

# My Project

Generated by Doxygen 1.9.4



<b>1 Module Index</b>	<b>1</b>
1.1 Modules	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Module Documentation</b>	<b>7</b>
4.1 C++ bindings	7
4.1.1 Detailed Description	8
4.1.2 Function Documentation	8
4.1.2.1 begin() [1/2]	8
4.1.2.2 begin() [2/2]	8
4.1.2.3 end() [1/2]	8
4.1.2.4 end() [2/2]	9
4.1.2.5 find_line()	9
4.1.2.6 make_chip_iter()	9
<b>5 Class Documentation</b>	<b>11</b>
5.1 gpiod::chip Class Reference	11
5.1.1 Detailed Description	12
5.1.2 Member Enumeration Documentation	12
5.1.2.1 anonymous enum	12
5.1.3 Constructor & Destructor Documentation	13
5.1.3.1 chip() [1/3]	13
5.1.3.2 chip() [2/3]	13
5.1.3.3 chip() [3/3]	13
5.1.4 Member Function Documentation	13
5.1.4.1 find_line()	14
5.1.4.2 find_lines()	14
5.1.4.3 get_all_lines()	14
5.1.4.4 get_line()	14
5.1.4.5 get_lines()	15
5.1.4.6 label()	15
5.1.4.7 name()	15
5.1.4.8 num_lines()	16
5.1.4.9 open()	16
5.1.4.10 operator bool()	16
5.1.4.11 operator"!()	16
5.1.4.12 operator"!=(	17
5.1.4.13 operator=(	17
5.1.4.14 operator=(	17

5.1.4.15 operator==( )	18
5.2 gpiod::chip_iter Class Reference	18
5.2.1 Detailed Description	19
5.2.2 Constructor & Destructor Documentation	19
5.2.2.1 chip_iter() [1/2]	19
5.2.2.2 chip_iter() [2/2]	19
5.2.3 Member Function Documentation	19
5.2.3.1 operator!=( )	19
5.2.3.2 operator*( )	20
5.2.3.3 operator++( )	20
5.2.3.4 operator->( )	20
5.2.3.5 operator=( ) [1/2]	20
5.2.3.6 operator=( ) [2/2]	21
5.2.3.7 operator==( )	21
5.2.4 Friends And Related Function Documentation	21
5.2.4.1 make_chip_iter	22
5.3 gpiod::line_bulk::iterator Class Reference	22
5.3.1 Detailed Description	23
5.3.2 Constructor & Destructor Documentation	23
5.3.2.1 iterator() [1/2]	23
5.3.2.2 iterator() [2/2]	23
5.3.3 Member Function Documentation	23
5.3.3.1 operator!=( )	23
5.3.3.2 operator*( )	24
5.3.3.3 operator++( )	24
5.3.3.4 operator->( )	24
5.3.3.5 operator=( ) [1/2]	24
5.3.3.6 operator=( ) [2/2]	25
5.3.3.7 operator==( )	25
5.4 gpiod::line Class Reference	26
5.4.1 Detailed Description	27
5.4.2 Member Enumeration Documentation	27
5.4.2.1 anonymous enum	28
5.4.2.2 anonymous enum	29
5.4.2.3 anonymous enum	29
5.4.3 Constructor & Destructor Documentation	29
5.4.3.1 line() [1/2]	29
5.4.3.2 line() [2/2]	30
5.4.4 Member Function Documentation	30
5.4.4.1 active_state()	30
5.4.4.2 bias()	30
5.4.4.3 consumer()	31

5.4.4.4 direction()	31
5.4.4.5 event_get_fd()	31
5.4.4.6 event_read()	31
5.4.4.7 event_read_multiple()	32
5.4.4.8 event_wait()	32
5.4.4.9 get_chip()	32
5.4.4.10 get_value()	32
5.4.4.11 is_open_drain()	33
5.4.4.12 is_open_source()	33
5.4.4.13 is_requested()	33
5.4.4.14 is_used()	33
5.4.4.15 name()	34
5.4.4.16 offset()	34
5.4.4.17 operator bool()	34
5.4.4.18 operator"!()	34
5.4.4.19 operator"!=(())	34
5.4.4.20 operator=() [1/2]	35
5.4.4.21 operator=() [2/2]	35
5.4.4.22 operator==(())	35
5.4.4.23 request()	37
5.4.4.24 reset()	37
5.4.4.25 set_config()	37
5.4.4.26 set_direction_output()	38
5.4.4.27 set_flags()	38
5.4.4.28 set_value()	38
5.5 gpiod::line_bulk Class Reference	38
5.5.1 Detailed Description	40
5.5.2 Constructor & Destructor Documentation	40
5.5.2.1 line_bulk() [1/3]	40
5.5.2.2 line_bulk() [2/3]	41
5.5.2.3 line_bulk() [3/3]	41
5.5.3 Member Function Documentation	41
5.5.3.1 append()	41
5.5.3.2 begin()	42
5.5.3.3 empty()	42
5.5.3.4 end()	42
5.5.3.5 event_wait()	42
5.5.3.6 get()	43
5.5.3.7 get_values()	43
5.5.3.8 operator bool()	43
5.5.3.9 operator"!()	44
5.5.3.10 operator=() [1/2]	44

5.5.3.11 operator=() [2/2]	44
5.5.3.12 operator[]()	45
5.5.3.13 request()	45
5.5.3.14 set_config()	45
5.5.3.15 set_direction_output()	46
5.5.3.16 set_flags()	46
5.5.3.17 set_values()	46
5.5.3.18 size()	46
5.6 gpiod::line_event Struct Reference	47
5.6.1 Detailed Description	47
5.6.2 Member Enumeration Documentation	47
5.6.2.1 anonymous enum	47
5.6.3 Member Data Documentation	48
5.6.3.1 event_type	48
5.6.3.2 source	48
5.6.3.3 timestamp	48
5.7 gpiod::line_iter Class Reference	48
5.7.1 Detailed Description	49
5.7.2 Constructor & Destructor Documentation	49
5.7.2.1 line_iter() [1/3]	49
5.7.2.2 line_iter() [2/3]	50
5.7.2.3 line_iter() [3/3]	50
5.7.3 Member Function Documentation	50
5.7.3.1 operator!=(())	50
5.7.3.2 operator*()	51
5.7.3.3 operator++()	51
5.7.3.4 operator->()	51
5.7.3.5 operator=() [1/2]	51
5.7.3.6 operator=() [2/2]	52
5.7.3.7 operator==(())	52
5.8 gpiod::line_request Struct Reference	52
5.8.1 Detailed Description	53
5.8.2 Member Enumeration Documentation	53
5.8.2.1 anonymous enum	53
5.8.3 Member Data Documentation	54
5.8.3.1 consumer	54
5.8.3.2 FLAG_ACTIVE_LOW	54
5.8.3.3 FLAG_BIAS_DISABLE	54
5.8.3.4 FLAG_BIAS_PULL_DOWN	54
5.8.3.5 FLAG_BIAS_PULL_UP	54
5.8.3.6 FLAG_OPEN_DRAIN	54
5.8.3.7 FLAG_OPEN_SOURCE	54

