

libgpiodcxx

Generated by Doxygen 1.9.4

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 <code>gpiod::line</code> Namespace Reference	9
5.1.1 Detailed Description	10
5.1.2 Enumeration Type Documentation	10
5.1.2.1 <code>bias</code>	10
5.1.2.2 <code>clock</code>	11
5.1.2.3 <code>direction</code>	11
5.1.2.4 <code>drive</code>	11
5.1.2.5 <code>edge</code>	11
5.1.2.6 <code>value</code>	12
5.1.3 Function Documentation	12
5.1.3.1 <code>operator<<()</code> [1/10]	12
5.1.3.2 <code>operator<<()</code> [2/10]	12
5.1.3.3 <code>operator<<()</code> [3/10]	13
5.1.3.4 <code>operator<<()</code> [4/10]	13
5.1.3.5 <code>operator<<()</code> [5/10]	14
5.1.3.6 <code>operator<<()</code> [6/10]	14
5.1.3.7 <code>operator<<()</code> [7/10]	14
5.1.3.8 <code>operator<<()</code> [8/10]	15
5.1.3.9 <code>operator<<()</code> [9/10]	15
5.1.3.10 <code>operator<<()</code> [10/10]	16
6 Class Documentation	17
6.1 <code>gpiod::bad_mapping</code> Class Reference	17
6.1.1 Detailed Description	17
6.1.2 Constructor & Destructor Documentation	18
6.1.2.1 <code>bad_mapping()</code> [1/3]	18
6.1.2.2 <code>bad_mapping()</code> [2/3]	18
6.1.2.3 <code>bad_mapping()</code> [3/3]	18
6.1.3 Member Function Documentation	18
6.1.3.1 <code>operator=()</code> [1/2]	19
6.1.3.2 <code>operator=()</code> [2/2]	19

6.2 gpiod::chip Class Reference	19
6.2.1 Detailed Description	20
6.2.2 Constructor & Destructor Documentation	21
6.2.2.1 chip() [1/2]	21
6.2.2.2 chip() [2/2]	21
6.2.3 Member Function Documentation	21
6.2.3.1 close()	21
6.2.3.2 fd()	22
6.2.3.3 get_info()	22
6.2.3.4 get_line_info()	22
6.2.3.5 get_line_offset_from_name()	22
6.2.3.6 operator bool()	23
6.2.3.7 operator=()	23
6.2.3.8 path()	23
6.2.3.9 prepare_request()	24
6.2.3.10 read_info_event()	24
6.2.3.11 unwatch_line_info()	24
6.2.3.12 wait_info_event()	24
6.2.3.13 watch_line_info()	25
6.3 gpiod::chip_closed Class Reference	25
6.3.1 Detailed Description	26
6.3.2 Constructor & Destructor Documentation	26
6.3.2.1 chip_closed() [1/3]	26
6.3.2.2 chip_closed() [2/3]	26
6.3.2.3 chip_closed() [3/3]	26
6.3.3 Member Function Documentation	27
6.3.3.1 operator=() [1/2]	27
6.3.3.2 operator=() [2/2]	27
6.4 gpiod::chip_info Class Reference	27
6.4.1 Detailed Description	28
6.4.2 Constructor & Destructor Documentation	28
6.4.2.1 chip_info() [1/2]	28
6.4.2.2 chip_info() [2/2]	28
6.4.3 Member Function Documentation	29
6.4.3.1 label()	29
6.4.3.2 name()	29
6.4.3.3 num_lines()	29
6.4.3.4 operator=() [1/2]	29
6.4.3.5 operator=() [2/2]	30
6.5 gpiod::edge_event Class Reference	30
6.5.1 Detailed Description	31
6.5.2 Member Enumeration Documentation	31

6.5.2.1 event_type	31
6.5.3 Constructor & Destructor Documentation	32
6.5.3.1 edge_event() [1/2]	32
6.5.3.2 edge_event() [2/2]	32
6.5.4 Member Function Documentation	32
6.5.4.1 global_seqno()	32
6.5.4.2 line_offset()	33
6.5.4.3 line_seqno()	33
6.5.4.4 operator=() [1/2]	33
6.5.4.5 operator=() [2/2]	33
6.5.4.6 timestamp_ns()	34
6.5.4.7 type()	34
6.6 gpiod::edge_event_buffer Class Reference	34
6.6.1 Detailed Description	35
6.6.2 Constructor & Destructor Documentation	35
6.6.2.1 edge_event_buffer() [1/2]	35
6.6.2.2 edge_event_buffer() [2/2]	36
6.6.3 Member Function Documentation	36
6.6.3.1 begin()	36
6.6.3.2 capacity()	36
6.6.3.3 end()	36
6.6.3.4 get_event()	37
6.6.3.5 num_events()	38
6.6.3.6 operator=()	38
6.7 gpiod::info_event Class Reference	38
6.7.1 Detailed Description	39
6.7.2 Member Enumeration Documentation	39
6.7.2.1 event_type	39
6.7.3 Constructor & Destructor Documentation	40
6.7.3.1 info_event() [1/2]	40
6.7.3.2 info_event() [2/2]	40
6.7.4 Member Function Documentation	40
6.7.4.1 get_line_info()	40
6.7.4.2 operator=() [1/2]	40
6.7.4.3 operator=() [2/2]	41
6.7.4.4 timestamp_ns()	41
6.7.4.5 type()	41
6.8 gpiod::line_config Class Reference	42
6.8.1 Detailed Description	42
6.8.2 Constructor & Destructor Documentation	42
6.8.2.1 line_config()	42
6.8.3 Member Function Documentation	43

6.8.3.1 add_line_settings() [1/2]	43
6.8.3.2 add_line_settings() [2/2]	43
6.8.3.3 get_line_settings()	44
6.8.3.4 operator=()	44
6.8.3.5 reset()	44
6.8.3.6 set_output_values()	44
6.9 gpiod::line_info Class Reference	45
6.9.1 Detailed Description	46
6.9.2 Constructor & Destructor Documentation	46
6.9.2.1 line_info() [1/2]	46
6.9.2.2 line_info() [2/2]	46
6.9.3 Member Function Documentation	46
6.9.3.1 active_low()	46
6.9.3.2 bias()	47
6.9.3.3 consumer()	47
6.9.3.4 debounce_period()	47
6.9.3.5 debounced()	48
6.9.3.6 direction()	48
6.9.3.7 drive()	48
6.9.3.8 edge_detection()	48
6.9.3.9 event_clock()	49
6.9.3.10 name()	49
6.9.3.11 offset()	49
6.9.3.12 operator=() [1/2]	49
6.9.3.13 operator=() [2/2]	50
6.9.3.14 used()	50
6.10 gpiod::line_request Class Reference	50
6.10.1 Detailed Description	52
6.10.2 Constructor & Destructor Documentation	52
6.10.2.1 line_request()	52
6.10.3 Member Function Documentation	52
6.10.3.1 chip_name()	52
6.10.3.2 fd()	52
6.10.3.3 get_value()	52
6.10.3.4 get_values() [1/4]	53
6.10.3.5 get_values() [2/4]	53
6.10.3.6 get_values() [3/4]	53
6.10.3.7 get_values() [4/4]	54
6.10.3.8 num_lines()	54
6.10.3.9 offsets()	54
6.10.3.10 operator bool()	55
6.10.3.11 operator=()	55

6.10.3.12 read_edge_events() [1/2]	55
6.10.3.13 read_edge_events() [2/2]	56
6.10.3.14 reconfigure_lines()	56
6.10.3.15 release()	56
6.10.3.16 set_value()	57
6.10.3.17 set_values() [1/3]	57
6.10.3.18 set_values() [2/3]	57
6.10.3.19 set_values() [3/3]	58
6.10.3.20 wait_edge_events()	58
6.11 gpiod::line_settings Class Reference	58
6.11.1 Detailed Description	60
6.11.2 Constructor & Destructor Documentation	60
6.11.2.1 line_settings() [1/2]	60
6.11.2.2 line_settings() [2/2]	60
6.11.3 Member Function Documentation	60
6.11.3.1 active_low()	60
6.11.3.2 bias()	61
6.11.3.3 debounce_period()	61
6.11.3.4 direction()	61
6.11.3.5 drive()	61
6.11.3.6 edge_detection()	62
6.11.3.7 event_clock()	62
6.11.3.8 operator=() [1/2]	62
6.11.3.9 operator=() [2/2]	62
6.11.3.10 output_value()	63
6.11.3.11 reset()	63
6.11.3.12 set_active_low()	63
6.11.3.13 set_bias()	64
6.11.3.14 set_debounce_period()	64
6.11.3.15 set_direction()	64
6.11.3.16 set_drive()	65
6.11.3.17 set_edge_detection()	65
6.11.3.18 set_event_clock()	65
6.11.3.19 set_output_value()	67
6.12 gpiod::line::offset Class Reference	67
6.12.1 Detailed Description	68
6.12.2 Constructor & Destructor Documentation	68
6.12.2.1 offset() [1/3]	68
6.12.2.2 offset() [2/3]	68
6.12.2.3 offset() [3/3]	68
6.12.3 Member Function Documentation	69
6.12.3.1 operator=() [1/2]	69

6.12.3.2 operator=() [2/2]	69
6.13 gpiod::request_builder Class Reference	70
6.13.1 Detailed Description	70
6.13.2 Constructor & Destructor Documentation	71
6.13.2.1 request_builder()	71
6.13.3 Member Function Documentation	71
6.13.3.1 add_line_settings() [1/2]	71
6.13.3.2 add_line_settings() [2/2]	71
6.13.3.3 do_request()	72
6.13.3.4 get_line_config()	72
6.13.3.5 get_request_config()	72
6.13.3.6 operator=()	72
6.13.3.7 set_consumer()	73
6.13.3.8 set_event_buffer_size()	73
6.13.3.9 set_line_config()	73
6.13.3.10 set_output_values()	75
6.13.3.11 set_request_config()	75
6.14 gpiod::request_config Class Reference	76
6.14.1 Detailed Description	76
6.14.2 Constructor & Destructor Documentation	76
6.14.2.1 request_config()	76
6.14.3 Member Function Documentation	77
6.14.3.1 consumer()	77
6.14.3.2 event_buffer_size()	77
6.14.3.3 operator=()	77
6.14.3.4 set_consumer()	78
6.14.3.5 set_event_buffer_size()	78
6.15 gpiod::request_released Class Reference	79
6.15.1 Detailed Description	79
6.15.2 Constructor & Destructor Documentation	79
6.15.2.1 request_released() [1/3]	79
6.15.2.2 request_released() [2/3]	80
6.15.2.3 request_released() [3/3]	80
6.15.3 Member Function Documentation	80
6.15.3.1 operator=() [1/2]	80
6.15.3.2 operator=() [2/2]	81
6.16 gpiod::timestamp Class Reference	81
6.16.1 Detailed Description	82
6.16.2 Constructor & Destructor Documentation	82
6.16.2.1 timestamp() [1/3]	82
6.16.2.2 timestamp() [2/3]	82
6.16.2.3 timestamp() [3/3]	82

6.16.3 Member Function Documentation	83
6.16.3.1 ns()	83
6.16.3.2 operator=() [1/2]	83
6.16.3.3 operator=() [2/2]	83
6.16.3.4 to_time_point_monotonic()	84
6.16.3.5 to_time_point_realtime()	84
7 File Documentation	85
7.1 gpiod.hpp File Reference	85
7.2 gpiod.hpp	85
7.3 gpiodcxx/chip-info.hpp File Reference	86
7.3.1 Function Documentation	86
7.3.1.1 operator<<()	86
7.4 chip-info.hpp	86
7.5 gpiodcxx/chip.hpp File Reference	87
7.5.1 Function Documentation	88
7.5.1.1 operator<<()	88
7.6 chip.hpp	88
7.7 gpiodcxx/edge-event-buffer.hpp File Reference	89
7.7.1 Function Documentation	89
7.7.1.1 operator<<()	90
7.8 edge-event-buffer.hpp	90
7.9 gpiodcxx/edge-event.hpp File Reference	91
7.9.1 Function Documentation	91
7.9.1.1 operator<<()	91
7.10 edge-event.hpp	92
7.11 gpiodcxx/exception.hpp File Reference	93
7.12 exception.hpp	93
7.13 info-event.hpp	94
7.14 gpiodcxx/line-config.hpp File Reference	95
7.14.1 Function Documentation	95
7.14.1.1 operator<<()	95
7.15 line-config.hpp	95
7.16 gpiodcxx/line-info.hpp File Reference	96
7.16.1 Function Documentation	97
7.16.1.1 operator<<()	97
7.17 line-info.hpp	97
7.18 gpiodcxx/line-request.hpp File Reference	98
7.18.1 Function Documentation	98
7.18.1.1 operator<<()	98
7.19 line-request.hpp	99
7.20 line-settings.hpp	100

7.21 gpiodcxx/line.hpp File Reference	101
7.22 line.hpp	103
7.23 gpiodcxx/misc.hpp File Reference	104
7.23.1 Function Documentation	104
7.23.1.1 api_version()	104
7.23.1.2 is_gpiochip_device()	105
7.24 misc.hpp	105
7.25 gpiodcxx/request-builder.hpp File Reference	105
7.25.1 Function Documentation	106
7.25.1.1 operator<<()	106
7.26 request-builder.hpp	106
7.27 gpiodcxx/request-config.hpp File Reference	107
7.27.1 Function Documentation	107
7.27.1.1 operator<<()	107
7.28 request-config.hpp	108
7.29 gpiodcxx/timestamp.hpp File Reference	109
7.30 timestamp.hpp	109
Index	111

Chapter 1

Namespace Index

1.1 Namespace List

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

gpiod::line	Namespace containing various type definitions for GPIO lines	9
-----------------------------	--	---

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

gpiod::chip	19
gpiod::chip_info	27
gpiod::edge_event	30
gpiod::edge_event_buffer	34
gpiod::info_event	38
gpiod::line_config	42
gpiod::line_info	45
gpiod::line_request	50
gpiod::line_settings	58
std::logic_error	
gpiod::chip_closed	25
gpiod::request_released	79
gpiod::line::offset	67
gpiod::request_builder	70
gpiod::request_config	76
std::runtime_error	
gpiod::bad_mapping	17
gpiod::timestamp	81

