My Project

Generated by Doxygen 1.9.8

1 Topic Index 1

1.1 Topics 1

2 Class Index 3

2.1 Class List 3

3 File Index 5

3.1 File List 5

4 Topic Documentation 7

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

5 Class Documentation 11

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() 13

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() 15

5.1.4.9 open() 16

5.1.4.10 operator bool() 16

5.1.4.11 operator"!() 16

5.1.4.12 operator"!=() 16

5.1.4.13 operator=() `[1/2]` 17

5.1.4.14 operator=() `[2/2]` 17

5.1.4.15 operator==() 17
5.2 gpiod::chip_iter Class Reference 18
5.2.1 Detailed Description 18
5.2.2 Constructor & Destructor Documentation 19
5.2.2.1 chip_iter() [1/2] 19
5.2.2.2 chip_iter() [2/2] 20
5.2.3 Member Function Documentation 20
5.2.3.1 operator"!=() 20
5.2.3.2 operator*() 20
5.2.3.3 operator++() 21
5.2.3.4 operator->() 21
5.2.3.5 operator=() [1/2] 21
5.2.3.6 operator=() [2/2] 21
5.2.3.7 operator==() 22
5.2.4 Friends And Related Symbol Documentation 22
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 24
5.3.3.1 operator"!=() 24
5.3.3.2 operator*() 24
5.3.3.3 operator++() 24
5.3.3.4 operator->() 25
5.3.3.5 operator=() [1/2] 25
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 28
5.4.2 Member Enumeration Documentation 28
5.4.2.1 anonymous enum 28
5.4.2.2 anonymous enum 28
5.4.2.3 anonymous enum 28
5.4.3 Constructor & Destructor Documentation 29
5.4.3.1 line() [1/2] 29
5.4.3.2 line() [2/2] 29
5.4.4 Member Function Documentation 29
5.4.4.1 active_state() 29
5.4.4.2 bias() 29
5.4.4.3 consumer() 30

5.4.4.4 direction() 30
5.4.4.5 event_get_fd() 30
5.4.4.6 event_read() 30
5.4.4.7 event_read_multiple() 30
5.4.4.8 event_wait() 30
5.4.4.9 get_chip() 31
5.4.4.10 get_value() 31
5.4.4.11 is_open_drain() 31
5.4.4.12 is_open_source() 31
5.4.4.13 is_requested() 32
5.4.4.14 is_used() 32
5.4.4.15 name() 32
5.4.4.16 offset() 32
5.4.4.17 operator bool() 33
5.4.4.18 operator"!() 33
5.4.4.19 operator"!=() 33
5.4.4.20 operator=() [1/2] 33
5.4.4.21 operator=() [2/2] 34
5.4.4.22 operator==() 34
5.4.4.23 request() 34
5.4.4.24 reset() 35
5.4.4.25 set_config() 35
5.4.4.26 set_direction_output() 35
5.4.4.27 set_flags() 35
5.4.4.28 set_value() 36
5.5 gpiod::line_bulk Class Reference 36
5.5.1 Detailed Description 37
5.5.2 Constructor & Destructor Documentation 37
5.5.2.1 line_bulk() [1/3] 37
5.5.2.2 line_bulk() [2/3] 38
5.5.2.3 line_bulk() [3/3] 38
5.5.3 Member Function Documentation 38
5.5.3.1 append() 38
5.5.3.2 begin() 39
5.5.3.3 empty() 39
5.5.3.4 end() 39
5.5.3.5 event_wait() 39
5.5.3.6 get() 40
5.5.3.7 get_values() 40
5.5.3.8 operator bool() 40
5.5.3.9 operator"!() 40
5.5.3.10 operator=() [1/2] 41