---

5.8.3.8 flags . . . . .	55
5.8.3.9 request_type . . . . .	55
<b>6 File Documentation</b>	<b>57</b>
6.1 gpiod.hpp File Reference . . . . .	57
6.2 gpiod.hpp . . . . .	58
<b>Index</b>	<b>65</b>





# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

C++ bindings . . . . .	<a href="#">7</a>
------------------------	-------------------



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">gpod::chip</a>	Represents a GPIO chip . . . . .	11
<a href="#">gpod::chip_iter</a>	Allows to iterate over all GPIO chips present on the system . . . . .	18
<a href="#">gpod::line_bulk::iterator</a>	Iterator for iterating over lines held by <a href="#">line_bulk</a> . . . . .	22
<a href="#">gpod::line</a>	Represents a single GPIO line . . . . .	26
<a href="#">gpod::line_bulk</a>	Represents a set of GPIO lines . . . . .	38
<a href="#">gpod::line_event</a>	Describes a single GPIO line event . . . . .	47
<a href="#">gpod::line_iter</a>	Allows to iterate over all lines owned by a GPIO chip . . . . .	48
<a href="#">gpod::line_request</a>	Stores the configuration for line requests . . . . .	52



## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">gpiod.hpp</a> . . . . .	57
-------------------------------------	----



## Chapter 4

# Module Documentation

### 4.1 C++ bindings

#### Classes

- 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::chip\\_iter](#)  
*Allows to iterate over all GPIO chips present on the system.*
- class [gpiod::line\\_iter](#)  
*Allows to iterate over all lines owned by a GPIO chip.*

#### Functions

- GPIOD\_API [line gpiod::find\\_line](#) (const ::std::string &name)  
*Find a GPIO line by name. Search all GPIO chips present on the system.*
- GPIOD\_API [chip\\_iter gpiod::make\\_chip\\_iter](#) (void)  
*Create a new [chip\\_iter](#).*
- GPIOD\_API [chip\\_iter gpiod::begin](#) ([chip\\_iter](#) iter) noexcept  
*Support for range-based loops for chip iterators.*
- GPIOD\_API [chip\\_iter gpiod::end](#) (const [chip\\_iter](#) &iter) noexcept  
*Support for range-based loops for chip iterators.*
- 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.*

### 4.1.1 Detailed Description

### 4.1.2 Function Documentation

#### 4.1.2.1 `begin()` [1/2]

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

Support for range-based loops for chip iterators.

##### Parameters

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

##### Returns

Iterator unchanged.

#### 4.1.2.2 `begin()` [2/2]

```
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.

#### 4.1.2.3 `end()` [1/2]

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

Support for range-based loops for chip iterators.



**Parameters**

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

**Returns**

New end iterator.

**4.1.2.4 end() [2/2]**

```
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.

**4.1.2.5 find\_line()**

```
GPIOD_API line gpiod::find_line (
    const ::std::string & name )
```

Find a GPIO line by name. Search all GPIO chips present on the system.

**Parameters**

<i>name</i>	Name of the line.
-------------	-------------------

**Returns**

Returns a line object - empty if the line was not found.

**4.1.2.6 make\_chip\_iter()**

```
GPIOD_API chip_iter gpiod::make_chip_iter (
    void )
```

Create a new [chip\\_iter](#).

**Returns**

New chip iterator object pointing to the first GPIO chip on the system.

**Note**

This function is needed as we already use the default constructor of [gpiod::chip\\_iter](#) as the return value of `gpiod::end`.

## Chapter 5

# Class Documentation

### 5.1 `gpio::chip` Class Reference

Represents a GPIO chip.

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

#### Public Types

- enum : int {  
    [OPEN\\_LOOKUP](#) = 1 , [OPEN\\_BY\\_PATH](#) , [OPEN\\_BY\\_NAME](#) , [OPEN\\_BY\\_LABEL](#) ,  
    [OPEN\\_BY\\_NUMBER](#) }  
    *Affect the way in which [chip::chip](#) and [chip::open](#) will try to open a GPIO chip character device.*

#### Public Member Functions

- `GPIO_API chip` (void)=default  
    *Default constructor. Creates an empty GPIO chip object.*
- `GPIO_API chip` (const ::std::string &device, int how=[OPEN\\_LOOKUP](#))  
    *Constructor. Opens the chip using [chip::open](#).*
- `GPIO_API chip` (const [chip](#) &other)=default  
    *Copy constructor. References the object held by other.*
- `GPIO_API chip` ([chip](#) &&other)=default  
    *Move constructor. References the object held by other.*
- `GPIO_API chip & operator=` (const [chip](#) &other)=default  
    *Assignment operator. References the object held by other.*
- `GPIO_API chip & operator=` ([chip](#) &&other)=default  
    *Move assignment operator. References the object held by other.*
- `GPIO_API ~chip` (void)=default  
    *Destructor. Unreferences the internal chip object.*
- `GPIO_API void open` (const ::std::string &device, int how=[OPEN\\_LOOKUP](#))  
    *Open a GPIO chip.*
- `GPIO_API void reset` (void) noexcept  
    *Reset the internal smart pointer owned by this object.*
- `GPIO_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.*
- `GPIOD_API line find_line` (const ::std::string &name) const  
*Get the line exposed by this chip by name.*
- `GPIOD_API line_bulk get_lines` (const ::std::vector< unsigned int > &offsets) const  
*Get a set of lines exposed by this chip at given offsets.*
- `GPIOD_API line_bulk get_all_lines` (void) const  
*Get all lines exposed by this chip.*
- `GPIOD_API line_bulk find_lines` (const ::std::vector<::std::string > &names) const  
*Get a set of lines exposed by this chip by their names.*
- `GPIOD_API bool operator==` (const `chip` &rhs) const noexcept  
*Equality operator.*
- `GPIOD_API bool operator!=` (const `chip` &rhs) const noexcept  
*Inequality operator.*
- `GPIOD_API operator bool` (void) const noexcept  
*Check if this object holds a reference to a GPIO chip.*
- `GPIOD_API bool operator!` (void) const noexcept  
*Check if this object doesn't hold a reference to a GPIO chip.*

### 5.1.1 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.

### 5.1.2 Member Enumeration Documentation

#### 5.1.2.1 anonymous enum

```
anonymous enum : int
```

Affect the way in which `chip::chip` and `chip::open` will try to open a GPIO chip character device.

Enumerator

OPEN_LOOKUP	Open based on the best guess what the supplied string is.
OPEN_BY_PATH	Assume the string is a path to the GPIO chardev.
OPEN_BY_NAME	Assume the string is the name of the chip
OPEN_BY_LABEL	Assume the string is the label of the GPIO chip.
OPEN_BY_NUMBER	Assume the string is the number of the GPIO chip.

## 5.1.3 Constructor & Destructor Documentation

### 5.1.3.1 chip() [1/3]

```
GPIOD_API gpiod::chip::chip (  
    const ::std::string & device,  
    int how = OPEN_LOOKUP )
```

Constructor. Opens the chip using [chip::open](#).

#### Parameters

<i>device</i>	String describing the GPIO chip.
<i>how</i>	Indicates how the chip should be opened.

### 5.1.3.2 chip() [2/3]

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

Copy constructor. References the object held by other.

#### Parameters

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

### 5.1.3.3 chip() [3/3]

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

Move constructor. References the object held by other.