Chapter 3

Class Index

3.1 Class List

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

gpiod::bad_mapping	Exception thrown when the core C library returns an invalid value for any of the line_info properties	17
gpiod::chip	Represents a GPIO chip	19
gpiod::chip_closed	Exception thrown when an already closed chip is used	25
gpiod::chip_info	Represents an immutable snapshot of GPIO chip information	27
gpiod::edge_event	Immutable object containing data about a single edge event	30
gpiod::edge_event_buffer	Object into which edge events are read for better performance	34
gpiod::info_event	Immutable object containing data about a single line info event	38
gpiod::line_config	Contains a set of line config options used in line requests and reconfiguration	42
gpiod::line_info	Contains an immutable snapshot of the line's state at the time when the object of this class was instantiated	45
gpiod::line_request	Stores the context of a set of requested GPIO lines	50
gpiod::line_settings	Stores GPIO line settings	58
gpiod::line::offset	Wrapper around unsigned int for representing line offsets	67
gpiod::request_builder	Intermediate object storing the configuration for a line request	70
gpiod::request_config	Stores a set of options passed to the kernel when making a line request	76
gpiod::request_released	Exception thrown when an already released line request is used	79
gpiod::timestamp	Stores the edge and info event timestamps as returned by the kernel and allows to convert them to std::chrono::time_point	81

Chapter 4

File Index

4.1 File List

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

gpiod.hpp	85
gpiodcxx/chip-info.hpp	86
gpiodcxx/chip.hpp	87
gpiodcxx/edge-event-buffer.hpp	89
gpiodcxx/edge-event.hpp	91
gpiodcxx/exception.hpp	93
gpiodcxx/info-event.hpp	94
gpiodcxx/line-config.hpp	95
gpiodcxx/line-info.hpp	96
gpiodcxx/line-request.hpp	98
gpiodcxx/line-settings.hpp	100
gpiodcxx/line.hpp	101
gpiodcxx/misc.hpp	104
gpiodcxx/request-builder.hpp	105
gpiodcxx/request-config.hpp	107
gpiodcxx/timestamp.hpp	109

Chapter 5

Namespace Documentation

5.1 gpiod::line Namespace Reference

Namespace containing various type definitions for GPIO lines.

Classes

- class [offset](#)
Wrapper around unsigned int for representing line offsets.

Typedefs

- using **offsets** = ::std::vector< [offset](#) >
Vector of line offsets.
- using **values** = ::std::vector< [value](#) >
Vector of line values.
- using **value_mapping** = ::std::pair< [offset](#), [value](#) >
Represents a mapping of a line offset to line logical state.
- using **value_mappings** = ::std::vector< [value_mapping](#) >
Vector of offset->value mappings. Each mapping is defined as a pair of an unsigned and signed integers.

Enumerations

- enum class [value](#) { [INACTIVE](#) = 0 , [ACTIVE](#) = 1 }
Logical line states.
- enum class [direction](#) { [AS_IS](#) = 1 , [INPUT](#) , [OUTPUT](#) }
Direction settings.
- enum class [edge](#) { [NONE](#) = 1 , [RISING](#) , [FALLING](#) , [BOTH](#) }
Edge detection settings.
- enum class [bias](#) {
 [AS_IS](#) = 1 , [UNKNOWN](#) , [DISABLED](#) , [PULL_UP](#) ,
 [PULL_DOWN](#) }
Internal bias settings.
- enum class [drive](#) { [PUSH_PULL](#) = 1 , [OPEN_DRAIN](#) , [OPEN_SOURCE](#) }
Drive settings.
- enum class [clock](#) { [MONOTONIC](#) = 1 , [REALTIME](#) , [HTE](#) }
Event clock settings.

Functions

- `::std::ostream & operator<< (::std::ostream &out, value val)`
Stream insertion operator for logical line values.
- `::std::ostream & operator<< (::std::ostream &out, direction dir)`
Stream insertion operator for direction values.
- `::std::ostream & operator<< (::std::ostream &out, edge edge)`
Stream insertion operator for edge detection values.
- `::std::ostream & operator<< (::std::ostream &out, bias bias)`
Stream insertion operator for bias values.
- `::std::ostream & operator<< (::std::ostream &out, drive drive)`
Stream insertion operator for drive values.
- `::std::ostream & operator<< (::std::ostream &out, clock clock)`
Stream insertion operator for event clock values.
- `::std::ostream & operator<< (::std::ostream &out, const values &vals)`
Stream insertion operator for the list of output values.
- `::std::ostream & operator<< (::std::ostream &out, const offsets &offs)`
Stream insertion operator for the list of line offsets.
- `::std::ostream & operator<< (::std::ostream &out, const value_mapping &mapping)`
Stream insertion operator for the offset-to-value mapping.
- `::std::ostream & operator<< (::std::ostream &out, const value_mappings &mappings)`
Stream insertion operator for the list of offset-to-value mappings.

5.1.1 Detailed Description

Namespace containing various type definitions for GPIO lines.

5.1.2 Enumeration Type Documentation

5.1.2.1 bias

```
enum class gpio::line::bias [strong]
```

Internal bias settings.

Enumerator

AS_IS	Don't change the bias setting when applying line config.
UNKNOWN	The internal bias state is unknown.
DISABLED	The internal bias is disabled.
PULL_UP	The internal pull-up bias is enabled.
PULL_DOWN	The internal pull-down bias is enabled.

5.1.2.2 clock

```
enum class gpiod::line::clock [strong]
```

Event clock settings.

Enumerator

MONOTONIC	Line uses the monotonic clock for edge event timestamps.
REALTIME	Line uses the realtime clock for edge event timestamps.

5.1.2.3 direction

```
enum class gpiod::line::direction [strong]
```

Direction settings.

Enumerator

AS_IS	Request the line(s), but don't change current direction.
INPUT	Direction is input - we're reading the state of a GPIO line.
OUTPUT	Direction is output - we're driving the GPIO line.

5.1.2.4 drive

```
enum class gpiod::line::drive [strong]
```

Drive settings.

Enumerator

PUSH_PULL	Drive setting is push-pull.
OPEN_DRAIN	Line output is open-drain.
OPEN_SOURCE	Line output is open-source.

5.1.2.5 edge

```
enum class gpiod::line::edge [strong]
```

Edge detection settings.

Enumerator

NONE	Line edge detection is disabled.
RISING	Line detects rising edge events.
FALLING	Line detect falling edge events.
BOTH	Line detects both rising and falling edge events.

5.1.2.6 value

```
enum class gpiod::line::value [strong]
```

Logical line states.

Enumerator

INACTIVE	Line is inactive.
ACTIVE	Line is active.

5.1.3 Function Documentation**5.1.3.1 operator<<() [1/10]**

```
::std::ostream & gpiod::line::operator<< (
    ::std::ostream & out,
    bias bias )
```

Stream insertion operator for bias values.

Parameters

<i>out</i>	Output stream.
<i>bias</i>	Value to insert into the output stream in a human-readable form.

Returns

Reference to out.

5.1.3.2 operator<<() [2/10]

```
::std::ostream & gpiod::line::operator<< (
    ::std::ostream & out,
    clock clock )
```

Stream insertion operator for event clock values.

Parameters

<i>out</i>	Output stream.
<i>clock</i>	Value to insert into the output stream in a human-readable form.

Returns

Reference to out.

5.1.3.3 operator<<() [3/10]

```
::std::ostream & gpiod::line::operator<< (
    ::std::ostream & out,
    const offsets & offs )
```

Stream insertion operator for the list of line offsets.

Parameters

<i>out</i>	Output stream.
<i>offs</i>	Object to insert into the output stream in a human-readable form.

Returns

Reference to out.

5.1.3.4 operator<<() [4/10]

```
::std::ostream & gpiod::line::operator<< (
    ::std::ostream & out,
    const value_mapping & mapping )
```

Stream insertion operator for the offset-to-value mapping.

Parameters

<i>out</i>	Output stream.
<i>mapping</i>	Value to insert into the output stream in a human-readable form.

Returns

Reference to out.

5.1.3.5 `operator<<()` [5/10]

```
::std::ostream & gpiod::line::operator<< (
    ::std::ostream & out,
    const value_mappings & mappings )
```

Stream insertion operator for the list of offset-to-value mappings.

Parameters

<i>out</i>	Output stream.
<i>mappings</i>	Object to insert into the output stream in a human-readable form.

Returns

Reference to out.

5.1.3.6 `operator<<()` [6/10]

```
::std::ostream & gpiod::line::operator<< (
    ::std::ostream & out,
    const values & vals )
```

Stream insertion operator for the list of output values.

Parameters

<i>out</i>	Output stream.
<i>vals</i>	Object to insert into the output stream in a human-readable form.

Returns

Reference to out.

5.1.3.7 `operator<<()` [7/10]

```
::std::ostream & gpiod::line::operator<< (
    ::std::ostream & out,
    direction dir )
```

Stream insertion operator for direction values.

Parameters

<i>out</i>	Output stream.
<i>dir</i>	Value to insert into the output stream in a human-readable form.

Returns

Reference to out.

5.1.3.8 operator<<() [8/10]

```
::std::ostream & gpiod::line::operator<< (
    ::std::ostream & out,
    drive drive )
```

Stream insertion operator for drive values.

Parameters

<i>out</i>	Output stream.
<i>drive</i>	Value to insert into the output stream in a human-readable form.

Returns

Reference to out.

5.1.3.9 operator<<() [9/10]

```
::std::ostream & gpiod::line::operator<< (
    ::std::ostream & out,
    edge edge )
```

Stream insertion operator for edge detection values.

Parameters

<i>out</i>	Output stream.
<i>edge</i>	Value to insert into the output stream in a human-readable form.

Returns

Reference to out.

5.1.3.10 operator<<() [10/10]

```
::std::ostream & gpiod::line::operator<< (  
    ::std::ostream & out,  
    value val )
```

Stream insertion operator for logical line values.

Parameters

<i>out</i>	Output stream.
<i>val</i>	Value to insert into the output stream in a human-readable form.

Returns

Reference to out.

Chapter 6

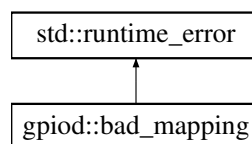
Class Documentation

6.1 gpiod::bad_mapping Class Reference

Exception thrown when the core C library returns an invalid value for any of the [line_info](#) properties.

```
#include <exception.hpp>
```

Inheritance diagram for gpiod::bad_mapping:



Public Member Functions

- [bad_mapping](#) (const ::std::string &what)
Constructor.
- [bad_mapping](#) (const [bad_mapping](#) &other) noexcept
Copy constructor.
- [bad_mapping](#) ([bad_mapping](#) &&other) noexcept
Move constructor.
- [bad_mapping](#) & [operator=](#) (const [bad_mapping](#) &other) noexcept
Assignment operator.
- [bad_mapping](#) & [operator=](#) ([bad_mapping](#) &&other) noexcept
Move assignment operator.

6.1.1 Detailed Description

Exception thrown when the core C library returns an invalid value for any of the [line_info](#) properties.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `bad_mapping()` [1/3]

```
gpiod::bad_mapping::bad_mapping (
    const ::std::string & what ) [explicit]
```

Constructor.

Parameters

<i>what</i>	Human readable reason for error.
-------------	----------------------------------

6.1.2.2 `bad_mapping()` [2/3]

```
gpiod::bad_mapping::bad_mapping (
    const bad_mapping & other ) [noexcept]
```

Copy constructor.

Parameters

<i>other</i>	Object to copy from.
--------------	----------------------

6.1.2.3 `bad_mapping()` [3/3]

```
gpiod::bad_mapping::bad_mapping (
    bad_mapping && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.1.3 Member Function Documentation

6.1.3.1 operator=() [1/2]

```
bad_mapping & gpiod::bad_mapping::operator= (
    bad_mapping && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.1.3.2 operator=() [2/2]

```
bad_mapping & gpiod::bad_mapping::operator= (
    const bad_mapping & other ) [noexcept]
```

Assignment operator.

Parameters

<i>other</i>	Object to copy from.
--------------	----------------------

Returns

Reference to self.

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

- [gpiodcxx/exception.hpp](#)

6.2 gpiod::chip Class Reference

Represents a GPIO chip.

```
#include <chip.hpp>
```

Public Member Functions

- [chip](#) (const ::std::filesystem::path &[path](#))
Instantiates a new chip object by opening the device file indicated by the `path` argument.
- [chip](#) ([chip](#) &&other) noexcept
Move constructor.
- [chip](#) & [operator=](#) (const [chip](#) &other)=delete
- [chip](#) & [operator=](#) ([chip](#) &&other) noexcept
Move assignment operator.
- [operator bool](#) () const noexcept
Check if this object is valid.
- void [close](#) ()
Close the GPIO chip device file and free associated resources.
- ::std::filesystem::path [path](#) () const
Get the filesystem path that was used to open this GPIO chip.
- [chip_info](#) [get_info](#) () const
Get information about the chip.
- [line_info](#) [get_line_info](#) ([line::offset](#) offset) const
Retrieve the current snapshot of line information for a single line.
- [line_info](#) [watch_line_info](#) ([line::offset](#) offset) const
Wrapper around `gpiod::chip::get_line_info` that retrieves the line info and starts watching the line for changes.
- void [unwatch_line_info](#) ([line::offset](#) offset) const
Stop watching the line at given offset for info events.
- int [fd](#) () const
Get the file descriptor associated with this chip.
- bool [wait_info_event](#) (const ::std::chrono::nanoseconds &timeout) const
Wait for line status events on any of the watched lines exposed by this chip.
- [info_event](#) [read_info_event](#) () const
Read a single line status change event from this chip.
- int [get_line_offset_from_name](#) (const ::std::string &name) const
Map a GPIO line's name to its offset within the chip.
- [request_builder](#) [prepare_request](#) ()
Create a [request_builder](#) associated with this chip.