5.5.3.11 operator=() [2/2] 42
5.5.3.12 operator[]() 42
5.5.3.13 request() 42
5.5.3.14 set_config() 43
5.5.3.15 set_direction_output() 43
5.5.3.16 set_flags() 43
5.5.3.17 set_values() 43
5.5.3.18 size() 44
5.6 gpiod::line_event Struct Reference 44
5.6.1 Detailed Description 45
5.6.2 Member Enumeration Documentation 45
5.6.2.1 anonymous enum 45
5.6.3 Member Data Documentation 45
5.6.3.1 event_type 45
5.6.3.2 source 45
5.6.3.3 timestamp 45
5.7 gpiod::line_iter Class Reference 46
5.7.1 Detailed Description 46
5.7.2 Constructor & Destructor Documentation 46
5.7.2.1 line_iter() [1/3] 46
5.7.2.2 line_iter() [2/3] 47
5.7.2.3 line_iter() [3/3] 47
5.7.3 Member Function Documentation 47
5.7.3.1 operator"!=() 47
5.7.3.2 operator*() 47
5.7.3.3 operator++() 48
5.7.3.4 operator->() 48
5.7.3.5 operator=() [1/2] 48
5.7.3.6 operator=() [2/2] 48
5.7.3.7 operator==() 49
5.8 gpiod::line_request Struct Reference 49
5.8.1 Detailed Description 50
5.8.2 Member Enumeration Documentation 50
5.8.2.1 anonymous enum 50
5.8.3 Member Data Documentation 50
5.8.3.1 consumer 50
5.8.3.2 FLAG_ACTIVE_LOW 50
5.8.3.3 FLAG_BIAS_DISABLE 51
5.8.3.4 FLAG_BIAS_PULL_DOWN 51
5.8.3.5 FLAG_BIAS_PULL_UP 51
5.8.3.6 FLAG_OPEN_DRAIN 51
5.8.3.7 FLAG_OPEN_SOURCE 51

5.8.3.8 flags 51

5.8.3.9 request_type 51

6 File Documentation **53**

6.1 gpiod.hpp File Reference 53

6.2 gpiod.hpp 54

Index **61**

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

C++ bindings 7

Chapter 2

Class Index

2.1 Class List

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

gpiod::chip
Represents a GPIO chip 11
gpiod::chip_iter
Allows to iterate over all GPIO chips present on the system 18
gpiod::line_bulk::iterator
Iterator for iterating over lines held by line_bulk 22
gpiod::line
Represents a single GPIO line 26
gpiod::line_bulk
Represents a set of GPIO lines 36
gpiod::line_event
Describes a single GPIO line event 44
gpiod::line_iter
Allows to iterate over all lines owned by a GPIO chip 46
gpiod::line_request
Stores the configuration for line requests 49

Chapter 3

File Index

3.1 File List

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

gpiod.hpp 53

Chapter 4

Topic 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

iter	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

iter	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

iter	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

iter	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

name	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 gpiod::chip Class Reference

Represents a GPIO chip.

```
#include <gpiod.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

- GPIOD_API **chip** (void)=default

  Default constructor. Creates an empty GPIO chip object.
- GPIOD_API chip (const ::std::string &device, int how=OPEN_LOOKUP)

  Constructor. Opens the chip using chip::open.
- GPIOD_API chip (const chip &other)=default

  Copy constructor. References the object held by other.
- GPIOD_API chip (chip &&other)=default

  Move constructor. References the object held by other.
- GPIOD_API chip & operator= (const chip &other)=default

  Assignment operator. References the object held by other.
- GPIOD_API chip & operator= (chip &&other)=default

  Move assignment operator. References the object held by other.
- GPIOD_API ∼**chip** (void)=default

  Destructor. Unreferences the internal chip object.
- GPIOD_API void open (const ::std::string &device, int how=OPEN_LOOKUP)

  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.
- 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

device	String describing the GPIO chip.
how	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

other	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

other	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

name	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

names	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

offset	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

offsets	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()

```
GPIOD_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()

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

Open a GPIO chip.