#### Parameters

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

## 5.1.4 Member Function Documentation

#### 5.1.4.1 find\_line()

```
GPIOD_API line gpiod::chip::find_line (
    const ::std::string & name ) const
```

Get the line exposed by this chip by name.

##### Parameters

<i>name</i>	Line name.
-------------	------------

##### Returns

Line object.

#### 5.1.4.2 find\_lines()

```
GPIOD_API line_bulk gpiod::chip::find_lines (
    const ::std::vector<::std::string > & names ) const
```

Get a set of lines exposed by this chip by their names.

##### Parameters

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

##### Returns

Set of lines held by a [line\\_bulk](#) object.

#### 5.1.4.3 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.

#### 5.1.4.4 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.

### 5.1.4.5 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.

### 5.1.4.6 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.

### 5.1.4.7 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.

#### 5.1.4.8 num\_lines()

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

Return the number of lines exposed by this chip.

##### Returns

Number of lines.

#### 5.1.4.9 open()

```
GPIO_API void gpiod::chip::open (
    const ::std::string & device,
    int how = OPEN_LOOKUP )
```

Open a GPIO chip.

##### Parameters

<i>device</i>	String describing the GPIO chip.
<i>how</i>	Indicates how the chip should be opened.

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

#### 5.1.4.10 operator bool()

```
GPIO_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.

#### 5.1.4.11 operator"!()

```
GPIO_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.



#### 5.1.4.12 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.

#### 5.1.4.13 operator=(()) [1/2]

```
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.

#### 5.1.4.14 operator=(()) [2/2]

```
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.

#### 5.1.4.15 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](#)

## 5.2 gpiod::chip\_iter Class Reference

Allows to iterate over all GPIO chips present on the system.

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

### Public Member Functions

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

## Friends

- [chip\\_iter make\\_chip\\_iter](#) (void)  
*Create a new [chip\\_iter](#).*

### 5.2.1 Detailed Description

Allows to iterate over all GPIO chips present on the system.

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 [chip\\_iter\(\)](#) [1/2]

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

Copy constructor.

##### Parameters

<i>other</i>	Other <a href="#">chip_iter</a> .
--------------	-----------------------------------

#### 5.2.2.2 [chip\\_iter\(\)](#) [2/2]

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

Move constructor.

##### Parameters

<i>other</i>	Other <a href="#">chip_iter</a> .
--------------	-----------------------------------

### 5.2.3 Member Function Documentation

#### 5.2.3.1 [operator"!="\(\)](#)

```
GPIOD_API bool gpiod::chip_iter::operator!= (  
    const chip\_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 [chip\\_iter](#), false otherwise.

### 5.2.3.2 operator\*()

```
GPIOD_API const chip & gpiod::chip_iter::operator* (
    void ) const
```

Dereference current element.

#### Returns

Current GPIO chip by reference.

### 5.2.3.3 operator++()

```
GPIOD_API chip\_iter & gpiod::chip_iter::operator++ (
    void )
```

Advance the iterator by one element.

#### Returns

Reference to this iterator.

### 5.2.3.4 operator->()

```
GPIOD_API const chip * gpiod::chip_iter::operator-> (
    void ) const
```

Member access operator.

#### Returns

Current GPIO chip by pointer.

### 5.2.3.5 operator=() [1/2]

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

Move assignment operator.

## Parameters

<i>other</i>	Other <a href="#">chip_iter</a> .
--------------	-----------------------------------

## Returns

Reference to this iterator.

### 5.2.3.6 operator=() [2/2]

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

Assignment operator.

## Parameters

<i>other</i>	Other <a href="#">chip_iter</a> .
--------------	-----------------------------------

## Returns

Reference to this iterator.

### 5.2.3.7 operator==()

```
GPIOD_API bool gpiod::chip_iter::operator== (
    const chip\_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 [chip\\_iter](#), false otherwise.

## 5.2.4 Friends And Related Function Documentation

### 5.2.4.1 make\_chip\_iter

```
chip_iter make_chip_iter (
    void ) [friend]
```

Create a new [chip\\_iter](#).

#### Returns

New chip iterator object pointing to the first GPIO chip on the system.

#### Note

This function is needed as we already use the default constructor of [gpiod::chip\\_iter](#) as the return value of [gpiod::end](#).

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

- [gpiod.hpp](#)

## 5.3 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. Builds an empty iterator object.*
- **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.*

### 5.3.1 Detailed Description

Iterator for iterating over lines held by [line\\_bulk](#).

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 iterator() [1/2]

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

Copy constructor.

##### Parameters

<i>other</i>	Other <a href="#">line_bulk</a> iterator.
--------------	---

#### 5.3.2.2 iterator() [2/2]

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

Move constructor.

##### Parameters

<i>other</i>	Other <a href="#">line_bulk</a> iterator.
--------------	---

### 5.3.3 Member Function Documentation

#### 5.3.3.1 operator!=(())

```
GPIOD_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.

**5.3.3.2 operator\*()**

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

Dereference current element.

**Returns**

Current GPIO line by reference.

**5.3.3.3 operator++()**

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

Advance the iterator by one element.

**Returns**

Reference to this iterator.

**5.3.3.4 operator->()**

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

Member access operator.

**Returns**

Current GPIO line by pointer.

**5.3.3.5 operator=() [1/2]**

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

Assignment operator.



## Parameters

<i>other</i>	Other <a href="#">line_bulk</a> iterator.
--------------	---

## Returns

Reference to this iterator.

### 5.3.3.6 operator=() [2/2]

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

Move assignment operator.

## Parameters

<i>other</i>	Other <a href="#">line_bulk</a> iterator.
--------------	---

## Returns

Reference to this iterator.

### 5.3.3.7 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](#)

## 5.4 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 { [ACTIVE\\_LOW](#) = 1 , [ACTIVE\\_HIGH](#) }  
*Possible active state settings.*
- enum : int { [BIAS\\_AS\\_IS](#) = 1 , [BIAS\\_DISABLE](#) , [BIAS\\_PULL\\_UP](#) , [BIAS\\_PULL\\_DOWN](#) }  
*Possible bias settings.*

### Public Member Functions

- **GPIOD\_API line** (void)  
*Default constructor. Creates an empty line object.*
- **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 int active\_state** (void) const  
*Get current active state of this line.*
- **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 bool is\_open\_drain** (void) const  
*Check if this line represents an open-drain GPIO.*
- **GPIOD\_API bool is\_open\_source** (void) const  
*Check if this line represents an open-source GPIO.*
- **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 reference to 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.*

### 5.4.1 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.

### 5.4.2 Member Enumeration Documentation

#### 5.4.2.1 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.

## 5.4.2.2 anonymous enum

```
anonymous enum : int
```

Possible active state settings.

## Enumerator

ACTIVE_LOW	Line's active state is low.
ACTIVE_HIGH	Line's active state is high.

## 5.4.2.3 anonymous enum

```
anonymous enum : int
```

Possible bias settings.

## Enumerator

BIAS_AS_IS	Line's bias state is unknown.
BIAS_DISABLE	Line's internal bias is disabled.
BIAS_PULL_UP	Line's internal pull-up bias is enabled.
BIAS_PULL_DOWN	Line's internal pull-down bias is enabled.

## 5.4.3 Constructor &amp; Destructor Documentation

## 5.4.3.1 line() [1/2]

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

Copy constructor.

**Parameters**

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

**5.4.3.2 line() [2/2]**

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

Move constructor.

**Parameters**

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

**5.4.4 Member Function Documentation****5.4.4.1 active\_state()**

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

Get current active state of this line.

**Returns**

Current active state setting.

**5.4.4.2 bias()**

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

Get current bias of this line.

**Returns**

Current bias setting.

#### 5.4.4.3 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.

#### 5.4.4.4 direction()

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

Get current direction of this line.

##### Returns

Current direction setting.

#### 5.4.4.5 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.

#### 5.4.4.6 event\_read()

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

Read a line event.

##### Returns

Line event object.

#### 5.4.4.7 event\_read\_multiple()

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

Read multiple line events.

##### Returns

Vector of line event objects.

#### 5.4.4.8 event\_wait()

```
GPIOD_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.