Private Member Functions

- [chip](#) (const [chip](#) &other)

Private Attributes

- ::std::shared_ptr< impl > [_m_priv](#)
- friend [request_builder](#)

6.2.1 Detailed Description

Represents a GPIO chip.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 chip() [1/2]

```
gpiod::chip::chip (
    const ::std::filesystem::path & path ) [explicit]
```

Instantiates a new chip object by opening the device file indicated by the `path` argument.

Parameters

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

6.2.2.2 chip() [2/2]

```
gpiod::chip::chip (
    chip && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.2.3 Member Function Documentation

6.2.3.1 close()

```
void gpiod::chip::close ( )
```

Close the GPIO chip device file and free associated resources.

Note

The chip object can live after calling this method but any of the chip's mutators will throw a `logic_error` exception.

6.2.3.2 fd()

```
int gpiod::chip::fd ( ) const
```

Get the file descriptor associated with this chip.

Returns

File descriptor number.

6.2.3.3 get_info()

```
chip_info gpiod::chip::get_info ( ) const
```

Get information about the chip.

Returns

New [chip_info](#) object.

6.2.3.4 get_line_info()

```
line_info gpiod::chip::get_line_info (
    line::offset offset ) const
```

Retrieve the current snapshot of line information for a single line.

Parameters

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

Returns

New [gpiod::line_info](#) object.

6.2.3.5 get_line_offset_from_name()

```
int gpiod::chip::get_line_offset_from_name (
    const ::std::string & name ) const
```

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

Parameters

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

Returns

Offset of the line within the chip or -1 if the line with given name is not exposed by this chip.

6.2.3.6 operator bool()

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

Check if this object is valid.

Returns

True if this object's methods can be used, false otherwise. False usually means the chip was closed. If the user calls any of the methods of this class on an object for which this operator returned false, a `logic_error` will be thrown.

6.2.3.7 operator=()

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

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.2.3.8 path()

```
::std::filesystem::path gpiod::chip::path ( ) const
```

Get the filesystem path that was used to open this GPIO chip.

Returns

Path to the underlying character device file.

6.2.3.9 prepare_request()

```
request_builder gpiod::chip::prepare_request ( )
```

Create a [request_builder](#) associated with this chip.

Returns

New [request_builder](#) object.

6.2.3.10 read_info_event()

```
info_event gpiod::chip::read_info_event ( ) const
```

Read a single line status change event from this chip.

Returns

New [info_event](#) object.

6.2.3.11 unwatch_line_info()

```
void gpiod::chip::unwatch_line_info (
    line::offset offset ) const
```

Stop watching the line at given offset for info events.

Parameters

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

6.2.3.12 wait_info_event()

```
bool gpiod::chip::wait_info_event (
    const ::std::chrono::nanoseconds & timeout ) const
```

Wait for line status events on any of the watched lines exposed by this chip.

Parameters

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

Returns

True if at least one event is ready to be read. False if the wait timed out.

6.2.3.13 watch_line_info()

```
line_info gpiod::chip::watch_line_info (
    line::offset offset ) const
```

Wrapper around [gpiod::chip::get_line_info](#) that retrieves the line info and starts watching the line for changes.

Parameters

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

Returns

New [gpiod::line_info](#) object.

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

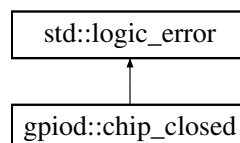
- [gpiodcxx/chip.hpp](#)

6.3 gpiod::chip_closed Class Reference

Exception thrown when an already closed chip is used.

```
#include <exception.hpp>
```

Inheritance diagram for `gpiod::chip_closed`:

**Public Member Functions**

- [chip_closed](#) (const ::std::string &what)
Constructor.
- [chip_closed](#) (const [chip_closed](#) &other) noexcept
Copy constructor.
- [chip_closed](#) ([chip_closed](#) &&other) noexcept
Move constructor.
- [chip_closed](#) & [operator=](#) (const [chip_closed](#) &other) noexcept
Assignment operator.
- [chip_closed](#) & [operator=](#) ([chip_closed](#) &&other) noexcept
Move assignment operator.

6.3.1 Detailed Description

Exception thrown when an already closed chip is used.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `chip_closed()` [1/3]

```
gpiod::chip_closed::chip_closed (
    const ::std::string & what ) [explicit]
```

Constructor.

Parameters

<i>what</i>	Human readable reason for error.
-------------	----------------------------------

6.3.2.2 `chip_closed()` [2/3]

```
gpiod::chip_closed::chip_closed (
    const chip_closed & other ) [noexcept]
```

Copy constructor.

Parameters

<i>other</i>	Object to copy from.
--------------	----------------------

6.3.2.3 `chip_closed()` [3/3]

```
gpiod::chip_closed::chip_closed (
    chip_closed && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.3.3 Member Function Documentation

6.3.3.1 operator=() [1/2]

```
chip_closed & gpiod::chip_closed::operator= (
    chip_closed && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.3.3.2 operator=() [2/2]

```
chip_closed & gpiod::chip_closed::operator= (
    const chip_closed & other ) [noexcept]
```

Assignment operator.

Parameters

<i>other</i>	Object to copy from.
--------------	----------------------

Returns

Reference to self.

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

- [gpiodcxx/exception.hpp](#)

6.4 gpiod::chip_info Class Reference

Represents an immutable snapshot of GPIO chip information.

```
#include <chip-info.hpp>
```

Public Member Functions

- `chip_info` (const `chip_info` &other)
Copy constructor.
- `chip_info` (`chip_info` &&other) noexcept
Move constructor.
- `chip_info` & `operator=` (const `chip_info` &other)
Assignment operator.
- `chip_info` & `operator=` (`chip_info` &&other) noexcept
Move assignment operator.
- `::std::string name` () const noexcept
Get the name of this GPIO chip.
- `::std::string label` () const noexcept
Get the label of this GPIO chip.
- `::std::size_t num_lines` () const noexcept
Return the number of lines exposed by this chip.

Private Attributes

- `::std::shared_ptr< impl > _m_priv`
- friend `chip`

6.4.1 Detailed Description

Represents an immutable snapshot of GPIO chip information.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 `chip_info()` [1/2]

```
gpiod::chip_info::chip_info (
    const chip_info & other )
```

Copy constructor.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

6.4.2.2 `chip_info()` [2/2]

```
gpiod::chip_info::chip_info (
    chip_info && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.4.3 Member Function Documentation

6.4.3.1 label()

```
::std::string gpiod::chip_info::label ( ) const [noexcept]
```

Get the label of this GPIO chip.

Returns

String containing the chip name.

6.4.3.2 name()

```
::std::string gpiod::chip_info::name ( ) const [noexcept]
```

Get the name of this GPIO chip.

Returns

String containing the chip name.

6.4.3.3 num_lines()

```
::std::size_t gpiod::chip_info::num_lines ( ) const [noexcept]
```

Return the number of lines exposed by this chip.

Returns

Number of lines.

6.4.3.4 operator=() [1/2]

```
chip_info & gpiod::chip_info::operator= (
    chip_info && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.4.3.5 operator=() [2/2]

```
chip_info & gpiod::chip_info::operator= (
    const chip_info & other )
```

Assignment operator.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

Returns

Reference to self.

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

- gpiodcxx/[chip-info.hpp](#)

6.5 gpiod::edge_event Class Reference

Immutable object containing data about a single edge event.

```
#include <edge-event.hpp>
```

Public Types

- enum class [event_type](#) { [RISING_EDGE](#) = 1 , [FALLING_EDGE](#) }
Edge event types.

Public Member Functions

- `edge_event` (const `edge_event` &other)
Copy constructor.
- `edge_event` (`edge_event` &&other) noexcept
Move constructor.
- `edge_event` & `operator=` (const `edge_event` &other)
Copy assignment operator.
- `edge_event` & `operator=` (`edge_event` &&other) noexcept
Move assignment operator.
- `event_type` type () const
Retrieve the event type.
- `timestamp` `timestamp_ns` () const noexcept
Retrieve the event time-stamp.
- `line::offset` `line_offset` () const noexcept
Read the offset of the line on which this event was registered.
- unsigned long `global_seqno` () const noexcept
Get the global sequence number of this event.
- unsigned long `line_seqno` () const noexcept
Get the event sequence number specific to the concerned line.

Private Attributes

- `::std::shared_ptr< impl > _m_priv`
- friend `edge_event_buffer`

6.5.1 Detailed Description

Immutable object containing data about a single edge event.

6.5.2 Member Enumeration Documentation

6.5.2.1 event_type

```
enum class gpiod::edge_event::event_type [strong]
```

Edge event types.

Enumerator

RISING_EDGE	This is a rising edge event.
FALLING_EDGE	This is falling edge event.

6.5.3 Constructor & Destructor Documentation

6.5.3.1 `edge_event()` [1/2]

```
gpiod::edge_event::edge_event (
    const edge_event & other )
```

Copy constructor.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

6.5.3.2 `edge_event()` [2/2]

```
gpiod::edge_event::edge_event (
    edge_event && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.5.4 Member Function Documentation

6.5.4.1 `global_seqno()`

```
unsigned long gpiod::edge_event::global_seqno ( ) const [noexcept]
```

Get the global sequence number of this event.

Returns

Sequence number of the event relative to all lines in the associated line request.

6.5.4.2 line_offset()

```
line::offset gpiod::edge_event::line_offset ( ) const [noexcept]
```

Read the offset of the line on which this event was registered.

Returns

Line offset.

6.5.4.3 line_seqno()

```
unsigned long gpiod::edge_event::line_seqno ( ) const [noexcept]
```

Get the event sequence number specific to the concerned line.

Returns

Sequence number of the event relative to this line within the lifetime of the associated line request.

6.5.4.4 operator=() [1/2]

```
edge_event & gpiod::edge_event::operator= (
    const edge_event & other )
```

Copy assignment operator.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

Returns

Reference to self.

6.5.4.5 operator=() [2/2]

```
edge_event & gpiod::edge_event::operator= (
    edge_event && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.5.4.6 timestamp_ns()

```
timestamp gpiod::edge_event::timestamp_ns ( ) const [noexcept]
```

Retrieve the event time-stamp.

Returns

Time-stamp in nanoseconds as registered by the kernel using the configured edge event clock.

6.5.4.7 type()

```
event_type gpiod::edge_event::type ( ) const
```

Retrieve the event type.

Returns

Event type (rising or falling edge).

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

- [gpiodcxx/edge-event.hpp](#)

6.6 gpiod::edge_event_buffer Class Reference

Object into which edge events are read for better performance.

```
#include <edge-event-buffer.hpp>
```

Public Types

- using **const_iterator** = ::std::vector< [edge_event](#) >::const_iterator
Constant iterator for iterating over edge events stored in the buffer.

Public Member Functions

- [edge_event_buffer](#) (::std::size_t capacity=64)
Constructor. Creates a new edge event buffer with given capacity.
- [edge_event_buffer](#) (const [edge_event_buffer](#) &other)=delete
- [edge_event_buffer](#) ([edge_event_buffer](#) &&other) noexcept
Move constructor.
- [edge_event_buffer](#) & [operator=](#) (const [edge_event_buffer](#) &other)=delete
- [edge_event_buffer](#) & [operator=](#) ([edge_event_buffer](#) &&other) noexcept
Move assignment operator.
- const [edge_event](#) & [get_event](#) (unsigned int index) const
Get the constant reference to the edge event at given index.
- ::std::size_t [num_events](#) () const
Get the number of edge events currently stored in the buffer.
- ::std::size_t [capacity](#) () const noexcept
Maximum capacity of the buffer.
- [const_iterator](#) [begin](#) () const noexcept
Get a constant iterator to the first edge event currently stored in the buffer.
- [const_iterator](#) [end](#) () const noexcept
Get a constant iterator to the element after the last edge event in the buffer.

Private Attributes

- ::std::unique_ptr< impl > [_m_priv](#)
- friend [line_request](#)

6.6.1 Detailed Description

Object into which edge events are read for better performance.

The [edge_event_buffer](#) allows reading [edge_event](#) objects into an existing buffer which improves the performance by avoiding needless memory allocations.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 [edge_event_buffer](#)() [1/2]

```
gpiod::edge_event_buffer::edge_event_buffer (
    ::std::size_t capacity = 64 ) [explicit]
```

Constructor. Creates a new edge event buffer with given capacity.

Parameters

<i>capacity</i>	Capacity of the new buffer.
-----------------	-----------------------------

6.6.2.2 edge_event_buffer() [2/2]

```
gpiod::edge_event_buffer::edge_event_buffer (
    edge_event_buffer && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.6.3 Member Function Documentation

6.6.3.1 begin()

```
const_iterator gpiod::edge_event_buffer::begin ( ) const [noexcept]
```

Get a constant iterator to the first edge event currently stored in the buffer.

Returns

Constant iterator to the first element.

6.6.3.2 capacity()

```
::std::size_t gpiod::edge_event_buffer::capacity ( ) const [noexcept]
```

Maximum capacity of the buffer.

Returns

Buffer capacity.

6.6.3.3 end()

```
const_iterator gpiod::edge_event_buffer::end ( ) const [noexcept]
```

Get a constant iterator to the element after the last edge event in the buffer.

Returns

Constant iterator to the element after the last edge event.

6.6.3.4 get_event()

```
const edge\_event & gpiod::edge_event_buffer::get_event (
    unsigned int index ) const
```

Get the constant reference to the edge event at given index.

Parameters

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

Returns

Constant reference to the edge event.

6.6.3.5 num_events()

```
::std::size_t gpiod::edge_event_buffer::num_events ( ) const
```

Get the number of edge events currently stored in the buffer.

Returns

Number of edge events in the buffer.

6.6.3.6 operator=()

```
edge_event_buffer & gpiod::edge_event_buffer::operator= (
    edge_event_buffer && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

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

- [gpiodcxx/edge-event-buffer.hpp](#)

6.7 gpiod::info_event Class Reference

Immutable object containing data about a single line info event.