Parameters

device	String describing the GPIO chip.
how	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()

```
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.

5.1.4.11 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.

5.1.4.12 operator"!=()

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

Inequality operator.

Parameters

rhs	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

other	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

other	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

rhs	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

other	Other chip_iter.

5.2.2.2 chip_iter() [2/2]

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

Move constructor.

Parameters

other	Other chip_iter.

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

rhs	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

other	Other chip_iter.

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

other	Other chip_iter.

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

rhs	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 Symbol 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

other	Other line_bulk iterator.

5.3.2.2 iterator() [2/2]

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

Move constructor.

Parameters

other	Other line_bulk 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

rhs	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

other	Other line_bulk 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

other	Other line_bulk 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

rhs	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 & Destructor Documentation

5.4.3.1 line() [1/2]

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

Copy constructor.

Parameters

other	Other line object.

5.4.3.2 line() [2/2]

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

Move constructor.

Parameters

other	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

timeout	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

Offet 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

rhs	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

other	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

other	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

rhs	Right-hand side of the equation.

Returns

True if both objects reference the same line, fale 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

config	Request config (see gpiod::line_request).
default_val	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

direction	New direction.
flags	Replacement flags.
value	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

value	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

flags	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

val	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

lines	Vector of gpiod::line 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

other	Other line_bulk object.

5.5.2.3 line_bulk() [3/3]

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

Move constructor.

Parameters

other	Other line_bulk 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

new_line	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

timeout	Number of nanoseconds to wait before returning an empty line_bulk.

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

offset	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

other	Other line_bulk 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

other	Other line_bulk 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

offset	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

config	Request config (see gpiod::line_request).
default_vals	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

direction	New direction.
flags	Replacement flags.
values	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.

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

values	Vector of values to set. Must be the same size as the number of lines held by this line_bulk.

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

flags	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

values	Vector of values to set. Must be the same size as the number of lines held by this line_bulk.

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:

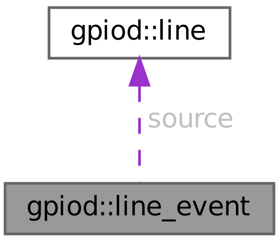
- gpiod.hpp

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

owner	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

other	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

other	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

rhs	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

other	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

other	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

rhs	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

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.

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

```
GPIOD_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

```
GPIOD_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

```
GPIOD_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

```
GPIOD_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