#### 5.4.4.9 get\_chip()

```
GPIOD_API const chip & gpiod::line::get_chip (
    void ) const
```

Get the reference to the parent chip.

##### Returns

Reference to the parent chip object.

#### 5.4.4.10 get\_value()

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

Read the line value.

##### Returns

Current value (0 or 1).



#### 5.4.4.11 is\_open\_drain()

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

Check if this line represents an open-drain GPIO.

##### Returns

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

#### 5.4.4.12 is\_open\_source()

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

Check if this line represents an open-source GPIO.

##### Returns

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

#### 5.4.4.13 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.

#### 5.4.4.14 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.

#### 5.4.4.15 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.

#### 5.4.4.16 offset()

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

Get the offset of this line.

##### Returns

Offset of this line.

#### 5.4.4.17 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.

#### 5.4.4.18 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.

#### 5.4.4.19 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.

**5.4.4.20 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.

**5.4.4.21 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.

**5.4.4.22 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.

#### 5.4.4.23 request()

```
GPIOD_API void gpiod::line::request (
    const line\_request & config,
    int default_val = 0 ) const
```

Request this line.

## Parameters

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

#### 5.4.4.24 reset()

```
GPIOD_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.

#### 5.4.4.25 set\_config()

```
GPIOD_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.

#### 5.4.4.26 set\_direction\_output()

```
GPIOD_API void gpiod::line::set_direction_output (
    int value = 0 ) const
```

Change the direction this lines to output.

##### Parameters

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

#### 5.4.4.27 set\_flags()

```
GPIOD_API void gpiod::line::set_flags (
    ::std::bitset< 32 > flags ) const
```

Set configuration flags of this line.

##### Parameters

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

#### 5.4.4.28 set\_value()

```
GPIOD_API void gpiod::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:

- [gpiod.hpp](#)

## 5.5 gpiod::line\_bulk Class Reference

Represents a set of GPIO lines.

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

## Classes

- class [iterator](#)

*Iterator for iterating over lines held by [line\\_bulk](#).*

## Public Member Functions

- `GPIOD_API line_bulk (void)=default`  
*Default constructor. Creates an empty [line\\_bulk](#) object.*
- `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 offset)`  
*Get the line at given offset.*
- `GPIOD_API line & operator[] (unsigned int offset)`  
*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.*

### 5.5.1 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.

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 `line_bulk()` [1/3]

```
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 <code>gpiod::line</code> objects.
--------------	---

#### Note

All lines must be owned by the same GPIO chip.



### 5.5.2.2 line\_bulk() [2/3]

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

Copy constructor.

#### Parameters

<i>other</i>	Other <a href="#">line_bulk</a> object.
--------------	---

### 5.5.2.3 line\_bulk() [3/3]

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

Move constructor.

#### Parameters

<i>other</i>	Other <a href="#">line_bulk</a> object.
--------------	---

## 5.5.3 Member Function Documentation

### 5.5.3.1 append()

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

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

#### Parameters

<i>new_line</i>	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.

### 5.5.3.2 begin()

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

Returns an iterator to the first line.

#### Returns

A [line\\_bulk](#) iterator.

### 5.5.3.3 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.

### 5.5.3.4 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.

### 5.5.3.5 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 <a href="#">line_bulk</a> .
----------------	---

**Returns**

Returns a [line\\_bulk](#) object containing lines on which events occurred.

**5.5.3.6 get()**

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

Get the line at given offset.

**Parameters**

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

**Returns**

Reference to the line object.

**5.5.3.7 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.

**5.5.3.8 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.

#### 5.5.3.9 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.

#### 5.5.3.10 operator=() [1/2]

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

Assignment operator.

##### Parameters

<i>other</i>	Other <a href="#">line_bulk</a> object.
--------------	---

##### Returns

Reference to this object.

#### 5.5.3.11 operator=() [2/2]

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

Move assignment operator.

##### Parameters

<i>other</i>	Other <a href="#">line_bulk</a> object.
--------------	---

##### Returns

Reference to this object.

### 5.5.3.12 operator[]()

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

Get the line at given offset without bounds checking.

#### Parameters

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

#### Returns

Reference to the line object.

#### Note

No bounds checking is performed.

### 5.5.3.13 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 <a href="#">gpiod::line_request</a> ).
<i>default_vals</i>	Vector of default values. Only relevant for output direction requests.

### 5.5.3.14 set\_config()

```
GPIOD_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 <a href="#">line_bulk</a> . Only relevant for output direction requests.

#### 5.5.3.15 set\_direction\_output()

```
GPIOD_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 <a href="#">line_bulk</a> .
---------------	--

#### 5.5.3.16 set\_flags()

```
GPIOD_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.
--------------	--------------------

#### 5.5.3.17 set\_values()

```
GPIOD_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 <a href="#">line_bulk</a> .
---------------	--

#### 5.5.3.18 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:

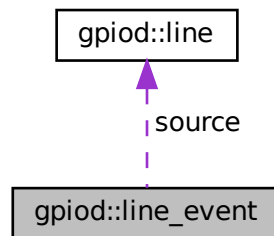
- [gpiod.hpp](#)

## 5.6 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.

**Public Attributes**

- ::std::chrono::nanoseconds [timestamp](#)
- int [event\\_type](#)
- [line](#) source

### 5.6.1 Detailed Description

Describes a single GPIO line event.

### 5.6.2 Member Enumeration Documentation

#### 5.6.2.1 anonymous enum

```
anonymous enum : int
```

Possible event types.

## Enumerator

RISING_EDGE	Rising edge event.
FALLING_EDGE	Falling edge event.

## 5.6.3 Member Data Documentation

### 5.6.3.1 event\_type

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

Type of the event that occurred.

### 5.6.3.2 source

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

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

### 5.6.3.3 timestamp

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

Best estimate of time of event occurrence in nanoseconds.