```
#include <info-event.hpp>
```


Public Types

- enum class `event_type` { `LINE_REQUESTED` = 1 , `LINE_RELEASED` , `LINE_CONFIG_CHANGED` }
Types of info events.

Public Member Functions

- `info_event` (const `info_event` &other)
Copy constructor.
- `info_event` (`info_event` &&other) noexcept
Move constructor.
- `info_event` & `operator=` (const `info_event` &other)
Copy assignment operator.
- `info_event` & `operator=` (`info_event` &&other) noexcept
Move assignment operator.
- `event_type type` () const
Type of this event.
- `::std::uint64_t timestamp_ns` () const noexcept
Timestamp of the event as returned by the kernel.
- const `line_info` & `get_line_info` () const noexcept
Get the new line information.

Private Attributes

- `::std::shared_ptr< impl > _m_priv`
- friend `chip`

6.7.1 Detailed Description

Immutable object containing data about a single line info event.

6.7.2 Member Enumeration Documentation

6.7.2.1 event_type

```
enum class gpiod::info_event::event_type [strong]
```

Types of info events.

Enumerator

<code>LINE_REQUESTED</code>	Line has been requested.
<code>LINE_RELEASED</code>	Previously requested line has been released.
<code>LINE_CONFIG_CHANGED</code>	Line configuration has changed.

6.7.3 Constructor & Destructor Documentation

6.7.3.1 info_event() [1/2]

```
gpiod::info_event::info_event (
    const info_event & other )
```

Copy constructor.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

6.7.3.2 info_event() [2/2]

```
gpiod::info_event::info_event (
    info_event && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.7.4 Member Function Documentation

6.7.4.1 get_line_info()

```
const line_info & gpiod::info_event::get_line_info ( ) const [noexcept]
```

Get the new line information.

Returns

Constant reference to the line info object containing the line data as read at the time of the info event.

6.7.4.2 operator=() [1/2]

```
info_event & gpiod::info_event::operator= (
    const info_event & other )
```

Copy assignment operator.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

Returns

Reference to self.

6.7.4.3 operator=() [2/2]

```
info_event & gpiod::info_event::operator= (  
    info_event && other )    [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.7.4.4 timestamp_ns()

```
::std::uint64_t gpiod::info_event::timestamp_ns ( ) const    [noexcept]
```

Timestamp of the event as returned by the kernel.

Returns

Timestamp as a 64-bit unsigned integer.

6.7.4.5 type()

```
event_type gpiod::info_event::type ( ) const
```

Type of this event.

Returns

Event type.

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

- gpiodcxx/info-event.hpp

6.8 gpiod::line_config Class Reference

Contains a set of line config options used in line requests and reconfiguration.

```
#include <line-config.hpp>
```

Public Member Functions

- **line_config** (const [line_config](#) &other)=delete
- **line_config** ([line_config](#) &&other) noexcept
Move constructor.
- **line_config** & **operator=** ([line_config](#) &&other) noexcept
Move assignment operator.
- **line_config** & **reset** () noexcept
Reset the line config object.
- **line_config** & **add_line_settings** ([line::offset](#) offset, const [line_settings](#) &settings)
Add line settings for a single offset.
- **line_config** & **add_line_settings** (const [line::offsets](#) &offsets, const [line_settings](#) &settings)
Add line settings for a set of offsets.
- **line_config** & **set_output_values** (const [line::values](#) &values)
Set output values for a number of lines.
- **::std::map**< [line::offset](#), [line_settings](#) > **get_line_settings** () const
Get a mapping of offsets to line settings stored by this object.

Private Member Functions

- **line_config** & **operator=** (const [line_config](#) &other)

Private Attributes

- **::std::shared_ptr**< impl > **_m_priv**
- friend **line_request**
- friend **request_builder**

6.8.1 Detailed Description

Contains a set of line config options used in line requests and reconfiguration.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 line_config()

```
gpiod::line_config::line_config (
    line\_config && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.8.3 Member Function Documentation

6.8.3.1 add_line_settings() [1/2]

```
line_config & gpiod::line_config::add_line_settings (
    const line::offsets & offsets,
    const line_settings & settings )
```

Add line settings for a set of offsets.

Parameters

<i>offsets</i>	Offsets for which to add settings.
<i>settings</i>	Line settings to add.

Returns

Reference to self.

6.8.3.2 add_line_settings() [2/2]

```
line_config & gpiod::line_config::add_line_settings (
    line::offset offset,
    const line_settings & settings )
```

Add line settings for a single offset.

Parameters

<i>offset</i>	Offset for which to add settings.
<i>settings</i>	Line settings to add.

Returns

Reference to self.

6.8.3.3 get_line_settings()

```
::std::map< line::offset, line_settings > gpiod::line_config::get_line_settings ( ) const
```

Get a mapping of offsets to line settings stored by this object.

Returns

Map in which keys represent line offsets and values are the settings corresponding with them.

6.8.3.4 operator=()

```
line_config & gpiod::line_config::operator= (
    line_config && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.8.3.5 reset()

```
line_config & gpiod::line_config::reset ( ) [noexcept]
```

Reset the line config object.

Returns

Reference to self.

6.8.3.6 set_output_values()

```
line_config & gpiod::line_config::set_output_values (
    const line::values & values )
```

Set output values for a number of lines.

Parameters

<i>values</i>	Buffer containing the output values.
---------------	--------------------------------------

Returns

Reference to self.

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

- [gpiodcxx/line-config.hpp](#)

6.9 gpiod::line_info Class Reference

Contains an immutable snapshot of the line's state at the time when the object of this class was instantiated.

```
#include <line-info.hpp>
```

Public Member Functions

- [line_info](#) (const [line_info](#) &other) noexcept
Copy constructor.
- [line_info](#) ([line_info](#) &&other) noexcept
Move constructor.
- [line_info](#) & [operator=](#) (const [line_info](#) &other) noexcept
Copy assignment operator.
- [line_info](#) & [operator=](#) ([line_info](#) &&other) noexcept
Move assignment operator.
- [line::offset](#) [offset](#) () const noexcept
Get the hardware offset of the line.
- [::std::string](#) [name](#) () const noexcept
Get the GPIO line name.
- bool [used](#) () const noexcept
Check if the line is currently in use.
- [::std::string](#) [consumer](#) () const noexcept
Read the GPIO line consumer name.
- [line::direction](#) [direction](#) () const
Read the GPIO line direction setting.
- [line::edge](#) [edge_detection](#) () const
Read the current edge detection setting of this line.
- [line::bias](#) [bias](#) () const
Read the GPIO line bias setting.
- [line::drive](#) [drive](#) () const
Read the GPIO line drive setting.
- bool [active_low](#) () const noexcept
Check if the signal of this line is inverted.
- bool [debounced](#) () const noexcept
Check if this line is debounced (either by hardware or by the kernel software debouncer).
- [::std::chrono::microseconds](#) [debounce_period](#) () const noexcept
Read the current debounce period in microseconds.
- [line::clock](#) [event_clock](#) () const
Read the current event clock setting used for edge event timestamps.

Private Attributes

- `::std::shared_ptr< impl > _m_priv`
- friend `chip`
- friend `info_event`

6.9.1 Detailed Description

Contains an immutable snapshot of the line's state at the time when the object of this class was instantiated.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 `line_info()` [1/2]

```
gpiod::line_info::line_info (  
    const line_info & other ) [noexcept]
```

Copy constructor.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

6.9.2.2 `line_info()` [2/2]

```
gpiod::line_info::line_info (  
    line_info && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.9.3 Member Function Documentation

6.9.3.1 `active_low()`

```
bool gpiod::line_info::active_low ( ) const [noexcept]
```


Check if the signal of this line is inverted.

Returns

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

6.9.3.2 bias()

```
line::bias gpiod::line_info::bias ( ) const
```

Read the GPIO line bias setting.

Returns

Returns BIAS_PULL_UP, BIAS_PULL_DOWN, BIAS_DISABLE or BIAS_UNKNOWN.

6.9.3.3 consumer()

```
::std::string gpiod::line_info::consumer ( ) const [noexcept]
```

Read the GPIO line consumer name.

Returns

Name of the GPIO consumer name as it is represented in the kernel. This routine returns an empty string if the line is not used.

6.9.3.4 debounce_period()

```
::std::chrono::microseconds gpiod::line_info::debounce_period ( ) const [noexcept]
```

Read the current debounce period in microseconds.

Returns

Current debounce period in microseconds, 0 if the line is not debounced.

6.9.3.5 debounced()

```
bool gpiod::line_info::debounced ( ) const [noexcept]
```

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

Returns

True if the line is debounced, false otherwise.

6.9.3.6 direction()

```
line::direction gpiod::line_info::direction ( ) const
```

Read the GPIO line direction setting.

Returns

Returns DIRECTION_INPUT or DIRECTION_OUTPUT.

6.9.3.7 drive()

```
line::drive gpiod::line_info::drive ( ) const
```

Read the GPIO line drive setting.

Returns

Returns DRIVE_PUSH_PULL, DRIVE_OPEN_DRAIN or DRIVE_OPEN_SOURCE.

6.9.3.8 edge_detection()

```
line::edge gpiod::line_info::edge_detection ( ) const
```

Read the current edge detection setting of this line.

Returns

Returns EDGE_NONE, EDGE_RISING, EDGE_FALLING or EDGE_BOTH.

6.9.3.9 event_clock()

```
line::clock gpiod::line_info::event_clock ( ) const
```

Read the current event clock setting used for edge event timestamps.

Returns

Returns MONOTONIC, REALTIME or HTE.

6.9.3.10 name()

```
::std::string gpiod::line_info::name ( ) const [noexcept]
```

Get the GPIO line name.

Returns

Name of the GPIO line as it is represented in the kernel. This routine returns an empty string if the line is unnamed.

6.9.3.11 offset()

```
line::offset gpiod::line_info::offset ( ) const [noexcept]
```

Get the hardware offset of the line.

Returns

Offset of the line within the parent chip.

6.9.3.12 operator=() [1/2]

```
line_info & gpiod::line_info::operator= (
    const line_info & other ) [noexcept]
```

Copy assignment operator.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

Returns

Reference to self.

6.9.3.13 operator=() [2/2]

```
line_info & gpiod::line_info::operator= (
    line_info && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.9.3.14 used()

```
bool gpiod::line_info::used ( ) const [noexcept]
```

Check if the line is currently in use.

Returns

True if the line is in use, false otherwise.

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

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

- [gpiodcxx/line-info.hpp](#)

6.10 gpiod::line_request Class Reference

Stores the context of a set of requested GPIO lines.

```
#include <line-request.hpp>
```

Public Member Functions

- **line_request** (const [line_request](#) &other)=delete
- **line_request** ([line_request](#) &&other) noexcept
Move constructor.
- **line_request** & **operator=** (const [line_request](#) &other)=delete
- **line_request** & **operator=** ([line_request](#) &&other) noexcept
Move assignment operator.
- **operator bool** () const noexcept
Check if this object is valid.
- void **release** ()
Release the requested lines and free all associated resources.
- ::std::string **chip_name** () const
Get the name of the chip this request was made on.
- ::std::size_t **num_lines** () const
Get the number of requested lines.
- [line::offsets](#) **offsets** () const
Get the list of offsets of requested lines.
- [line::value](#) **get_value** ([line::offset](#) offset)
Get the value of a single requested line.
- [line::values](#) **get_values** (const [line::offsets](#) &offsets)
Get the values of a subset of requested lines.
- [line::values](#) **get_values** ()
Get the values of all requested lines.
- void **get_values** (const [line::offsets](#) &offsets, [line::values](#) &values)
Get the values of a subset of requested lines into a vector supplied by the caller.
- void **get_values** ([line::values](#) &values)
Get the values of all requested lines.
- [line_request](#) & **set_value** ([line::offset](#) offset, [line::value](#) value)
Set the value of a single requested line.
- [line_request](#) & **set_values** (const [line::value_mappings](#) &values)
Set the values of a subset of requested lines.
- [line_request](#) & **set_values** (const [line::offsets](#) &offsets, const [line::values](#) &values)
Set the values of a subset of requested lines.
- [line_request](#) & **set_values** (const [line::values](#) &values)
Set the values of all requested lines.
- [line_request](#) & **reconfigure_lines** (const [line_config](#) &config)
Apply new config options to requested lines.
- int **fd** () const
Get the file descriptor number associated with this line request.
- bool **wait_edge_events** (const ::std::chrono::nanoseconds &timeout) const
Wait for edge events on any of the lines requested with edge detection enabled.
- ::std::size_t **read_edge_events** ([edge_event_buffer](#) &buffer)
Read a number of edge events from this request up to the maximum capacity of the buffer.
- ::std::size_t **read_edge_events** ([edge_event_buffer](#) &buffer, ::std::size_t max_events)
Read a number of edge events from this request.

Private Attributes

- ::std::unique_ptr< impl > **_m_priv**
- friend **request_builder**

6.10.1 Detailed Description

Stores the context of a set of requested GPIO lines.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 line_request()

```
gpiod::line_request::line_request (
    line_request && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.10.3 Member Function Documentation

6.10.3.1 chip_name()

```
::std::string gpiod::line_request::chip_name ( ) const
```

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

Returns

Name to the GPIO chip.

6.10.3.2 fd()

```
int gpiod::line_request::fd ( ) const
```

Get the file descriptor number associated with this line request.

Returns

File descriptor number.

6.10.3.3 get_value()

```
line::value gpiod::line_request::get_value (
    line::offset offset )
```

Get the value of a single requested line.

Parameters

<i>offset</i>	Offset of the line to read within the chip.
---------------	---

Returns

Current line value.

6.10.3.4 get_values() [1/4]

```
line::values gpiod::line_request::get_values ( )
```

Get the values of all requested lines.

Returns

List of read values.

6.10.3.5 get_values() [2/4]

```
line::values gpiod::line_request::get_values (
    const line::offsets & offsets )
```

Get the values of a subset of requested lines.

Parameters

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

Returns

Vector of lines values with indexes of values corresponding to those of the offsets.