```
GPIOD_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

```
GPIOD_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

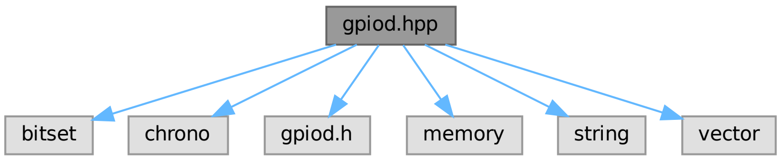
Chapter 6

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.

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

#ifndef __LIBGPIOD_GPIOD_CXX_HPP__
#define __LIBGPIOD_GPIOD_CXX_HPP__

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

namespace gpiod {

class line;
class line_bulk;
class line_iter;
class chip_iter;
struct line_event;

class chip
{
public:

    GPIOD_API chip(void) = default;

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

    GPIOD_API chip(const chip& other) = default;

    GPIOD_API chip(chip&& other) = default;

    GPIOD_API chip& operator=(const chip& other) = default;

    GPIOD_API chip& operator=(chip&& other) = default;

    GPIOD_API ~chip(void) = default;

    GPIOD_API void open(const ::std::string &device, int how = OPEN_LOOKUP);

    GPIOD_API void reset(void) noexcept;

    GPIOD_API ::std::string name(void) const;

    GPIOD_API ::std::string label(void) const;

    GPIOD_API unsigned int num_lines(void) const;

    GPIOD_API line get_line(unsigned int offset) const;

    GPIOD_API line find_line(const ::std::string& name) const;

    GPIOD_API line_bulk get_lines(const ::std::vector<unsigned int>& offsets) const;

    GPIOD_API line_bulk get_all_lines(void) const;

    GPIOD_API line_bulk find_lines(const ::std::vector<::std::string>& names) const;

    GPIOD_API bool operator==(const chip& rhs) const noexcept;

    GPIOD_API bool operator!=(const chip& rhs) const noexcept;

    GPIOD_API explicit operator bool(void) const noexcept;

    GPIOD_API bool operator!(void) const noexcept;

    enum : int {
        OPEN_LOOKUP = 1,
        OPEN_BY_PATH,
        OPEN_BY_NAME,
        OPEN_BY_LABEL,
        OPEN_BY_NUMBER,
    };

private:

    chip(::gpiod_chip* chip);

    void throw_if_noref(void) const;

    ::std::shared_ptr<::gpiod_chip> _m_chip;

    friend chip_iter;
    friend line_iter;
};

struct line_request
{
    enum : int {
        DIRECTION_AS_IS = 1,
        DIRECTION_INPUT,
        DIRECTION_OUTPUT,
        EVENT_FALLING_EDGE,
        EVENT_RISING_EDGE,
        EVENT_BOTH_EDGES,
    };

    GPIOD_API static const ::std::bitset<32> FLAG_ACTIVE_LOW;
    GPIOD_API static const ::std::bitset<32> FLAG_OPEN_SOURCE;
    GPIOD_API static const ::std::bitset<32> FLAG_OPEN_DRAIN;
    GPIOD_API static const ::std::bitset<32> FLAG_BIAS_DISABLE;
    GPIOD_API static const ::std::bitset<32> FLAG_BIAS_PULL_DOWN;
    GPIOD_API static const ::std::bitset<32> FLAG_BIAS_PULL_UP;
    ::std::string consumer;
    int request_type;
    ::std::bitset<32> flags;
};

class line
{
public:

    GPIOD_API line(void);

    GPIOD_API line(const line& other) = default;

    GPIOD_API line(line&& other) = default;

    GPIOD_API line& operator=(const line& other) = default;

    GPIOD_API line& operator=(line&& other) = default;

    GPIOD_API ~line(void) = default;

    GPIOD_API unsigned int offset(void) const;

    GPIOD_API ::std::string name(void) const;

    GPIOD_API ::std::string consumer(void) const;

    GPIOD_API int direction(void) const;

    GPIOD_API int active_state(void) const;

    GPIOD_API int bias(void) const;

    GPIOD_API bool is_used(void) const;

    GPIOD_API bool is_open_drain(void) const;

    GPIOD_API bool is_open_source(void) const;

    GPIOD_API void request(const line_request& config, int default_val = 0) const;

    GPIOD_API void release(void) const;

    GPIOD_API bool is_requested(void) const;

    GPIOD_API int get_value(void) const;

    GPIOD_API void set_value(int val) const;

    GPIOD_API void set_config(int direction, ::std::bitset<32> flags,
                              int value = 0) const;

    GPIOD_API void set_flags(::std::bitset<32> flags) const;

    GPIOD_API void set_direction_input() const;

    GPIOD_API void set_direction_output(int value = 0) const;

    GPIOD_API bool event_wait(const ::std::chrono::nanoseconds& timeout) const;

    GPIOD_API line_event event_read(void) const;

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

    GPIOD_API int event_get_fd(void) const;

    GPIOD_API const chip& get_chip(void) const;

    GPIOD_API void update(void) const;

    GPIOD_API void reset(void);

    GPIOD_API bool operator==(const line& rhs) const noexcept;

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

    GPIOD_API explicit operator bool(void) const noexcept;

    GPIOD_API bool operator!(void) const noexcept;

    enum : int {
        DIRECTION_INPUT = 1,
        DIRECTION_OUTPUT,
    };

    enum : int {
        ACTIVE_LOW = 1,
        ACTIVE_HIGH,
    };

    enum : int {
        BIAS_AS_IS = 1,
        BIAS_DISABLE,
        BIAS_PULL_UP,
        BIAS_PULL_DOWN,
    };

private:

    line(::gpiod_line* line, const chip& owner);

    void throw_if_null(void) const;