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

- [gpiod.hpp](#)

## 5.7 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. Creates the end iterator.*
- `GPIOD_API line_iter (const chip &owner)`  
*Constructor. Creates the begin iterator.*
- `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.*

### 5.7.1 Detailed Description

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

### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 line\_iter() [1/3]

```
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.
--------------	---

### 5.7.2.2 `line_iter()` [2/3]

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

Copy constructor.

#### Parameters

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

### 5.7.2.3 `line_iter()` [3/3]

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

Move constructor.

#### Parameters

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

## 5.7.3 Member Function Documentation

### 5.7.3.1 `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.

### 5.7.3.2 operator\*()

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

Dereference current element.

#### Returns

Current GPIO line by reference.

### 5.7.3.3 operator++()

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

Advance the iterator by one element.

#### Returns

Reference to this iterator.

### 5.7.3.4 operator->()

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

Member access operator.

#### Returns

Current GPIO line by pointer.

### 5.7.3.5 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](#).

**5.7.3.6 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](#).

**5.7.3.7 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](#)

**5.8 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.*

## Public Attributes

- ::std::string [consumer](#)
- int [request\\_type](#)
- ::std::bitset< 32 > [flags](#)

## Static Public Attributes

- static GPIOD\_API const ::std::bitset< 32 > [FLAG\\_ACTIVE\\_LOW](#)
- static GPIOD\_API const ::std::bitset< 32 > [FLAG\\_OPEN\\_SOURCE](#)
- static GPIOD\_API const ::std::bitset< 32 > [FLAG\\_OPEN\\_DRAIN](#)
- static GPIOD\_API const ::std::bitset< 32 > [FLAG\\_BIAS\\_DISABLE](#)
- static GPIOD\_API const ::std::bitset< 32 > [FLAG\\_BIAS\\_PULL\\_DOWN](#)
- static GPIOD\_API const ::std::bitset< 32 > [FLAG\\_BIAS\\_PULL\\_UP](#)

### 5.8.1 Detailed Description

Stores the configuration for line requests.

### 5.8.2 Member Enumeration Documentation

#### 5.8.2.1 anonymous enum

anonymous enum : int

Request types.

##### Enumerator

<a href="#">DIRECTION_AS_IS</a>	Request for values, don't change the direction.
<a href="#">DIRECTION_INPUT</a>	Request for reading line values.
<a href="#">DIRECTION_OUTPUT</a>	Request for driving the GPIO lines.
<a href="#">EVENT_FALLING_EDGE</a>	Listen for falling edge events.
<a href="#">EVENT_RISING_EDGE</a>	Listen for rising edge events.
<a href="#">EVENT_BOTH_EDGES</a>	Listen for all types of events.

## 5.8.3 Member Data Documentation

### 5.8.3.1 consumer

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

Consumer name to pass to the request.

### 5.8.3.2 FLAG\_ACTIVE\_LOW

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

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

### 5.8.3.3 FLAG\_BIAS\_DISABLE

```
GPIO_API const ::std::bitset<32> gpiod::line_request::FLAG_BIAS_DISABLE [static]
```

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

### 5.8.3.4 FLAG\_BIAS\_PULL\_DOWN

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

The line has a configurable pull-down resistor enabled.

### 5.8.3.5 FLAG\_BIAS\_PULL\_UP

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

The line has a configurable pull-up resistor enabled.

### 5.8.3.6 FLAG\_OPEN\_DRAIN

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

The line is an open-drain port.

### 5.8.3.7 FLAG\_OPEN\_SOURCE

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

The line is an open-source port.

### 5.8.3.8 flags

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

Additional request flags.

### 5.8.3.9 request\_type

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

Type of the request.

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

- [gpiod.hpp](#)





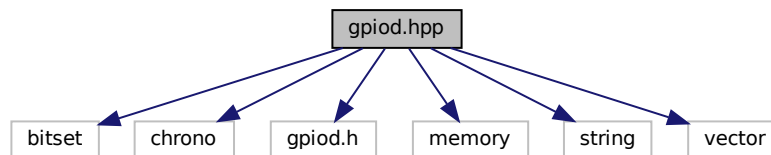
## Chapter 6

# File Documentation

### 6.1 gpiod.hpp File Reference

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

Include dependency graph for gpiod.hpp:



### Classes

- 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::chip\\_iter](#)  
*Allows to iterate over all GPIO chips present on the system.*
- class [gpiod::line\\_iter](#)  
*Allows to iterate over all lines owned by a GPIO chip.*

## Functions

- GPIOD\_API line `gpiod::find_line` (const ::std::string &name)  
*Find a GPIO line by name. Search all GPIO chips present on the system.*
- GPIOD\_API chip\_iter `gpiod::make_chip_iter` (void)  
*Create a new `chip_iter`.*
- GPIOD\_API chip\_iter `gpiod::begin` (chip\_iter iter) noexcept  
*Support for range-based loops for chip iterators.*
- GPIOD\_API chip\_iter `gpiod::end` (const chip\_iter &iter) noexcept  
*Support for range-based loops for chip iterators.*
- 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.*

## 6.2 gpiod.hpp

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

```

1  /* SPDX-License-Identifier: LGPL-2.1-or-later */
2  /*
3   * This file is part of libgpiod.
4   *
5   * Copyright (C) 2017-2018 Bartosz Golaszewski <bartekgola@gmail.com>
6   */
7
8  #ifndef __LIBGPIOD_GPIOD_CXX_HPP__
9  #define __LIBGPIOD_GPIOD_CXX_HPP__
10
11 #include <bitset>
12 #include <chrono>
13 #include <gpiod.h>
14 #include <memory>
15 #include <string>
16 #include <vector>
17
18 namespace gpiod {
19
20 class line;
21 class line_bulk;
22 class line_iter;
23 class chip_iter;
24 struct line_event;
25
26 class chip
27 {
28 public:
29
30     GPIOD_API chip(void) = default;
31
32     GPIOD_API chip(const ::std::string& device, int how = OPEN_LOOKUP);
33
34     GPIOD_API chip(const chip& other) = default;
35
36     GPIOD_API chip(chip&& other) = default;
37
38     GPIOD_API chip& operator=(const chip& other) = default;
39
40     GPIOD_API chip& operator=(chip&& other) = default;
41
42     GPIOD_API ~chip(void) = default;
43
44     GPIOD_API void open(const ::std::string &device, int how = OPEN_LOOKUP);
45
46     GPIOD_API void reset(void) noexcept;
47
48     GPIOD_API ::std::string name(void) const;
49
50     GPIOD_API ::std::string label(void) const;
51
52     GPIOD_API unsigned int num_lines(void) const;
53
54     GPIOD_API line get_line(unsigned int offset) const;

```