6.10.3.6 get_values() [3/4]

```
void gpiod::line_request::get_values (
    const line::offsets & offsets,
    line::values & values )
```

Get the values of a subset of requested lines into a vector supplied by the caller.

Parameters

<i>offsets</i>	Vector of line offsets.
<i>values</i>	Vector for storing the values. Its size must be at least that of the offsets vector. The indexes of read values will correspond with those in the offsets vector.

6.10.3.7 `get_values()` [4/4]

```
void gpiod::line_request::get_values (
    line::values & values )
```

Get the values of all requested lines.

Parameters

<i>values</i>	Array in which the values will be stored. Must hold at least the number of elements returned by <code>line_request::num_lines</code> .
---------------	--

6.10.3.8 `num_lines()`

```
::std::size_t gpiod::line_request::num_lines ( ) const
```

Get the number of requested lines.

Returns

Number of lines in this request.

6.10.3.9 `offsets()`

```
line::offsets gpiod::line_request::offsets ( ) const
```

Get the list of offsets of requested lines.

Returns

List of hardware offsets of the lines in this request.

6.10.3.10 operator bool()

```
gpiod::line_request::operator bool ( ) const [explicit], [noexcept]
```

Check if this object is valid.

Returns

True if this object's methods can be used, false otherwise. False usually means the request was released. If the user calls any of the methods of this class on an object for which this operator returned false, a `logic_error` will be thrown.

6.10.3.11 operator=()

```
line_request & gpiod::line_request::operator= (
    line_request && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.10.3.12 read_edge_events() [1/2]

```
::std::size_t gpiod::line_request::read_edge_events (
    edge_event_buffer & buffer )
```

Read a number of edge events from this request up to the maximum capacity of the buffer.

Parameters

<i>buffer</i>	Edge event buffer to read events into.
---------------	--

Returns

Number of events read.

6.10.3.13 read_edge_events() [2/2]

```
::std::size_t gpiod::line_request::read_edge_events (
    edge_event_buffer & buffer,
    ::std::size_t max_events )
```

Read a number of edge events from this request.

Parameters

<i>buffer</i>	Edge event buffer to read events into.
<i>max_events</i>	Maximum number of events to read. Limited by the capacity of the buffer.

Returns

Number of events read.

6.10.3.14 reconfigure_lines()

```
line_request & gpiod::line_request::reconfigure_lines (
    const line_config & config )
```

Apply new config options to requested lines.

Parameters

<i>config</i>	New configuration.
---------------	--------------------

Returns

Reference to self.

6.10.3.15 release()

```
void gpiod::line_request::release ( )
```

Release the requested lines and free all associated resources.

Note

The object can still be used after this method is called but using any of the mutators will result in throwing a `logic_error` exception.

6.10.3.16 set_value()

```
line_request & gpiod::line_request::set_value (
    line::offset offset,
    line::value value )
```

Set the value of a single requested line.

Parameters

<i>offset</i>	Offset of the line to set within the chip.
<i>value</i>	New line value.

Returns

Reference to self.

6.10.3.17 set_values() [1/3]

```
line_request & gpiod::line_request::set_values (
    const line::offsets & offsets,
    const line::values & values )
```

Set the values of a subset of requested lines.

Parameters

<i>offsets</i>	Vector containing the offsets of lines to set.
<i>values</i>	Vector containing new values with indexes corresponding with those in the offsets vector.

Returns

Reference to self.

6.10.3.18 set_values() [2/3]

```
line_request & gpiod::line_request::set_values (
    const line::value_mappings & values )
```

Set the values of a subset of requested lines.

Parameters

<i>values</i>	Vector containing a set of offset->value mappings.
---------------	--

Returns

Reference to self.

6.10.3.19 set_values() [3/3]

```
line_request & gpiod::line_request::set_values (
    const line::values & values )
```

Set the values of all requested lines.

Parameters

<i>values</i>	Array of new line values. The size must be equal to the value returned by line_request::num_lines .
---------------	---

Returns

Reference to self.

6.10.3.20 wait_edge_events()

```
bool gpiod::line_request::wait_edge_events (
    const ::std::chrono::nanoseconds & timeout ) const
```

Wait for edge events on any of the lines requested with edge detection enabled.

Parameters

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

Returns

True if at least one event is ready to be read. False if the wait timed out.

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

- [gpiodcxx/line-request.hpp](#)

6.11 gpiod::line_settings Class Reference

Stores GPIO line settings.

```
#include <line-settings.hpp>
```

Public Member Functions

- **line_settings** ()
Initializes the [line_settings](#) object with default values.
- **line_settings** (const [line_settings](#) &other)
Copy constructor.
- **line_settings** ([line_settings](#) &&other) noexcept
Move constructor.
- **line_settings & operator=** (const [line_settings](#) &other)
Copy assignment operator.
- **line_settings & operator=** ([line_settings](#) &&other)
Move assignment operator.
- **line_settings & reset** () noexcept
Reset the line settings to default values.
- **line_settings & set_direction** ([line::direction](#) direction)
Set direction.
- **line::direction** direction () const
Get direction.
- **line_settings & set_edge_detection** ([line::edge](#) edge)
Set edge detection.
- **line::edge** edge_detection () const
Get edge detection.
- **line_settings & set_bias** ([line::bias](#) bias)
Set bias setting.
- **line::bias** bias () const
Get bias setting.
- **line_settings & set_drive** ([line::drive](#) drive)
Set drive setting.
- **line::drive** drive () const
Get drive setting.
- **line_settings & set_active_low** (bool [active_low](#))
Set the active-low setting.
- bool [active_low](#) () const noexcept
Get the active-low setting.
- **line_settings & set_debounce_period** (const ::std::chrono::microseconds &period)
Set debounce period.
- ::std::chrono::microseconds [debounce_period](#) () const noexcept
Get debounce period.
- **line_settings & set_event_clock** ([line::clock](#) event_clock)
Set the event clock to use for edge event timestamps.
- **line::clock** event_clock () const
Get the event clock used for edge event timestamps.
- **line_settings & set_output_value** ([line::value](#) value)
Set the output value.
- **line::value** output_value () const
Get the output value.

Private Attributes

- ::std::unique_ptr< impl > **_m_priv**
- friend **line_config**

6.11.1 Detailed Description

Stores GPIO line settings.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 `line_settings()` [1/2]

```
gpiod::line_settings::line_settings (
    const line_settings & other )
```

Copy constructor.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

6.11.2.2 `line_settings()` [2/2]

```
gpiod::line_settings::line_settings (
    line_settings && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.11.3 Member Function Documentation

6.11.3.1 `active_low()`

```
bool gpiod::line_settings::active_low ( ) const [noexcept]
```

Get the active-low setting.

Returns

Current active-low setting.

6.11.3.2 bias()

```
line::bias gpio::line_settings::bias ( ) const
```

Get bias setting.

Returns

Current bias.

6.11.3.3 debounce_period()

```
::std::chrono::microseconds gpio::line_settings::debounce_period ( ) const [noexcept]
```

Get debounce period.

Returns

Current debounce period.

6.11.3.4 direction()

```
line::direction gpio::line_settings::direction ( ) const
```

Get direction.

Returns

Current direction setting.

6.11.3.5 drive()

```
line::drive gpio::line_settings::drive ( ) const
```

Get drive setting.

Returns

Current drive.

6.11.3.6 edge_detection()

```
line::edge gpio::line_settings::edge_detection ( ) const
```

Get edge detection.

Returns

Current edge detection setting.

6.11.3.7 event_clock()

```
line::clock gpio::line_settings::event_clock ( ) const
```

Get the event clock used for edge event timestamps.

Returns

Current event clock type.

6.11.3.8 operator=() [1/2]

```
line_settings & gpio::line_settings::operator= (
    const line_settings & other )
```

Copy assignment operator.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

Returns

Reference to self.

6.11.3.9 operator=() [2/2]

```
line_settings & gpio::line_settings::operator= (
    line_settings && other )
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.11.3.10 output_value()

```
line::value gpio::line_settings::output_value ( ) const
```

Get the output value.

Returns

Current output value.

6.11.3.11 reset()

```
line_settings & gpio::line_settings::reset ( ) [noexcept]
```

Reset the line settings to default values.

Returns

Reference to self.

6.11.3.12 set_active_low()

```
line_settings & gpio::line_settings::set_active_low (
    bool active_low )
```

Set the active-low setting.

Parameters

<i>active_low</i>	New active-low setting.
-------------------	-------------------------

Returns

Reference to self.

6.11.3.13 set_bias()

```
line_settings & gpiod::line_settings::set_bias (
    line::bias bias )
```

Set bias setting.

Parameters

<i>bias</i>	New bias.
-------------	-----------

Returns

Reference to self.

6.11.3.14 set_debounce_period()

```
line_settings & gpiod::line_settings::set_debounce_period (
    const ::std::chrono::microseconds & period )
```

Set debounce period.

Parameters

<i>period</i>	New debounce period in microseconds.
---------------	--------------------------------------

Returns

Reference to self.

6.11.3.15 set_direction()

```
line_settings & gpiod::line_settings::set_direction (
    line::direction direction )
```

Set direction.

Parameters

<i>direction</i>	New direction.
------------------	----------------

Returns

Reference to self.

6.11.3.16 set_drive()

```
line_settings & gpio::line_settings::set_drive (
    line::drive drive )
```

Set drive setting.

Parameters

<i>drive</i>	New drive.
--------------	------------

Returns

Reference to self.

6.11.3.17 set_edge_detection()

```
line_settings & gpio::line_settings::set_edge_detection (
    line::edge edge )
```

Set edge detection.

Parameters

<i>edge</i>	New edge detection setting.
-------------	-----------------------------

Returns

Reference to self.

6.11.3.18 set_event_clock()

```
line_settings & gpio::line_settings::set_event_clock (
    line::clock event_clock )
```

Set the event clock to use for edge event timestamps.

Parameters

<code>event_clock</code>	Clock to use.
--------------------------	---------------

Returns

Reference to self.

6.11.3.19 set_output_value()

```
line_settings & gpio::line_settings::set_output_value (
    line::value value )
```

Set the output value.

Parameters

<code>value</code>	New output value.
--------------------	-------------------

Returns

Reference to self.

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

- `gpiodcxx/line-settings.hpp`

6.12 gpio::line::offset Class Reference

Wrapper around unsigned int for representing line offsets.

```
#include <line.hpp>
```

Public Member Functions

- `offset` (unsigned int off=0)
Constructor with implicit conversion from unsigned int.
- `offset` (const `offset` &other)=default
Copy constructor.
- `offset` (`offset` &&other)=default
Move constructor.
- `offset` & `operator=` (const `offset` &other)=default
Assignment operator.
- `offset` & `operator=` (`offset` &&other) noexcept=default
Move assignment operator.
- `operator unsigned int` () const noexcept
Conversion operator to unsigned int.

Private Attributes

- unsigned int `_m_offset`

6.12.1 Detailed Description

Wrapper around unsigned int for representing line offsets.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 `offset()` [1/3]

```
gpiod::line::offset::offset (
    unsigned int off = 0 ) [inline]
```

Constructor with implicit conversion from unsigned int.

Parameters

<i>off</i>	Line offset.
------------	--------------

6.12.2.2 `offset()` [2/3]

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

Copy constructor.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

6.12.2.3 `offset()` [3/3]

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

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.12.3 Member Function Documentation

6.12.3.1 operator=() [1/2]

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

Assignment operator.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

Returns

Reference to self.

6.12.3.2 operator=() [2/2]

```
offset & gpiod::line::offset::operator= (  
    offset && other ) [default], [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

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

- gpiodcxx/[line.hpp](#)

6.13 gpiod::request_builder Class Reference

Intermediate object storing the configuration for a line request.

```
#include <request-builder.hpp>
```

Public Member Functions

- **request_builder** (const [request_builder](#) &other)=delete
- **request_builder** ([request_builder](#) &&other) noexcept
Move constructor.
- **request_builder** & **operator=** (const [request_builder](#) &other)=delete
- **request_builder** & **operator=** ([request_builder](#) &&other) noexcept
Move assignment operator.
- **request_builder** & **set_request_config** ([request_config](#) &req_cfg)
Set the request config for the request.
- const [request_config](#) & **get_request_config** () const noexcept
Get the current request config.
- **request_builder** & **set_consumer** (const ::std::string &consumer) noexcept
Set consumer in the request config stored by this object.
- **request_builder** & **set_event_buffer_size** (::std::size_t event_buffer_size) noexcept
Set the event buffer size in the request config stored by this object.
- **request_builder** & **set_line_config** ([line_config](#) &line_cfg)
Set the line config for this request.
- const [line_config](#) & **get_line_config** () const noexcept
Get the current line config.
- **request_builder** & **add_line_settings** ([line::offset](#) offset, const [line_settings](#) &settings)
Add line settings to the line config stored by this object for a single offset.
- **request_builder** & **add_line_settings** (const [line::offsets](#) &offsets, const [line_settings](#) &settings)
Add line settings to the line config stored by this object for a set of offsets.
- **request_builder** & **set_output_values** (const [line::values](#) &values)
Set output values for a number of lines in the line config stored by this object.
- [line_request](#) **do_request** ()
Make the line request.