    line_event make_line_event(const ::gpiod_line_event& event) const noexcept;

    ::gpiod_line* _m_line;
    chip _m_chip;

    friend chip;
    friend line_bulk;
    friend line_iter;
};

GPIOD_API line find_line(const ::std::string& name);

struct line_event
{
    enum : int {
        RISING_EDGE = 1,
        FALLING_EDGE,
    };

    ::std::chrono::nanoseconds timestamp;
    int event_type;
    line source;
};

class line_bulk
{
public:

    GPIOD_API line_bulk(void) = default;

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

    GPIOD_API line_bulk(const line_bulk& other) = default;

    GPIOD_API line_bulk(line_bulk&& other) = default;

    GPIOD_API line_bulk& operator=(const line_bulk& other) = default;

    GPIOD_API line_bulk& operator=(line_bulk&& other) = default;

    GPIOD_API ~line_bulk(void) = default;

    GPIOD_API void append(const line& new_line);

    GPIOD_API line& get(unsigned int offset);

    GPIOD_API line& operator[](unsigned int offset);

    GPIOD_API unsigned int size(void) const noexcept;

    GPIOD_API bool empty(void) const noexcept;

    GPIOD_API void clear(void);

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

    GPIOD_API void release(void) const;

    GPIOD_API ::std::vector<int> get_values(void) const;

    GPIOD_API void set_values(const ::std::vector<int>& values) const;

    GPIOD_API void set_config(int direction, ::std::bitset<32> flags,
                          const ::std::vector<int> values = ::std::vector<int>()) const;

    GPIOD_API void set_flags(::std::bitset<32> flags) const;

    GPIOD_API void set_direction_input() const;

    GPIOD_API void set_direction_output(const ::std::vector<int>& values) const;

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

    GPIOD_API explicit operator bool(void) const noexcept;

    GPIOD_API bool operator!(void) const noexcept;

    GPIOD_API static const unsigned int MAX_LINES;

    class iterator
    {
    public:

        GPIOD_API iterator(void) = default;

        GPIOD_API iterator(const iterator& other) = default;

        GPIOD_API iterator(iterator&& other) = default;

        GPIOD_API iterator& operator=(const iterator& other) = default;

        GPIOD_API iterator& operator=(iterator&& other) = default;

        GPIOD_API ~iterator(void) = default;

        GPIOD_API iterator& operator++(void);

        GPIOD_API const line& operator*(void) const;

        GPIOD_API const line* operator->(void) const;

        GPIOD_API bool operator==(const iterator& rhs) const noexcept;

        GPIOD_API bool operator!=(const iterator& rhs) const noexcept;

    private:

        iterator(const ::std::vector<line>::iterator& it);

        ::std::vector<line>::iterator _m_iter;

        friend line_bulk;
    };

    GPIOD_API iterator begin(void) noexcept;

    GPIOD_API iterator end(void) noexcept;

private:

    void throw_if_empty(void) const;
    void to_line_bulk(::gpiod_line_bulk* bulk) const;

    ::std::vector<line> _m_bulk;
};

GPIOD_API chip_iter make_chip_iter(void);

GPIOD_API chip_iter begin(chip_iter iter) noexcept;

GPIOD_API chip_iter end(const chip_iter& iter) noexcept;

class chip_iter
{
public:

    GPIOD_API chip_iter(void) = default;

    GPIOD_API chip_iter(const chip_iter& other) = default;

    GPIOD_API chip_iter(chip_iter&& other) = default;

    GPIOD_API chip_iter& operator=(const chip_iter& other) = default;

    GPIOD_API chip_iter& operator=(chip_iter&& other) = default;

    GPIOD_API ~chip_iter(void) = default;

    GPIOD_API chip_iter& operator++(void);

    GPIOD_API const chip& operator*(void) const;

    GPIOD_API const chip* operator->(void) const;

    GPIOD_API bool operator==(const chip_iter& rhs) const noexcept;

    GPIOD_API bool operator!=(const chip_iter& rhs) const noexcept;

private:

    chip_iter(::gpiod_chip_iter* iter);

    ::std::shared_ptr<::gpiod_chip_iter> _m_iter;
    chip _m_current;

    friend chip_iter make_chip_iter(void);
};

GPIOD_API line_iter begin(line_iter iter) noexcept;

GPIOD_API line_iter end(const line_iter& iter) noexcept;

class line_iter
{
public:

    GPIOD_API line_iter(void) = default;

    GPIOD_API line_iter(const chip& owner);

    GPIOD_API line_iter(const line_iter& other) = default;

    GPIOD_API line_iter(line_iter&& other) = default;

    GPIOD_API line_iter& operator=(const line_iter& other) = default;

    GPIOD_API line_iter& operator=(line_iter&& other) = default;

    GPIOD_API ~line_iter(void) = default;

    GPIOD_API line_iter& operator++(void);

    GPIOD_API const line& operator*(void) const;

    GPIOD_API const line* operator->(void) const;

    GPIOD_API bool operator==(const line_iter& rhs) const noexcept;

    GPIOD_API bool operator!=(const line_iter& rhs) const noexcept;

private:

    ::std::shared_ptr<::gpiod_line_iter> _m_iter;
    line _m_current;
};

} /* namespace gpiod */

#endif /* __LIBGPIOD_GPIOD_CXX_HPP__ */
```