```

128
134     GPIOD_API line find_line(const ::std::string& name) const;
135
141     GPIOD_API line_bulk get_lines(const ::std::vector<unsigned int>& offsets) const;
142
147     GPIOD_API line_bulk get_all_lines(void) const;
148
154     GPIOD_API line_bulk find_lines(const ::std::vector<::std::string>& names) const;
155
161     GPIOD_API bool operator==(const chip& rhs) const noexcept;
162
168     GPIOD_API bool operator!=(const chip& rhs) const noexcept;
169
174     GPIOD_API explicit operator bool(void) const noexcept;
175
180     GPIOD_API bool operator!(void) const noexcept;
181
186     enum : int {
187         OPEN_LOOKUP = 1,
189         OPEN_BY_PATH,
191         OPEN_BY_NAME,
193         OPEN_BY_LABEL,
195         OPEN_BY_NUMBER,
197     };
198
199 private:
200
201     chip(::gpiod_chip* chip);
202
203     void throw_if_noref(void) const;
204
205     ::std::shared_ptr<::gpiod_chip> _m_chip;
206
207     friend chip_iter;
208     friend line_iter;
209 };
210
214 struct line_request
215 {
219     enum : int {
220         DIRECTION_AS_IS = 1,
222         DIRECTION_INPUT,
224         DIRECTION_OUTPUT,
226         EVENT_FALLING_EDGE,
228         EVENT_RISING_EDGE,
230         EVENT_BOTH_EDGES,
232     };
233
234     GPIOD_API static const ::std::bitset<32> FLAG_ACTIVE_LOW;
236     GPIOD_API static const ::std::bitset<32> FLAG_OPEN_SOURCE;
238     GPIOD_API static const ::std::bitset<32> FLAG_OPEN_DRAIN;
240     GPIOD_API static const ::std::bitset<32> FLAG_BIAS_DISABLE;
242     GPIOD_API static const ::std::bitset<32> FLAG_BIAS_PULL_DOWN;
244     GPIOD_API static const ::std::bitset<32> FLAG_BIAS_PULL_UP;
247     ::std::string consumer;
249     int request_type;
251     ::std::bitset<32> flags;
253 };
254
262 class line
263 {
264 public:
265
269     GPIOD_API line(void);
270
275     GPIOD_API line(const line& other) = default;
276
281     GPIOD_API line(line&& other) = default;
282
288     GPIOD_API line& operator=(const line& other) = default;
289
295     GPIOD_API line& operator=(line&& other) = default;
296
300     GPIOD_API ~line(void) = default;
301
306     GPIOD_API unsigned int offset(void) const;
307
312     GPIOD_API ::std::string name(void) const;
313
319     GPIOD_API ::std::string consumer(void) const;
320
325     GPIOD_API int direction(void) const;
326
331     GPIOD_API int active_state(void) const;
332
337     GPIOD_API int bias(void) const;
338

```

```

344     GPIOD_API bool is_used(void) const;
345
350     GPIOD_API bool is_open_drain(void) const;
351
356     GPIOD_API bool is_open_source(void) const;
357
363     GPIOD_API void request(const line_request& config, int default_val = 0) const;
364
368     GPIOD_API void release(void) const;
369
374     GPIOD_API bool is_requested(void) const;
375
380     GPIOD_API int get_value(void) const;
381
386     GPIOD_API void set_value(int val) const;
387
394     GPIOD_API void set_config(int direction, ::std::bitset<32> flags,
395                               int value = 0) const;
396
401     GPIOD_API void set_flags(::std::bitset<32> flags) const;
402
406     GPIOD_API void set_direction_input() const;
407
412     GPIOD_API void set_direction_output(int value = 0) const;
413
420     GPIOD_API bool event_wait(const ::std::chrono::nanoseconds& timeout) const;
421
426     GPIOD_API line_event event_read(void) const;
427
432     GPIOD_API ::std::vector<line_event> event_read_multiple(void) const;
433
438     GPIOD_API int event_get_fd(void) const;
439
444     GPIOD_API const chip& get_chip(void) const;
445
449     GPIOD_API void update(void) const;
450
458     GPIOD_API void reset(void);
459
465     GPIOD_API bool operator==(const line& rhs) const noexcept;
466
472     GPIOD_API bool operator!=(const line& rhs) const noexcept;
473
478     GPIOD_API explicit operator bool(void) const noexcept;
479
485     GPIOD_API bool operator!(void) const noexcept;
486
490     enum : int {
491         DIRECTION_INPUT = 1,
493         DIRECTION_OUTPUT,
495     };
496
500     enum : int {
501         ACTIVE_LOW = 1,
503         ACTIVE_HIGH,
505     };
506
510     enum : int {
511         BIAS_AS_IS = 1,
513         BIAS_DISABLE,
515         BIAS_PULL_UP,
517         BIAS_PULL_DOWN,
519     };
520
521 private:
522
523     line(::gpiod_line* line, const chip& owner);
524
525     void throw_if_null(void) const;
526     line_event make_line_event(const ::gpiod_line_event& event) const noexcept;
527
528     ::gpiod_line* _m_line;
529     chip _m_chip;
530
531     friend chip;
532     friend line_bulk;
533     friend line_iter;
534 };
535
541 GPIOD_API line find_line(const ::std::string& name);
542
546 struct line_event
547 {
551     enum : int {
552         RISING_EDGE = 1,
554         FALLING_EDGE,
556     };

```

```

557
558     ::std::chrono::nanoseconds timestamp;
559     int event_type;
560     line source;
561 };
562
563 class line_bulk
564 {
565 public:
566     GPIOD_API line_bulk(void) = default;
567
568     GPIOD_API line_bulk(const ::std::vector<line>& lines);
569
570     GPIOD_API line_bulk(const line_bulk& other) = default;
571
572     GPIOD_API line_bulk(line_bulk&& other) = default;
573
574     GPIOD_API line_bulk& operator=(const line_bulk& other) = default;
575
576     GPIOD_API line_bulk& operator=(line_bulk&& other) = default;
577
578     GPIOD_API ~line_bulk(void) = default;
579
580     GPIOD_API void append(const line& new_line);
581
582     GPIOD_API line& get(unsigned int offset);
583
584     GPIOD_API line& operator[](unsigned int offset);
585
586     GPIOD_API unsigned int size(void) const noexcept;
587
588     GPIOD_API bool empty(void) const noexcept;
589
590     GPIOD_API void clear(void);
591
592     GPIOD_API void request(const line_request& config,
593                           const ::std::vector<int> default_vals = ::std::vector<int>()) const;
594
595     GPIOD_API void release(void) const;
596
597     GPIOD_API ::std::vector<int> get_values(void) const;
598
599     GPIOD_API void set_values(const ::std::vector<int>& values) const;
600
601     GPIOD_API void set_config(int direction, ::std::bitset<32> flags,
602                              const ::std::vector<int> values = ::std::vector<int>()) const;
603
604     GPIOD_API void set_flags(::std::bitset<32> flags) const;
605
606     GPIOD_API void set_direction_input() const;
607
608     GPIOD_API void set_direction_output(const ::std::vector<int>& values) const;
609
610     GPIOD_API line_bulk event_wait(const ::std::chrono::nanoseconds& timeout) const;
611
612     GPIOD_API explicit operator bool(void) const noexcept;
613
614     GPIOD_API bool operator!(void) const noexcept;
615
616     GPIOD_API static const unsigned int MAX_LINES;
617
618     class iterator
619     {
620     public:
621         GPIOD_API iterator(void) = default;
622
623         GPIOD_API iterator(const iterator& other) = default;
624
625         GPIOD_API iterator(iterator&& other) = default;
626
627         GPIOD_API iterator& operator=(const iterator& other) = default;
628
629         GPIOD_API iterator& operator=(iterator&& other) = default;
630
631         GPIOD_API ~iterator(void) = default;
632
633         GPIOD_API iterator& operator++(void);
634
635         GPIOD_API const line& operator*(void) const;
636
637         GPIOD_API const line* operator--(void) const;
638
639         GPIOD_API bool operator==(const iterator& rhs) const noexcept;
640
641         GPIOD_API bool operator!=(const iterator& rhs) const noexcept;

```

