
 - gpiod_line_bulk_reset, 20
- operator bool
 - gpiod::chip, 67
 - gpiod::line, 83
 - gpiod::line_bulk, 91
- operator!
 - gpiod::chip, 67
 - gpiod::line, 83
 - gpiod::line_bulk, 92
- operator!=
 - gpiod::chip, 67
 - gpiod::line, 83
 - gpiod::line_bulk::iterator, 73
 - gpiod::line_iter, 100

- operator*
 - gpiod::line_bulk::iterator, 73
 - gpiod::line_iter, 100
- operator++
 - gpiod::line_bulk::iterator, 73
 - gpiod::line_iter, 100
- operator->
 - gpiod::line_bulk::iterator, 74
 - gpiod::line_iter, 101
- operator=
 - gpiod::chip, 67, 68
 - gpiod::line, 84
 - gpiod::line_bulk, 92
 - gpiod::line_bulk::iterator, 74
 - gpiod::line_iter, 101
- operator==
 - gpiod::chip, 68
 - gpiod::line, 84
 - gpiod::line_bulk::iterator, 75
 - gpiod::line_iter, 102
- operator[]
 - gpiod::line_bulk, 93
- Reading & setting line values, 43
 - gpiod_line_get_value, 43
 - gpiod_line_get_value_bulk, 44
 - gpiod_line_set_value, 44
 - gpiod_line_set_value_bulk, 44
- request
 - gpiod::line, 85
 - gpiod::line_bulk, 93
- request_type
 - gpiod::line_request, 105
 - gpiod_line_request_config, 71
- reset
 - gpiod::line, 85
- set_config
 - gpiod::line, 85
 - gpiod::line_bulk, 93
- set_direction_output
 - gpiod::line, 86
 - gpiod::line_bulk, 94
- set_flags
 - gpiod::line, 86
 - gpiod::line_bulk, 94
- set_value
 - gpiod::line, 86
- set_values
 - gpiod::line_bulk, 94
- Setting line configuration, 47
 - gpiod_line_set_config, 47
 - gpiod_line_set_config_bulk, 48
 - gpiod_line_set_direction_input, 48
 - gpiod_line_set_direction_input_bulk, 49
 - gpiod_line_set_direction_output, 49
 - gpiod_line_set_direction_output_bulk, 49
 - gpiod_line_set_flags, 50
 - gpiod_line_set_flags_bulk, 50
- size
 - gpiod::line_bulk, 96
- source
 - gpiod::line_event, 97
- Stuff that didn't fit anywhere else, 57
 - gpiod_version_string, 57
- timestamp
 - gpiod::line_event, 98
- ts
 - gpiod_line_event, 70