Index

ACTIVE_HIGH
    gpiod::line, 28
ACTIVE_LOW
    gpiod::line, 28
active_state
    gpiod::line, 29
append
    gpiod::line_bulk, 38

begin
    C++ bindings, 8
    gpiod::line_bulk, 39
bias
    gpiod::line, 29
BIAS_AS_IS
    gpiod::line, 28
BIAS_DISABLE
    gpiod::line, 28
BIAS_PULL_DOWN
    gpiod::line, 28
BIAS_PULL_UP
    gpiod::line, 28

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, 20
consumer
    gpiod::line, 29
    gpiod::line_request, 50

direction
    gpiod::line, 30
DIRECTION_AS_IS
    gpiod::line_request, 50
DIRECTION_INPUT
    gpiod::line, 28
    gpiod::line_request, 50
DIRECTION_OUTPUT
    gpiod::line, 28
    gpiod::line_request, 50

empty
    gpiod::line_bulk, 39
end
    C++ bindings, 8, 9
    gpiod::line_bulk, 39
EVENT_BOTH_EDGES
    gpiod::line_request, 50
EVENT_FALLING_EDGE
    gpiod::line_request, 50
event_get_fd
    gpiod::line, 30
event_read
    gpiod::line, 30
event_read_multiple
    gpiod::line, 30
EVENT_RISING_EDGE
    gpiod::line_request, 50
event_type
    gpiod::line_event, 45
event_wait
    gpiod::line, 30
    gpiod::line_bulk, 39

FALLING_EDGE
    gpiod::line_event, 45
find_line
    C++ bindings, 9
    gpiod::chip, 13
find_lines
    gpiod::chip, 14
FLAG_ACTIVE_LOW
    gpiod::line_request, 50
FLAG_BIAS_DISABLE
    gpiod::line_request, 50
FLAG_BIAS_PULL_DOWN
    gpiod::line_request, 51
FLAG_BIAS_PULL_UP
    gpiod::line_request, 51
FLAG_OPEN_DRAIN
    gpiod::line_request, 51
FLAG_OPEN_SOURCE
    gpiod::line_request, 51
flags
    gpiod::line_request, 51

get
    gpiod::line_bulk, 40
get_all_lines
    gpiod::chip, 14
get_chip
    gpiod::line, 31
get_line
    gpiod::chip, 14

get_lines
    gpiod::chip, 15
get_value
    gpiod::line, 31
get_values
    gpiod::line_bulk, 40
gpiod.hpp, 53
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, 15
    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==, 17
gpiod::chip_iter, 18
    chip_iter, 19, 20
    make_chip_iter, 22
    operator!=, 20
    operator++, 20
    operator->, 21
    operator=, 21
    operator==, 22
    operator*, 20
gpiod::line, 26
    ACTIVE_HIGH, 28
    ACTIVE_LOW, 28
    active_state, 29
    bias, 29
    BIAS_AS_IS, 28
    BIAS_DISABLE, 28
    BIAS_PULL_DOWN, 28
    BIAS_PULL_UP, 28
    consumer, 29
    direction, 30
    DIRECTION_INPUT, 28
    DIRECTION_OUTPUT, 28
    event_get_fd, 30
    event_read, 30
    event_read_multiple, 30
    event_wait, 30
    get_chip, 31
    get_value, 31
    is_open_drain, 31
    is_open_source, 31
    is_requested, 31
    is_used, 32
    line, 29
    name, 32
    offset, 32
    operator bool, 32
    operator!, 33
    operator!=, 33
    operator=, 33, 34
    operator==, 34
    request, 34
    reset, 34
    set_config, 35
    set_direction_output, 35
    set_flags, 35
    set_value, 35
gpiod::line_bulk, 36
    append, 38
    begin, 39
    empty, 39
    end, 39
    event_wait, 39
    get, 40
    get_values, 40
    line_bulk, 37, 38
    operator bool, 40
    operator!, 40
    operator=, 40, 42
    operator[], 42
    request, 42
    set_config, 43
    set_direction_output, 43
    set_flags, 43
    set_values, 43
    size, 44
gpiod::line_bulk::iterator, 22
    iterator, 23
    operator!=, 24
    operator++, 24
    operator->, 24
    operator=, 25
    operator==, 25
    operator*, 24
gpiod::line_event, 44
    event_type, 45
    FALLING_EDGE, 45
    RISING_EDGE, 45
    source, 45
    timestamp, 45
gpiod::line_iter, 46
    line_iter, 46, 47
    operator!=, 47
    operator++, 48
    operator->, 48
    operator=, 48
    operator==, 49
    operator*, 47
gpiod::line_request, 49

