My Project

Generated by Doxygen 1.9.4

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

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

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 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

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:

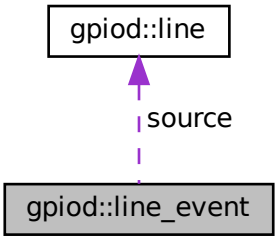
- gpiod.hpp

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

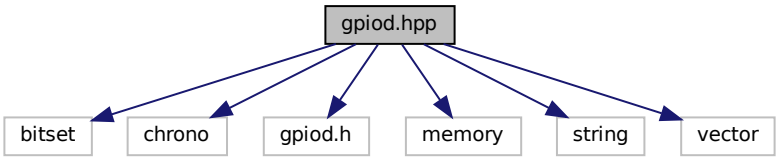
Chapter 6

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.

```
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
42 class chip
43 {
44 public:
45
49     GPIOD_API chip(void) = default;
50
56     GPIOD_API chip(const ::std::string& device, int how = OPEN_LOOKUP);
57
62     GPIOD_API chip(const chip& other) = default;
63
68     GPIOD_API chip(chip&& other) = default;
69
75     GPIOD_API chip& operator=(const chip& other) = default;
76
82     GPIOD_API chip& operator=(chip&& other) = default;
83
87     GPIOD_API ~chip(void) = default;
88
97     GPIOD_API void open(const ::std::string &device, int how = OPEN_LOOKUP);
98
102     GPIOD_API void reset(void) noexcept;
103
108     GPIOD_API ::std::string name(void) const;
109
114     GPIOD_API ::std::string label(void) const;
115
120     GPIOD_API unsigned int num_lines(void) const;
121
127     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;
560     int event_type;
562     line source;
564 };
565
572 class line_bulk
573 {
574 public:
575
579     GPIOD_API line_bulk(void) = default;
580
586     GPIOD_API line_bulk(const ::std::vector<line>& lines);
587
592     GPIOD_API line_bulk(const line_bulk& other) = default;
593
598     GPIOD_API line_bulk(line_bulk&& other) = default;
599
605     GPIOD_API line_bulk& operator=(const line_bulk& other) = default;
606
612     GPIOD_API line_bulk& operator=(line_bulk&& other) = default;
613
617     GPIOD_API ~line_bulk(void) = default;
618
625     GPIOD_API void append(const line& new_line);
626
632     GPIOD_API line& get(unsigned int offset);
633
640     GPIOD_API line& operator[](unsigned int offset);
641
646     GPIOD_API unsigned int size(void) const noexcept;
647
652     GPIOD_API bool empty(void) const noexcept;
653
657     GPIOD_API void clear(void);
658
665     GPIOD_API void request(const line_request& config,
666                            const ::std::vector<int> default_vals = ::std::vector<int>()) const;
667
671     GPIOD_API void release(void) const;
672
678     GPIOD_API ::std::vector<int> get_values(void) const;
679
685     GPIOD_API void set_values(const ::std::vector<int>& values) const;
686
695     GPIOD_API void set_config(int direction, ::std::bitset<32> flags,
696                               const ::std::vector<int> values = ::std::vector<int>()) const;
697
702     GPIOD_API void set_flags(::std::bitset<32> flags) const;
703
707     GPIOD_API void set_direction_input() const;
708
714     GPIOD_API void set_direction_output(const ::std::vector<int>& values) const;
715
723     GPIOD_API line_bulk event_wait(const ::std::chrono::nanoseconds& timeout) const;
724
729     GPIOD_API explicit operator bool(void) const noexcept;
730
735     GPIOD_API bool operator!(void) const noexcept;
736
740     GPIOD_API static const unsigned int MAX_LINES;
741
745     class iterator
746     {
747     public:
748
752         GPIOD_API iterator(void) = default;
753
758         GPIOD_API iterator(const iterator& other) = default;
759
764         GPIOD_API iterator(iterator&& other) = default;
765
771         GPIOD_API iterator& operator=(const iterator& other) = default;
772
778         GPIOD_API iterator& operator=(iterator&& other) = default;
779
783         GPIOD_API ~iterator(void) = default;
784
789         GPIOD_API iterator& operator++(void);
790
795         GPIOD_API const line& operator*(void) const;
796
801         GPIOD_API const line* operator->(void) const;
802
809         GPIOD_API bool operator==(const iterator& rhs) const noexcept;
810
817         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
832     GPIOD_API iterator begin(void) noexcept;
833
838     GPIOD_API iterator end(void) noexcept;
839
840 private:
841
842     void throw_if_empty(void) const;
843     void to_line_bulk(::gpiod_line_bulk* bulk) const;
844
845     ::std::vector<line> _m_bulk;
846 };
847
854 GPIOD_API chip_iter make_chip_iter(void);
855
861 GPIOD_API chip_iter begin(chip_iter iter) noexcept;
862
868 GPIOD_API chip_iter end(const chip_iter& iter) noexcept;
869
873 class chip_iter
874 {
875 public:
876
880     GPIOD_API chip_iter(void) = default;
881
886     GPIOD_API chip_iter(const chip_iter& other) = default;
887
892     GPIOD_API chip_iter(chip_iter&& other) = default;
893
899     GPIOD_API chip_iter& operator=(const chip_iter& other) = default;
900
906     GPIOD_API chip_iter& operator=(chip_iter&& other) = default;
907
911     GPIOD_API ~chip_iter(void) = default;
912
917     GPIOD_API chip_iter& operator++(void);
918
923     GPIOD_API const chip& operator*(void) const;
924
929     GPIOD_API const chip* operator->(void) const;
930
937     GPIOD_API bool operator==(const chip_iter& rhs) const noexcept;
938
945     GPIOD_API bool operator!=(const chip_iter& rhs) const noexcept;
946
947 private:
948
949     chip_iter(::gpiod_chip_iter* iter);
950
951     ::std::shared_ptr<::gpiod_chip_iter> _m_iter;
952     chip _m_current;
953
954     friend chip_iter make_chip_iter(void);
955 };
956
962 GPIOD_API line_iter begin(line_iter iter) noexcept;
963
969 GPIOD_API line_iter end(const line_iter& iter) noexcept;
970
974 class line_iter
975 {
976 public:
977
981     GPIOD_API line_iter(void) = default;
982
987     GPIOD_API line_iter(const chip& owner);
988
993     GPIOD_API line_iter(const line_iter& other) = default;
994
999     GPIOD_API line_iter(line_iter&& other) = default;
1000
1006     GPIOD_API line_iter& operator=(const line_iter& other) = default;
1007
1013     GPIOD_API line_iter& operator=(line_iter&& other) = default;
1014
1018     GPIOD_API ~line_iter(void) = default;
1019
1024     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