Private Member Functions

- **request_builder** ([chip](#) &chip)
- friend ::std::ostream & **operator**<< (::std::ostream &out, const [request_builder](#) &builder)

Private Attributes

- ::std::unique_ptr< impl > **_m_priv**
- friend **chip**

6.13.1 Detailed Description

Intermediate object storing the configuration for a line request.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 request_builder()

```
gpiod::request_builder::request_builder (
    request_builder && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to be moved.
--------------	---------------------

6.13.3 Member Function Documentation

6.13.3.1 add_line_settings() [1/2]

```
request_builder & gpiod::request_builder::add_line_settings (
    const line::offsets & offsets,
    const line_settings & settings )
```

Add line settings to the line config stored by this object for a set of offsets.

Parameters

<i>offsets</i>	Offsets for which to add settings.
<i>settings</i>	Settings to add.

Returns

Reference to self.

6.13.3.2 add_line_settings() [2/2]

```
request_builder & gpiod::request_builder::add_line_settings (
    line::offset offset,
    const line_settings & settings )
```

Add line settings to the line config stored by this object for a single offset.

Parameters

<i>offset</i>	Offset for which to add settings.
<i>settings</i>	Line settings to use.

Returns

Reference to self.

6.13.3.3 do_request()

```
line_request gpiod::request_builder::do_request ( )
```

Make the line request.

Returns

New `line_request` object.

6.13.3.4 get_line_config()

```
const line_config & gpiod::request_builder::get_line_config ( ) const [noexcept]
```

Get the current line config.

Returns

Const reference to the current line config stored by this object.

6.13.3.5 get_request_config()

```
const request_config & gpiod::request_builder::get_request_config ( ) const [noexcept]
```

Get the current request config.

Returns

Const reference to the current request config stored by this object.

6.13.3.6 operator=()

```
request_builder & gpiod::request_builder::operator= (
    request_builder && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to be moved.
--------------	---------------------

Returns

Reference to self.

6.13.3.7 set_consumer()

```
request_builder & gpiod::request_builder::set_consumer (
    const ::std::string & consumer ) [noexcept]
```

Set consumer in the request config stored by this object.

Parameters

<i>consumer</i>	New consumer string.
-----------------	----------------------

Returns

Reference to self.

6.13.3.8 set_event_buffer_size()

```
request_builder & gpiod::request_builder::set_event_buffer_size (
    ::std::size_t event_buffer_size ) [noexcept]
```

Set the event buffer size in the request config stored by this object.

Parameters

<i>event_buffer_size</i>	New event buffer size.
--------------------------	------------------------

Returns

Reference to self.

6.13.3.9 set_line_config()

```
request_builder & gpiod::request_builder::set_line_config (
    line_config & line_cfg )
```

Set the line config for this request.

Parameters

<i>line_cfg</i>	Line config to use.
-----------------	---------------------

Returns

Reference to self.

6.13.3.10 set_output_values()

```
request_builder & gpiod::request_builder::set_output_values (
    const line::values & values )
```

Set output values for a number of lines in the line config stored by this object.

Parameters

<i>values</i>	Buffer containing the output values.
---------------	--------------------------------------

Returns

Reference to self.

6.13.3.11 set_request_config()

```
request_builder & gpiod::request_builder::set_request_config (
    request_config & req_cfg )
```

Set the request config for the request.

Parameters

<i>req_cfg</i>	Request config to use.
----------------	------------------------

Returns

Reference to self.

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

- [gpiodcxx/request-builder.hpp](#)

6.14 gpiod::request_config Class Reference

Stores a set of options passed to the kernel when making a line request.

```
#include <request-config.hpp>
```

Public Member Functions

- **request_config** ()
Constructor.
- **request_config** (const [request_config](#) &other)=delete
- **request_config** ([request_config](#) &&other) noexcept
Move constructor.
- **request_config** & **operator=** ([request_config](#) &&other) noexcept
Move assignment operator.
- **request_config** & **set_consumer** (const ::std::string &[consumer](#)) noexcept
Set the consumer name.
- ::std::string **consumer** () const noexcept
Get the consumer name.
- **request_config** & **set_event_buffer_size** (::std::size_t [event_buffer_size](#)) noexcept
Set the size of the kernel event buffer.
- ::std::size_t **event_buffer_size** () const noexcept
Get the edge event buffer size from this request config.

Private Member Functions

- **request_config** & **operator=** (const [request_config](#) &other)

Private Attributes

- ::std::shared_ptr< impl > **_m_priv**
- friend **request_builder**

6.14.1 Detailed Description

Stores a set of options passed to the kernel when making a line request.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 request_config()

```
gpiod::request_config::request_config (
    request\_config && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.14.3 Member Function Documentation

6.14.3.1 consumer()

```
::std::string gpiod::request_config::consumer ( ) const [noexcept]
```

Get the consumer name.

Returns

Currently configured consumer name. May be an empty string.

6.14.3.2 event_buffer_size()

```
::std::size_t gpiod::request_config::event_buffer_size ( ) const [noexcept]
```

Get the edge event buffer size from this request config.

Returns

Current edge event buffer size setting.

6.14.3.3 operator=()

```
request_config & gpiod::request_config::operator= (
    request_config && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.14.3.4 set_consumer()

```
request_config & gpiod::request_config::set_consumer (
    const ::std::string & consumer ) [noexcept]
```

Set the consumer name.

Parameters

<i>consumer</i>	New consumer name.
-----------------	--------------------

Returns

Reference to self.

6.14.3.5 set_event_buffer_size()

```
request_config & gpiod::request_config::set_event_buffer_size (
    ::std::size_t event_buffer_size ) [noexcept]
```

Set the size of the kernel event buffer.

Parameters

<i>event_buffer_size</i>	New event buffer size.
--------------------------	------------------------

Returns

Reference to self.

Note

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

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

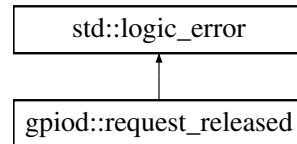
- [gpiodcxx/request-config.hpp](#)

6.15 gpod::request_released Class Reference

Exception thrown when an already released line request is used.

```
#include <exception.hpp>
```

Inheritance diagram for gpod::request_released:



Public Member Functions

- [request_released](#) (const ::std::string &what)
Constructor.
- [request_released](#) (const [request_released](#) &other) noexcept
Copy constructor.
- [request_released](#) ([request_released](#) &&other) noexcept
Move constructor.
- [request_released](#) & [operator=](#) (const [request_released](#) &other) noexcept
Assignment operator.
- [request_released](#) & [operator=](#) ([request_released](#) &&other) noexcept
Move assignment operator.

6.15.1 Detailed Description

Exception thrown when an already released line request is used.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 request_released() [1/3]

```
gpod::request_released::request_released (
    const ::std::string & what ) [explicit]
```

Constructor.

Parameters

<i>what</i>	Human readable reason for error.
-------------	----------------------------------

6.15.2.2 request_released() [2/3]

```
gpiod::request_released::request_released (
    const request_released & other ) [noexcept]
```

Copy constructor.

Parameters

<i>other</i>	Object to copy from.
--------------	----------------------

6.15.2.3 request_released() [3/3]

```
gpiod::request_released::request_released (
    request_released && other ) [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.15.3 Member Function Documentation

6.15.3.1 operator=() [1/2]

```
request_released & gpiod::request_released::operator= (
    const request_released & other ) [noexcept]
```

Assignment operator.

Parameters

<i>other</i>	Object to copy from.
--------------	----------------------

Returns

Reference to self.

6.15.3.2 operator=() [2/2]

```
request_released & gpiod::request_released::operator= (
    request_released && other ) [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

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

- gpiodcxx/[exception.hpp](#)

6.16 gpiod::timestamp Class Reference

Stores the edge and info event timestamps as returned by the kernel and allows to convert them to `std::chrono::time_point`.

```
#include <timestamp.hpp>
```

Public Types

- using **time_point_monotonic** = `std::chrono::time_point<std::chrono::steady_clock>`
Monotonic time_point.
- using **time_point_realtime** = `std::chrono::time_point<std::chrono::system_clock, std::chrono::nanoseconds>`
Real-time time_point.

Public Member Functions

- **timestamp** (`std::uint64_t ns`)
Constructor with implicit conversion from `uint64_t`.
- **timestamp** (const **timestamp** &other) noexcept=default
Copy constructor.
- **timestamp** (**timestamp** &&other) noexcept=default
Move constructor.
- **timestamp** & **operator=** (const **timestamp** &other) noexcept=default
Assignment operator.
- **timestamp** & **operator=** (**timestamp** &&other) noexcept=default
Move assignment operator.
- **operator::std::uint64_t** () noexcept
Conversion operator to `std::uint64_t`.
- `std::uint64_t ns` () const noexcept
Get the timestamp in nanoseconds.
- **time_point_monotonic to_time_point_monotonic** () const
Convert the timestamp to a monotonic time_point.
- **time_point_realtime to_time_point_realtime** () const
Convert the timestamp to a real-time time_point.

Private Attributes

- `::std::uint64_t _m_ns`

6.16.1 Detailed Description

Stores the edge and info event timestamps as returned by the kernel and allows to convert them to `std::chrono::time_point`.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 `timestamp()` [1/3]

```
gpiod::timestamp::timestamp (
    ::std::uint64_t ns ) [inline]
```

Constructor with implicit conversion from `uint64_t`.

Parameters

<i>ns</i>	Timestamp in nanoseconds.
-----------	---------------------------

6.16.2.2 `timestamp()` [2/3]

```
gpiod::timestamp::timestamp (
    const timestamp & other ) [default], [noexcept]
```

Copy constructor.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

6.16.2.3 `timestamp()` [3/3]

```
gpiod::timestamp::timestamp (
    timestamp && other ) [default], [noexcept]
```

Move constructor.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

6.16.3 Member Function Documentation

6.16.3.1 ns()

```
::std::uint64_t gpiod::timestamp::ns ( ) const [inline], [noexcept]
```

Get the timestamp in nanoseconds.

Returns

Timestamp in nanoseconds.

6.16.3.2 operator=() [1/2]

```
timestamp & gpiod::timestamp::operator= (
    const timestamp & other ) [default], [noexcept]
```

Assignment operator.

Parameters

<i>other</i>	Object to copy.
--------------	-----------------

Returns

Reference to self.

6.16.3.3 operator=() [2/2]

```
timestamp & gpiod::timestamp::operator= (
    timestamp && other ) [default], [noexcept]
```

Move assignment operator.

Parameters

<i>other</i>	Object to move.
--------------	-----------------

Returns

Reference to self.

6.16.3.4 to_time_point_monotonic()

```
time_point_monotonic gpiod::timestamp::to_time_point_monotonic ( ) const [inline]
```

Convert the timestamp to a monotonic time_point.

Returns

time_point associated with the steady clock.

6.16.3.5 to_time_point_realtime()

```
time_point_realtime gpiod::timestamp::to_time_point_realtime ( ) const [inline]
```

Convert the timestamp to a real-time time_point.

Returns

time_point associated with the system clock.

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

- [gpiodcxx/timestamp.hpp](#)

Chapter 7

File Documentation

7.1 gpiod.hpp File Reference

```
#include "gpiodcxx/chip.hpp"
#include "gpiodcxx/chip-info.hpp"
#include "gpiodcxx/edge-event.hpp"
#include "gpiodcxx/edge-event-buffer.hpp"
#include "gpiodcxx/exception.hpp"
#include "gpiodcxx/info-event.hpp"
#include "gpiodcxx/line.hpp"
#include "gpiodcxx/line-config.hpp"
#include "gpiodcxx/line-info.hpp"
#include "gpiodcxx/line-request.hpp"
#include "gpiodcxx/line-settings.hpp"
#include "gpiodcxx/request-builder.hpp"
#include "gpiodcxx/request-config.hpp"
```

7.2 gpiod.hpp

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

```
1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4 #ifndef __LIBGPIOD_GPIOD_CXX_HPP__
5 #define __LIBGPIOD_GPIOD_CXX_HPP__
6
7 /*
8  * We don't make this symbol private because it needs to be accessible by
9  * the declarations in exception.hpp in order to expose the symbols of classes
10  * inheriting from standard exceptions.
11  */
12 #define GPIOD_CXX_API __attribute__((visibility("default")))
13
14 #define __LIBGPIOD_GPIOD_CXX_INSIDE__
15 #include "gpiodcxx/chip.hpp"
16 #include "gpiodcxx/chip-info.hpp"
17 #include "gpiodcxx/edge-event.hpp"
18 #include "gpiodcxx/edge-event-buffer.hpp"
19 #include "gpiodcxx/exception.hpp"
20 #include "gpiodcxx/info-event.hpp"
21 #include "gpiodcxx/line.hpp"
22 #include "gpiodcxx/line-config.hpp"
23 #include "gpiodcxx/line-info.hpp"
24 #include "gpiodcxx/line-request.hpp"
25 #include "gpiodcxx/line-settings.hpp"
26 #include "gpiodcxx/request-builder.hpp"
27 #include "gpiodcxx/request-config.hpp"
28 #undef __LIBGPIOD_GPIOD_CXX_INSIDE__
29
30 #endif /* __LIBGPIOD_GPIOD_CXX_HPP__ */
```

7.3 gpiodcxx/chip-info.hpp File Reference

```
#include <memory>
#include <ostream>
```

Classes

- class [gpiod::chip_info](#)
Represents an immutable snapshot of GPIO chip information.

Functions

- `::std::ostream & gpiod::operator<< (::std::ostream &out, const chip_info &chip)`
Stream insertion operator for GPIO chip objects.

7.3.1 Function Documentation

7.3.1.1 operator<<()

```
::std::ostream & gpiod::operator<< (
    ::std::ostream & out,
    const chip_info & chip )
```

Stream insertion operator for GPIO chip objects.

Parameters

<i>out</i>	Output stream to write to.
<i>chip</i>	GPIO chip to insert into the output stream.

Returns

Reference to out.

7.4 chip-info.hpp

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

```
1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4 #ifndef __LIBGPIOD_CXX_CHIP_INFO_HPP__
5 #define __LIBGPIOD_CXX_CHIP_INFO_HPP__
6
7 #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
8 #error "Only gpiod.hpp can be included directly."
```



```

13 #endif
14
15 #include <memory>
16 #include <ostream>
17
18 namespace gpiod {
19
20 class chip;
21
22 class chip_info final
23 {
24 public:
25     chip_info(const chip_info& other);
26
27     chip_info(chip_info&& other) noexcept;
28
29     ~chip_info();
30
31     chip_info& operator=(const chip_info& other);
32
33     chip_info& operator=(chip_info&& other) noexcept;
34
35     ::std::string name() const noexcept;
36
37     ::std::string label() const noexcept;
38
39     ::std::size_t num_lines() const noexcept;
40
41 private:
42     chip_info();
43
44     struct impl;
45
46     ::std::shared_ptr<impl> _m_priv;
47
48     friend chip;
49 };
50
51 ::std::ostream& operator<< (::std::ostream& out, const chip_info& chip);
52
53 } /* namespace gpiod */
54
55 #endif /* __LIBGPIOD_CXX_CHIP_INFO_HPP__ */

```

7.5 gpiodcxx/chip.hpp File Reference

```

#include <chrono>
#include <cstdint>
#include <iostream>
#include <filesystem>
#include <memory>
#include "line.hpp"

```

Classes

- class [gpiod::chip](#)
Represents a GPIO chip.