consumer, 50
DIRECTION_AS_IS, 50
DIRECTION_INPUT, 50
DIRECTION_OUTPUT, 50
EVENT_BOTH_EDGES, 50
EVENT_FALLING_EDGE, 50
EVENT_RISING_EDGE, 50
FLAG_ACTIVE_LOW, 50
FLAG_BIAS_DISABLE, 50
FLAG_BIAS_PULL_DOWN, 51
FLAG_BIAS_PULL_UP, 51
FLAG_OPEN_DRAIN, 51
FLAG_OPEN_SOURCE, 51
flags, 51
request_type, 51

is_open_drain
gpiod::line, 31
is_open_source
gpiod::line, 31
is_requested
gpiod::line, 31
is_used
gpiod::line, 32
iterator
gpiod::line_bulk::iterator, 23

label
gpiod::chip, 15
line
gpiod::line, 29
line_bulk
gpiod::line_bulk, 37, 38
line_iter
gpiod::line_iter, 46, 47

make_chip_iter
C++ bindings, 9
gpiod::chip_iter, 22

name
gpiod::chip, 15
gpiod::line, 32
num_lines
gpiod::chip, 15

offset
gpiod::line, 32
open
gpiod::chip, 15
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, 32
gpiod::line_bulk, 40
operator!
gpiod::chip, 16
gpiod::line, 33
gpiod::line_bulk, 40
operator!=
gpiod::chip, 16
gpiod::chip_iter, 20
gpiod::line, 33
gpiod::line_bulk::iterator, 24
gpiod::line_iter, 47
operator++
gpiod::chip_iter, 20
gpiod::line_bulk::iterator, 24
gpiod::line_iter, 48
operator->
gpiod::chip_iter, 21
gpiod::line_bulk::iterator, 24
gpiod::line_iter, 48
operator=
gpiod::chip, 17
gpiod::chip_iter, 21
gpiod::line, 33, 34
gpiod::line_bulk, 40, 42
gpiod::line_bulk::iterator, 25
gpiod::line_iter, 48
operator==
gpiod::chip, 17
gpiod::chip_iter, 22
gpiod::line, 34
gpiod::line_bulk::iterator, 25
gpiod::line_iter, 49
operator[]
gpiod::line_bulk, 42
operator*
gpiod::chip_iter, 20
gpiod::line_bulk::iterator, 24
gpiod::line_iter, 47

request
gpiod::line, 34
gpiod::line_bulk, 42
request_type
gpiod::line_request, 51
reset
gpiod::line, 34
RISING_EDGE
gpiod::line_event, 45

set_config
gpiod::line, 35
gpiod::line_bulk, 43
set_direction_output
gpiod::line, 35
gpiod::line_bulk, 43

set_flags
gpiod::line, 35
gpiod::line_bulk, 43
set_value
gpiod::line, 35
set_values
gpiod::line_bulk, 43
size
gpiod::line_bulk, 44
source
gpiod::line_event, 45

timestamp
gpiod::line_event, 45