```

818
819     private:
820
821         iterator(const ::std::vector<line>::iterator& it);
822
823         ::std::vector<line>::iterator _m_iter;
824
825         friend line_bulk;
826     };
827
828     GPIOD_API iterator begin(void) noexcept;
829
830     GPIOD_API iterator end(void) noexcept;
831
832 private:
833
834     void throw_if_empty(void) const;
835     void to_line_bulk(::gpiod_line_bulk* bulk) const;
836
837     ::std::vector<line> _m_bulk;
838 };
839
840 GPIOD_API chip_iter make_chip_iter(void);
841
842 GPIOD_API chip_iter begin(chip_iter iter) noexcept;
843
844 GPIOD_API chip_iter end(const chip_iter& iter) noexcept;
845
846 class chip_iter
847 {
848 public:
849     GPIOD_API chip_iter(void) = default;
850
851     GPIOD_API chip_iter(const chip_iter& other) = default;
852
853     GPIOD_API chip_iter(chip_iter&& other) = default;
854
855     GPIOD_API chip_iter& operator=(const chip_iter& other) = default;
856
857     GPIOD_API chip_iter& operator=(chip_iter&& other) = default;
858
859     GPIOD_API ~chip_iter(void) = default;
860
861     GPIOD_API chip_iter& operator++(void);
862
863     GPIOD_API const chip& operator*(void) const;
864
865     GPIOD_API const chip* operator->(void) const;
866
867     GPIOD_API bool operator==(const chip_iter& rhs) const noexcept;
868
869     GPIOD_API bool operator!=(const chip_iter& rhs) const noexcept;
870
871 private:
872
873     chip_iter(::gpiod_chip_iter* iter);
874
875     ::std::shared_ptr<::gpiod_chip_iter> _m_iter;
876     chip _m_current;
877
878     friend chip_iter make_chip_iter(void);
879 };
880
881 GPIOD_API line_iter begin(line_iter iter) noexcept;
882
883 GPIOD_API line_iter end(const line_iter& iter) noexcept;
884
885 class line_iter
886 {
887 public:
888     GPIOD_API line_iter(void) = default;
889
890     GPIOD_API line_iter(const chip& owner);
891
892     GPIOD_API line_iter(const line_iter& other) = default;
893
894     GPIOD_API line_iter(line_iter&& other) = default;
895
896     GPIOD_API line_iter& operator=(const line_iter& other) = default;
897
898     GPIOD_API line_iter& operator=(line_iter&& other) = default;
899
900     GPIOD_API ~line_iter(void) = default;
901
902     GPIOD_API line_iter& operator++(void);

```