Functions

- [::std::ostream & gpiod::operator<< \(::std::ostream &out, const chip &chip\)](#)
Stream insertion operator for GPIO chip objects.

7.5.1 Function Documentation

7.5.1.1 operator<<()

```
::std::ostream & gpiod::operator<< (
    ::std::ostream & out,
    const chip & chip )
```

Stream insertion operator for GPIO chip objects.

Parameters

<i>out</i>	Output stream to write to.
<i>chip</i>	GPIO chip to insert into the output stream.

Returns

Reference to out.

7.6 chip.hpp

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

```
1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4 #ifndef __LIBGPIOD_CXX_CHIP_HPP__
5 #define __LIBGPIOD_CXX_CHIP_HPP__
6
7 #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
8 #error "Only gpiod.hpp can be included directly."
9 #endif
10
11 #include <chrono>
12 #include <cstdint>
13 #include <iostream>
14 #include <filesystem>
15 #include <memory>
16
17 #include "line.hpp"
18
19 namespace gpiod {
20
21 class chip_info;
22 class info_event;
23 class line_config;
24 class line_info;
25 class line_request;
26 class request_builder;
27 class request_config;
28
29 class chip final
30 {
31 public:
32     explicit chip(const ::std::filesystem::path& path);
33     chip(chip&& other) noexcept;
34     ~chip();
35     chip& operator=(const chip& other) = delete;
36     chip& operator=(chip&& other) noexcept;
```

```

63
71     explicit operator bool() const noexcept;
72
78     void close();
79
84     ::std::filesystem::path path() const;
85
90     chip_info get_info() const;
91
98     line_info get_line_info(line::offset offset) const;
99
106    line_info watch_line_info(line::offset offset) const;
107
112    void unwatch_line_info(line::offset offset) const;
113
118    int fd() const;
119
130    bool wait_info_event(const ::std::chrono::nanoseconds& timeout) const;
131
136    info_event read_info_event() const;
137
144    int get_line_offset_from_name(const ::std::string& name) const;
145
150    request_builder prepare_request();
151
152 private:
153
154     struct impl;
155
156     ::std::shared_ptr<impl> _m_priv;
157
158     chip(const chip& other);
159
160     friend request_builder;
161 };
162
169 ::std::ostream& operator<< (::std::ostream& out, const chip& chip);
170
171 } /* namespace gpiod */
172
173 #endif /* __LIBGPIOD_CXX_CHIP_HPP__ */

```

7.7 gpiodcxx/edge-event-buffer.hpp File Reference

```

#include <cstdint>
#include <iostream>
#include <memory>
#include <vector>

```

Classes

- class [gpiod::edge_event_buffer](#)
Object into which edge events are read for better performance.

Functions

- `::std::ostream & gpiod::operator<< (::std::ostream &out, const edge_event_buffer &buf)`
Stream insertion operator for GPIO edge event buffer objects.

7.7.1 Function Documentation

7.7.1.1 operator<<()

```
::std::ostream & gpiod::operator<< (
    ::std::ostream & out,
    const edge_event_buffer & buf )
```

Stream insertion operator for GPIO edge event buffer objects.

Parameters

<i>out</i>	Output stream to write to.
<i>buf</i>	GPIO edge event buffer object to insert into the output stream.

Returns

Reference to out.

7.8 edge-event-buffer.hpp

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

```
1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4 #ifndef __LIBGPIO_CXX_EDGE_EVENT_BUFFER_HPP__
5 #define __LIBGPIO_CXX_EDGE_EVENT_BUFFER_HPP__
6
7 #if !defined(__LIBGPIO_GPIOD_CXX_INSIDE__)
8 #error "Only gpiod.hpp can be included directly."
9 #endif
10
11 #include <cstdint>
12 #include <iostream>
13 #include <memory>
14 #include <vector>
15
16 namespace gpiod {
17
18     class edge_event;
19     class line_request;
20
21     class edge_event_buffer final
22     {
23     public:
24         using const_iterator = ::std::vector<edge_event>::const_iterator;
25
26         explicit edge_event_buffer(::std::size_t capacity = 64);
27
28         edge_event_buffer(const edge_event_buffer& other) = delete;
29
30         edge_event_buffer(edge_event_buffer&& other) noexcept;
31
32         ~edge_event_buffer();
33
34         edge_event_buffer& operator=(const edge_event_buffer& other) = delete;
35
36         edge_event_buffer& operator=(edge_event_buffer&& other) noexcept;
37
38         const edge_event& get_event(unsigned int index) const;
39
40         ::std::size_t num_events() const;
41
42         ::std::size_t capacity() const noexcept;
43
44         const_iterator begin() const noexcept;
45
46         const_iterator end() const noexcept;
47     private:
48     }
```

```

103     struct impl;
104
105     ::std::unique_ptr<impl> _m_priv;
106
107     friend line_request;
108 };
109
110 ::std::ostream& operator<< (::std::ostream& out, const edge_event_buffer& buf);
111
112 } /* namespace gpiod */
113
114 #endif /* __LIBGPIOD_CXX_EDGE_EVENT_BUFFER_HPP__ */

```

7.9 gpiodcxx/edge-event.hpp File Reference

```

#include <cstdint>
#include <iostream>
#include <memory>
#include "timestamp.hpp"

```

Classes

- class [gpiod::edge_event](#)
Immutable object containing data about a single edge event.

Functions

- [::std::ostream & gpiod::operator<< \(::std::ostream &out, const edge_event &event\)](#)
Stream insertion operator for edge events.

7.9.1 Function Documentation

7.9.1.1 operator<<()

```

::std::ostream & gpiod::operator<< (
    ::std::ostream & out,
    const edge_event & event )

```

Stream insertion operator for edge events.

Parameters

<i>out</i>	Output stream to write to.
<i>event</i>	Edge event to insert into the output stream.

Returns

Reference to out.

7.10 edge-event.hpp

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

```

1  /* SPDX-License-Identifier: LGPL-2.1-or-later */
2  /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4  #ifndef __LIBGPIOD_CXX_EDGE_EVENT_HPP__
5  #define __LIBGPIOD_CXX_EDGE_EVENT_HPP__
6
7  #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
8  #error "Only gpiod.hpp can be included directly."
9  #endif
10
11 #include <cstdint>
12 #include <iostream>
13 #include <memory>
14
15 #include "timestamp.hpp"
16
17 namespace gpiod {
18
19     class edge_event_buffer;
20
21     class edge_event final
22     {
23     public:
24
25         enum class event_type
26         {
27             RISING_EDGE = 1,
28             FALLING_EDGE,
29         };
30
31         edge_event(const edge_event& other);
32
33         edge_event(edge_event&& other) noexcept;
34
35         ~edge_event();
36
37         edge_event& operator=(const edge_event& other);
38
39         edge_event& operator=(edge_event&& other) noexcept;
40
41         event_type type() const;
42
43         timestamp timestamp_ns() const noexcept;
44
45         line::offset line_offset() const noexcept;
46
47         unsigned long global_seqno() const noexcept;
48
49         unsigned long line_seqno() const noexcept;
50
51     private:
52
53         edge_event();
54
55         struct impl;
56         struct impl_managed;
57         struct impl_external;
58
59         ::std::shared_ptr<impl> _m_priv;
60
61         friend edge_event_buffer;
62     };
63
64     ::std::ostream& operator<< (::std::ostream& out, const edge_event& event);
65
66 } /* namespace gpiod */
67
68 #endif /* __LIBGPIOD_CXX_EDGE_EVENT_HPP__ */

```

7.11 gpiodcxx/exception.hpp File Reference

```
#include <stdexcept>
#include <string>
```

Classes

- class [gpiod::chip_closed](#)
Exception thrown when an already closed chip is used.
- class [gpiod::request_released](#)
Exception thrown when an already released line request is used.
- class [gpiod::bad_mapping](#)
Exception thrown when the core C library returns an invalid value for any of the [line_info](#) properties.

7.12 exception.hpp

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

```
1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4 #ifndef __LIBGPIOD_CXX_EXCEPTION_HPP__
5 #define __LIBGPIOD_CXX_EXCEPTION_HPP__
6
7 #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
8 #error "Only gpiod.hpp can be included directly."
9 #endif
10
11 #include <stdexcept>
12 #include <string>
13
14 namespace gpiod {
15
16     class GPIOD_CXX_API chip_closed final : public ::std::logic_error
17     {
18     public:
19         explicit chip_closed(const ::std::string& what);
20         chip_closed(const chip_closed& other) noexcept;
21         chip_closed(chip_closed&& other) noexcept;
22         chip_closed& operator=(const chip_closed& other) noexcept;
23         chip_closed& operator=(chip_closed&& other) noexcept;
24         ~chip_closed();
25     };
26
27     class GPIOD_CXX_API request_released final : public ::std::logic_error
28     {
29     public:
30         explicit request_released(const ::std::string& what);
31         request_released(const request_released& other) noexcept;
32         request_released(request_released&& other) noexcept;
33         request_released& operator=(const request_released& other) noexcept;
34         request_released& operator=(request_released&& other) noexcept;
35         ~request_released();
36     };
37
38     class GPIOD_CXX_API bad_mapping final : public ::std::runtime_error
39     {
40     public:
41         explicit bad_mapping(const ::std::string& what);
42         bad_mapping(const bad_mapping& other) noexcept;
43         bad_mapping(bad_mapping&& other) noexcept;
44         bad_mapping& operator=(const bad_mapping& other) noexcept;
45         bad_mapping& operator=(bad_mapping&& other) noexcept;
46         ~bad_mapping();
47     };
48 }
```

```

111
112     explicit bad_mapping(const ::std::string& what);
113
114     bad_mapping(const bad_mapping& other) noexcept;
115
116     bad_mapping(bad_mapping&& other) noexcept;
117
118     bad_mapping& operator=(const bad_mapping& other) noexcept;
119
120     bad_mapping& operator=(bad_mapping&& other) noexcept;
121
122     ~bad_mapping();
123 };
124 } /* namespace gpod */
125
126 #endif /* __LIBGPID_CXX_EXCEPTION_HPP__ */

```

7.13 info-event.hpp

```

1  /* SPDX-License-Identifier: LGPL-2.1-or-later */
2  /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4  #ifndef __LIBGPID_CXX_INFO_EVENT_HPP__
5  #define __LIBGPID_CXX_INFO_EVENT_HPP__
6
7  #if !defined(__LIBGPID_GPID_CXX_INSIDE__)
8  #error "Only gpod.hpp can be included directly."
9  #endif
10
11 #include <stdint>
12 #include <iostream>
13 #include <memory>
14
15 #include "timestamp.hpp"
16
17 namespace gpod {
18
19     class chip;
20     class line_info;
21
22     class info_event final
23     {
24     public:
25         enum class event_type
26         {
27             LINE_REQUESTED = 1,
28             LINE_RELEASED,
29             LINE_CONFIG_CHANGED,
30         };
31
32         info_event(const info_event& other);
33
34         info_event(info_event&& other) noexcept;
35
36         ~info_event();
37
38         info_event& operator=(const info_event& other);
39
40         info_event& operator=(info_event&& other) noexcept;
41
42         event_type type() const;
43
44         ::std::uint64_t timestamp_ns() const noexcept;
45
46         const line_info& get_line_info() const noexcept;
47
48     private:
49         info_event();
50
51         struct impl;
52
53         ::std::shared_ptr<impl> _m_priv;
54
55         friend chip;
56     };
57
58     ::std::ostream& operator<< (::std::ostream& out, const info_event& event);
59
60 } /* namespace gpod */
61
62 #endif /* __LIBGPID_CXX_INFO_EVENT_HPP__ */

```


7.14 gpiodcxx/line-config.hpp File Reference

```
#include <map>
#include <memory>
```

Classes

- class [gpiod::line_config](#)
Contains a set of line config options used in line requests and reconfiguration.

Functions

- `::std::ostream & gpiod::operator<< (::std::ostream &out, const line_config &config)`
Stream insertion operator for GPIO line config objects.

7.14.1 Function Documentation

7.14.1.1 `operator<<()`

```
::std::ostream & gpiod::operator<< (
    ::std::ostream & out,
    const line_config & config )
```

Stream insertion operator for GPIO line config objects.

Parameters

<i>out</i>	Output stream to write to.
<i>config</i>	Line config object to insert into the output stream.

Returns

Reference to out.

7.15 line-config.hpp

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

```
1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4 #ifndef __LIBGPIO_CXX_LINE_CONFIG_HPP__
5 #define __LIBGPIO_CXX_LINE_CONFIG_HPP__
6
7 #if !defined(__LIBGPIO_GPIOD_CXX_INSIDE__)
8 #error "Only gpiod.hpp can be included directly."
```

```

13 #endif
14
15 #include <map>
16 #include <memory>
17
18 namespace gpiod {
19
20 class chip;
21 class line_request;
22 class line_settings;
23
24 class line_config final
25 {
26 public:
27
28     line_config();
29
30     line_config(const line_config& other) = delete;
31
32     line_config(line_config&& other) noexcept;
33
34     ~line_config();
35
36     line_config& operator=(line_config&& other) noexcept;
37
38     line_config& reset() noexcept;
39
40     line_config& add_line_settings(line::offset offset, const line_settings& settings);
41
42     line_config& add_line_settings(const line::offsets& offsets, const line_settings& settings);
43
44     line_config& set_output_values(const line::values& values);
45
46     ::std::map<line::offset, line_settings> get_line_settings() const;
47
48 private:
49
50     struct impl;
51
52     ::std::shared_ptr<impl> _m_priv;
53
54     line_config& operator=(const line_config& other);
55
56     friend line_request;
57     friend request_builder;
58 };
59
60 ::std::ostream& operator<< (::std::ostream& out, const line_config& config);
61
62 } /* namespace gpiod */
63
64 #endif /* __LIBGPIOD_CXX_LINE_CONFIG_HPP__ */

```

7.16 gpiodcxx/line-info.hpp File Reference

```

#include <chrono>
#include <iostream>
#include <memory>
#include <string>

```

Classes

- class [gpiod::line_info](#)

Contains an immutable snapshot of the line's state at the time when the object of this class was instantiated.