```
1025
1030     GPIOD_API const line& operator*(void) const;
1031
1036     GPIOD_API const line* operator->(void) const;
1037
1044     GPIOD_API bool operator==(const line_iter& rhs) const noexcept;
1045
1052     GPIOD_API bool operator!=(const line_iter& rhs) const noexcept;
1053
1054 private:
1055
1056     ::std::shared_ptr<::gpiod_line_iter> _m_iter;
1057     line _m_current;
1058 };
1059
1064 } /* namespace gpiod */
1065
1066 #endif /* __LIBGPIOD_GPIOD_CXX_HPP__ */
```





# Index

- ACTIVE\_HIGH
  - [gpiod::line](#), [29](#)
- ACTIVE\_LOW
  - [gpiod::line](#), [29](#)
- active\_state
  - [gpiod::line](#), [30](#)
- append
  - [gpiod::line\\_bulk](#), [41](#)
- begin
  - [C++ bindings](#), [8](#)
  - [gpiod::line\\_bulk](#), [41](#)
- bias
  - [gpiod::line](#), [30](#)
- BIAS\_AS\_IS
  - [gpiod::line](#), [29](#)
- BIAS\_DISABLE
  - [gpiod::line](#), [29](#)
- BIAS\_PULL\_DOWN
  - [gpiod::line](#), [29](#)
- BIAS\_PULL\_UP
  - [gpiod::line](#), [29](#)
- C++ bindings, [7](#)
  - [begin](#), [8](#)
  - [end](#), [8](#), [9](#)
  - [find\\_line](#), [9](#)
  - [make\\_chip\\_iter](#), [9](#)
- chip
  - [gpiod::chip](#), [13](#)
- chip\_iter
  - [gpiod::chip\\_iter](#), [19](#)
- consumer
  - [gpiod::line](#), [30](#)
  - [gpiod::line\\_request](#), [54](#)
- direction
  - [gpiod::line](#), [31](#)
- DIRECTION\_AS\_IS
  - [gpiod::line\\_request](#), [53](#)
- DIRECTION\_INPUT
  - [gpiod::line](#), [29](#)
  - [gpiod::line\\_request](#), [53](#)
- DIRECTION\_OUTPUT
  - [gpiod::line](#), [29](#)
  - [gpiod::line\\_request](#), [53](#)
- empty
  - [gpiod::line\\_bulk](#), [42](#)
- end
  - [C++ bindings](#), [8](#), [9](#)
  - [gpiod::line\\_bulk](#), [42](#)
- EVENT\_BOTH\_EDGES
  - [gpiod::line\\_request](#), [53](#)
- EVENT\_FALLING\_EDGE
  - [gpiod::line\\_request](#), [53](#)
- event\_get\_fd
  - [gpiod::line](#), [31](#)
- event\_read
  - [gpiod::line](#), [31](#)
- event\_read\_multiple
  - [gpiod::line](#), [31](#)
- EVENT\_RISING\_EDGE
  - [gpiod::line\\_request](#), [53](#)
- event\_type
  - [gpiod::line\\_event](#), [48](#)
- event\_wait
  - [gpiod::line](#), [32](#)
  - [gpiod::line\\_bulk](#), [42](#)
- FALLING\_EDGE
  - [gpiod::line\\_event](#), [48](#)
- find\_line
  - [C++ bindings](#), [9](#)
  - [gpiod::chip](#), [13](#)
- find\_lines
  - [gpiod::chip](#), [14](#)
- FLAG\_ACTIVE\_LOW
  - [gpiod::line\\_request](#), [54](#)
- FLAG\_BIAS\_DISABLE
  - [gpiod::line\\_request](#), [54](#)
- FLAG\_BIAS\_PULL\_DOWN
  - [gpiod::line\\_request](#), [54](#)
- FLAG\_BIAS\_PULL\_UP
  - [gpiod::line\\_request](#), [54](#)
- FLAG\_OPEN\_DRAIN
  - [gpiod::line\\_request](#), [54](#)
- FLAG\_OPEN\_SOURCE
  - [gpiod::line\\_request](#), [54](#)
- flags
  - [gpiod::line\\_request](#), [54](#)
- get
  - [gpiod::line\\_bulk](#), [43](#)
- get\_all\_lines
  - [gpiod::chip](#), [14](#)
- get\_chip
  - [gpiod::line](#), [32](#)
- get\_line
  - [gpiod::chip](#), [14](#)

- get\_lines
  - gpiod::chip, 15
- get\_value
  - gpiod::line, 32
- get\_values
  - gpiod::line\_bulk, 43
- gpiod.hpp, 57
- gpiod::chip, 11
  - chip, 13
  - find\_line, 13
  - find\_lines, 14
  - get\_all\_lines, 14
  - get\_line, 14
  - get\_lines, 15
  - label, 15
  - name, 15
  - num\_lines, 15
  - open, 16
  - OPEN\_BY\_LABEL, 12
  - OPEN\_BY\_NAME, 12
  - OPEN\_BY\_NUMBER, 12
  - OPEN\_BY\_PATH, 12
  - OPEN\_LOOKUP, 12
  - operator bool, 16
  - operator!, 16
  - operator!=, 16
  - operator=, 17
  - operator==, 18
- gpiod::chip\_iter, 18
  - chip\_iter, 19
  - make\_chip\_iter, 21
  - operator!=, 19
  - operator\*, 20
  - operator++, 20
  - operator->, 20
  - operator=, 20, 21
  - operator==, 21
- gpiod::line, 26
  - ACTIVE\_HIGH, 29
  - ACTIVE\_LOW, 29
  - active\_state, 30
  - bias, 30
  - BIAS\_AS\_IS, 29
  - BIAS\_DISABLE, 29
  - BIAS\_PULL\_DOWN, 29
  - BIAS\_PULL\_UP, 29
  - consumer, 30
  - direction, 31
  - DIRECTION\_INPUT, 29
  - DIRECTION\_OUTPUT, 29
  - event\_get\_fd, 31
  - event\_read, 31
  - event\_read\_multiple, 31
  - event\_wait, 32
  - get\_chip, 32
  - get\_value, 32
  - is\_open\_drain, 32
  - is\_open\_source, 33
  - is\_requested, 33
  - is\_used, 33
  - line, 29, 30
  - name, 33
  - offset, 34
  - operator bool, 34
  - operator!, 34
  - operator!=, 34
  - operator=, 35
  - operator==, 35
  - request, 37
  - reset, 37
  - set\_config, 37
  - set\_direction\_output, 38
  - set\_flags, 38
  - set\_value, 38
- gpiod::line\_bulk, 38
  - append, 41
  - begin, 41
  - empty, 42
  - end, 42
  - event\_wait, 42
  - get, 43
  - get\_values, 43
  - line\_bulk, 40, 41
  - operator bool, 43
  - operator!, 43
  - operator=, 44
  - operator[], 44
  - request, 45
  - set\_config, 45
  - set\_direction\_output, 46
  - set\_flags, 46
  - set\_values, 46
  - size, 46
- gpiod::line\_bulk::iterator, 22
  - iterator, 23
  - operator!=, 23
  - operator\*, 24
  - operator++, 24
  - operator->, 24
  - operator=, 24, 25
  - operator==, 25
- gpiod::line\_event, 47
  - event\_type, 48
  - FALLING\_EDGE, 48
  - RISING\_EDGE, 48
  - source, 48
  - timestamp, 48
- gpiod::line\_iter, 48
  - line\_iter, 49, 50
  - operator!=, 50
  - operator\*, 50
  - operator++, 51
  - operator->, 51
  - operator=, 51, 52
  - operator==, 52
- gpiod::line\_request, 52

- consumer, [54](#)
- DIRECTION\_AS\_IS, [53](#)
- DIRECTION\_INPUT, [53](#)
- DIRECTION\_OUTPUT, [53](#)
- EVENT\_BOTH\_EDGES, [53](#)
- EVENT\_FALLING\_EDGE, [53](#)
- EVENT\_RISING\_EDGE, [53](#)
- FLAG\_ACTIVE\_LOW, [54](#)
- FLAG\_BIAS\_DISABLE, [54](#)
- FLAG\_BIAS\_PULL\_DOWN, [54](#)
- FLAG\_BIAS\_PULL\_UP, [54](#)
- FLAG\_OPEN\_DRAIN, [54](#)
- FLAG\_OPEN\_SOURCE, [54](#)
- flags, [54](#)
- request\_type, [55](#)

is\_open\_drain

- gpiod::line, [32](#)

is\_open\_source

- gpiod::line, [33](#)

is\_requested

- gpiod::line, [33](#)

is\_used

- gpiod::line, [33](#)

iterator

- gpiod::line\_bulk::iterator, [23](#)

label

- gpiod::chip, [15](#)

line

- gpiod::line, [29](#), [30](#)

line\_bulk

- gpiod::line\_bulk, [40](#), [41](#)

line\_iter

- gpiod::line\_iter, [49](#), [50](#)

make\_chip\_iter

- C++ bindings, [9](#)
- gpiod::chip\_iter, [21](#)

name

- gpiod::chip, [15](#)
- gpiod::line, [33](#)

num\_lines

- gpiod::chip, [15](#)

offset

- gpiod::line, [34](#)

open

- gpiod::chip, [16](#)

OPEN\_BY\_LABEL

- gpiod::chip, [12](#)

OPEN\_BY\_NAME

- gpiod::chip, [12](#)

OPEN\_BY\_NUMBER

- gpiod::chip, [12](#)

OPEN\_BY\_PATH

- gpiod::chip, [12](#)

OPEN\_LOOKUP

- gpiod::chip, [12](#)

operator bool

- gpiod::chip, [16](#)
- gpiod::line, [34](#)
- gpiod::line\_bulk, [43](#)

operator!

- gpiod::chip, [16](#)
- gpiod::line, [34](#)
- gpiod::line\_bulk, [43](#)

operator!=

- gpiod::chip, [16](#)
- gpiod::chip\_iter, [19](#)
- gpiod::line, [34](#)
- gpiod::line\_bulk::iterator, [23](#)
- gpiod::line\_iter, [50](#)

operator\*

- gpiod::chip\_iter, [20](#)
- gpiod::line\_bulk::iterator, [24](#)
- gpiod::line\_iter, [50](#)

operator++

- gpiod::chip\_iter, [20](#)
- gpiod::line\_bulk::iterator, [24](#)
- gpiod::line\_iter, [51](#)

operator->

- gpiod::chip\_iter, [20](#)
- gpiod::line\_bulk::iterator, [24](#)
- gpiod::line\_iter, [51](#)

operator=

- gpiod::chip, [17](#)
- gpiod::chip\_iter, [20](#), [21](#)
- gpiod::line, [35](#)
- gpiod::line\_bulk, [44](#)
- gpiod::line\_bulk::iterator, [24](#), [25](#)
- gpiod::line\_iter, [51](#), [52](#)

operator==

- gpiod::chip, [18](#)
- gpiod::chip\_iter, [21](#)
- gpiod::line, [35](#)
- gpiod::line\_bulk::iterator, [25](#)
- gpiod::line\_iter, [52](#)

operator[]

- gpiod::line\_bulk, [44](#)

request

- gpiod::line, [37](#)
- gpiod::line\_bulk, [45](#)

request\_type

- gpiod::line\_request, [55](#)

reset

- gpiod::line, [37](#)

RISING\_EDGE

- gpiod::line\_event, [48](#)

set\_config

- gpiod::line, [37](#)
- gpiod::line\_bulk, [45](#)

set\_direction\_output

- gpiod::line, [38](#)
- gpiod::line\_bulk, [46](#)

- set\_flags
  - gpiod::line, [38](#)
  - gpiod::line\_bulk, [46](#)
- set\_value
  - gpiod::line, [38](#)
- set\_values
  - gpiod::line\_bulk, [46](#)
- size
  - gpiod::line\_bulk, [46](#)
- source
  - gpiod::line\_event, [48](#)
- timestamp
  - gpiod::line\_event, [48](#)