Functions

- [::std::ostream & gpiod::operator<< \(::std::ostream &out, const line_info &info\)](#)

Stream insertion operator for GPIO line info objects.

7.16.1 Function Documentation

7.16.1.1 operator<<()

```
::std::ostream & gpiod::operator<< (
    ::std::ostream & out,
    const line_info & info )
```

Stream insertion operator for GPIO line info objects.

Parameters

<i>out</i>	Output stream to write to.
<i>info</i>	GPIO line info object to insert into the output stream.

Returns

Reference to out.

7.17 line-info.hpp

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

```
1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4 #ifndef __LIBGPIO_CXX_LINE_INFO_HPP__
5 #define __LIBGPIO_CXX_LINE_INFO_HPP__
6
7 #if !defined(__LIBGPIO_GPIOD_CXX_INSIDE__)
8 #error "Only gpiod.hpp can be included directly."
9 #endif
10
11 #include <chrono>
12 #include <iostream>
13 #include <memory>
14 #include <string>
15
16 namespace gpiod {
17
18     class chip;
19     class info_event;
20
21     class line_info final
22     {
23     public:
24
25         line_info(const line_info& other) noexcept;
26
27         line_info(line_info&& other) noexcept;
28
29         ~line_info();
30
31         line_info& operator=(const line_info& other) noexcept;
32
33         line_info& operator=(line_info&& other) noexcept;
34
35         line::offset offset() const noexcept;
36
37         ::std::string name() const noexcept;
38
39         bool used() const noexcept;
40
41         ::std::string consumer() const noexcept;
```

```

91
92     line::direction direction() const;
93
94     line::edge edge_detection() const;
95
96     line::bias bias() const;
97
98     line::drive drive() const;
99
100    bool active_low() const noexcept;
101
102    bool debounced() const noexcept;
103
104    ::std::chrono::microseconds debounce_period() const noexcept;
105
106    line::clock event_clock() const;
107
108 private:
109
110     line_info();
111
112     struct impl;
113
114     ::std::shared_ptr<impl> _m_priv;
115
116     friend chip;
117     friend info_event;
118 };
119
120 ::std::ostream& operator<< (::std::ostream& out, const line_info& info);
121
122 } /* namespace gpiod */
123
124 #endif /* __LIBGPIOD_CXX_LINE_INFO_HPP__ */

```

7.18 gpiodcxx/line-request.hpp File Reference

```

#include <chrono>
#include <cstdint>
#include <iostream>
#include <memory>
#include "misc.hpp"

```

Classes

- class [gpiod::line_request](#)
Stores the context of a set of requested GPIO lines.

Functions

- [::std::ostream & gpiod::operator<<](#) (::std::ostream &out, const line_request &request)
Stream insertion operator for line requests.

7.18.1 Function Documentation

7.18.1.1 operator<<()

```

::std::ostream & gpiod::operator<< (
    ::std::ostream & out,
    const line_request & request )

```

Stream insertion operator for line requests.

Parameters

<i>out</i>	Output stream to write to.
<i>request</i>	Line request object to insert into the output stream.

Returns

Reference to out.

7.19 line-request.hpp

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

```

1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4 #ifndef __LIBGPIOD_CXX_LINE_REQUEST_HPP__
5 #define __LIBGPIOD_CXX_LINE_REQUEST_HPP__
6
7 #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
8 #error "Only gpiod.hpp can be included directly."
9 #endif
10
11 #include <chrono>
12 #include <cstdint>
13 #include <iostream>
14 #include <memory>
15
16 #include "misc.hpp"
17
18 namespace gpiod {
19
20 class chip;
21 class edge_event;
22 class edge_event_buffer;
23 class line_config;
24
25 class line_request final
26 {
27 public:
28     line_request(const line_request& other) = delete;
29     line_request(line_request&& other) noexcept;
30     ~line_request();
31     line_request& operator=(const line_request& other) = delete;
32     line_request& operator=(line_request&& other) noexcept;
33     explicit operator bool() const noexcept;
34     void release();
35     ::std::string chip_name() const;
36     ::std::size_t num_lines() const;
37     line::offsets offsets() const;
38     line::value get_value(line::offset offset);
39     line::values get_values(const line::offsets& offsets);
40     line::values get_values();
41     void get_values(const line::offsets& offsets, line::values& values);
42     void get_values(line::values& values);
43     line_request& set_value(line::offset offset, line::value value);
44     line_request& set_values(const line::value_mappings& values);
45     line_request& set_values(const line::offsets& offsets, const line::values& values);
46

```

```

161     line_request& set_values(const line::values& values);
162
163     line_request& reconfigure_lines(const line_config& config);
164
165     int fd() const;
166
167     bool wait_edge_events(const ::std::chrono::nanoseconds& timeout) const;
168
169     ::std::size_t read_edge_events(edge_event_buffer& buffer);
170
171     ::std::size_t read_edge_events(edge_event_buffer& buffer, ::std::size_t max_events);
172
173 private:
174     line_request();
175
176     struct impl;
177
178     ::std::unique_ptr<impl> _m_priv;
179
180     friend request_builder;
181 };
182
183 ::std::ostream& operator<< (::std::ostream& out, const line_request& request);
184
185 } /* namespace gpiod */
186
187 #endif /* __LIBGPIOD_CXX_LINE_REQUEST_HPP__ */

```

7.20 line-settings.hpp

```

1  /* SPDX-License-Identifier: LGPL-2.1-or-later */
2  /* SPDX-FileCopyrightText: 2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4  #ifndef __LIBGPIOD_CXX_LINE_SETTINGS_HPP__
5  #define __LIBGPIOD_CXX_LINE_SETTINGS_HPP__
6
7  #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
8  #error "Only gpiod.hpp can be included directly."
9  #endif
10
11 #include <chrono>
12 #include <memory>
13
14 #include "line.hpp"
15
16 namespace gpiod {
17
18     class line_config;
19
20     class line_settings final
21     {
22     public:
23
24         line_settings();
25
26         line_settings(const line_settings& other);
27
28         line_settings(line_settings&& other) noexcept;
29
30         ~line_settings();
31
32         line_settings& operator=(const line_settings& other);
33
34         line_settings& operator=(line_settings&& other);
35
36         line_settings& reset() noexcept;
37
38         line_settings& set_direction(line::direction direction);
39
40         line::direction direction() const;
41
42         line_settings& set_edge_detection(line::edge edge);
43
44         line::edge edge_detection() const;
45
46         line_settings& set_bias(line::bias bias);
47
48         line::bias bias() const;
49
50         line_settings& set_drive(line::drive drive);
51
52         line::drive drive() const;

```

```

121
127     line_settings& set_active_low(bool active_low);
128
133     bool active_low() const noexcept;
134
140     line_settings& set_debounce_period(const ::std::chrono::microseconds& period);
141
146     ::std::chrono::microseconds debounce_period() const noexcept;
147
153     line_settings& set_event_clock(line::clock event_clock);
154
159     line::clock event_clock() const;
160
166     line_settings& set_output_value(line::value value);
167
172     line::value output_value() const;
173
174 private:
175
176     struct impl;
177
178     ::std::unique_ptr<impl> _m_priv;
179
180     friend line_config;
181 };
182
189 ::std::ostream& operator<< (::std::ostream& out, const line_settings& settings);
190
191 } /* namespace gpiod */
192
193 #endif /* __LIBGPIOD_CXX_LINE_SETTINGS_HPP__ */

```

7.21 gpiodcxx/line.hpp File Reference

```

#include <ostream>
#include <utility>
#include <vector>

```

Classes

- class [gpiod::line::offset](#)
Wrapper around unsigned int for representing line offsets.

Namespaces

- namespace [gpiod::line](#)
Namespace containing various type definitions for GPIO lines.

Typedefs

- using [gpiod::line::offsets](#) = ::std::vector< offset >
Vector of line offsets.
- using [gpiod::line::values](#) = ::std::vector< value >
Vector of line values.
- using [gpiod::line::value_mapping](#) = ::std::pair< offset, value >
Represents a mapping of a line offset to line logical state.
- using [gpiod::line::value_mappings](#) = ::std::vector< value_mapping >
Vector of offset->value mappings. Each mapping is defined as a pair of an unsigned and signed integers.

Enumerations

- enum class `gpiod::line::value` { `gpiod::line::INACTIVE` = 0 , `gpiod::line::ACTIVE` = 1 }
Logical line states.
- enum class `gpiod::line::direction` { `gpiod::line::AS_IS` = 1 , `gpiod::line::INPUT` , `gpiod::line::OUTPUT` }
Direction settings.
- enum class `gpiod::line::edge` { `gpiod::line::NONE` = 1 , `gpiod::line::RISING` , `gpiod::line::FALLING` , `gpiod::line::BOTH` }
Edge detection settings.
- enum class `gpiod::line::bias` { `gpiod::line::AS_IS` = 1 , `gpiod::line::UNKNOWN` , `gpiod::line::DISABLED` , `gpiod::line::PULL_UP` , `gpiod::line::PULL_DOWN` }
Internal bias settings.
- enum class `gpiod::line::drive` { `gpiod::line::PUSH_PULL` = 1 , `gpiod::line::OPEN_DRAIN` , `gpiod::line::OPEN_SOURCE` }
Drive settings.
- enum class `gpiod::line::clock` { `gpiod::line::MONOTONIC` = 1 , `gpiod::line::REALTIME` , `HTE` }
Event clock settings.

Functions

- `::std::ostream & gpiod::line::operator<< (::std::ostream &out, value val)`
Stream insertion operator for logical line values.
- `::std::ostream & gpiod::line::operator<< (::std::ostream &out, direction dir)`
Stream insertion operator for direction values.
- `::std::ostream & gpiod::line::operator<< (::std::ostream &out, edge edge)`
Stream insertion operator for edge detection values.
- `::std::ostream & gpiod::line::operator<< (::std::ostream &out, bias bias)`
Stream insertion operator for bias values.
- `::std::ostream & gpiod::line::operator<< (::std::ostream &out, drive drive)`
Stream insertion operator for drive values.
- `::std::ostream & gpiod::line::operator<< (::std::ostream &out, clock clock)`
Stream insertion operator for event clock values.
- `::std::ostream & gpiod::line::operator<< (::std::ostream &out, const values &vals)`
Stream insertion operator for the list of output values.
- `::std::ostream & gpiod::line::operator<< (::std::ostream &out, const offsets &offs)`
Stream insertion operator for the list of line offsets.
- `::std::ostream & gpiod::line::operator<< (::std::ostream &out, const value_mapping &mapping)`
Stream insertion operator for the offset-to-value mapping.
- `::std::ostream & gpiod::line::operator<< (::std::ostream &out, const value_mappings &mappings)`
Stream insertion operator for the list of offset-to-value mappings.

7.22 line.hpp

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

```

1  /* SPDX-License-Identifier: LGPL-2.1-or-later */
2  /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4  #ifndef __LIBGPIOD_CXX_LINE_HPP__
5  #define __LIBGPIOD_CXX_LINE_HPP__
6
7  #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
8  #error "Only gpiod.hpp can be included directly."
9  #endif
10
11 #include <ostream>
12 #include <utility>
13 #include <vector>
14
15 namespace gpiod {
16
17     namespace line {
18
19         class offset
20         {
21         public:
22             offset(unsigned int off = 0) : _m_offset(off) {}
23
24             offset(const offset& other) = default;
25
26             offset(offset&& other) = default;
27
28             ~offset() = default;
29
30             offset& operator=(const offset& other) = default;
31             offset& operator=(offset&& other) noexcept = default;
32
33             operator unsigned int() const noexcept
34             {
35                 return this->_m_offset;
36             }
37
38         private:
39             unsigned int _m_offset;
40         };
41
42         enum class value
43         {
44             INACTIVE = 0,
45             ACTIVE = 1,
46         };
47
48         enum class direction
49         {
50             AS_IS = 1,
51             INPUT,
52             OUTPUT,
53         };
54
55         enum class edge
56         {
57             NONE = 1,
58             RISING,
59             FALLING,
60             BOTH,
61         };
62
63         enum class bias
64         {
65             AS_IS = 1,
66             UNKNOWN,
67             DISABLED,
68             PULL_UP,
69             PULL_DOWN,
70         };
71
72         enum class drive
73         {
74             PUSH_PULL = 1,
75             OPEN_DRAIN,
76             OPEN_SOURCE,
77         };
78
79         enum class clock
80         {
81             MONOTONIC = 1,

```

```

154     REALTIME,
155     HTE,
156     /*« Line uses the hardware timestamp engine for event timestamps. */
157 };
158
159 using offsets = ::std::vector<offset>;
160 using values = ::std::vector<value>;
161
162 using value_mapping = ::std::pair<offset, value>;
163
164 using value_mappings = ::std::vector<value_mapping>;
165
166 ::std::ostream& operator<< (::std::ostream& out, value val);
167
168 ::std::ostream& operator<< (::std::ostream& out, direction dir);
169
170 ::std::ostream& operator<< (::std::ostream& out, edge edge);
171
172 ::std::ostream& operator<< (::std::ostream& out, bias bias);
173
174 ::std::ostream& operator<< (::std::ostream& out, drive drive);
175
176 ::std::ostream& operator<< (::std::ostream& out, clock clock);
177
178 ::std::ostream& operator<< (::std::ostream& out, const values& vals);
179
180 ::std::ostream& operator<< (::std::ostream& out, const offsets& offs);
181
182 ::std::ostream& operator<< (::std::ostream& out, const value_mapping& mapping);
183
184 ::std::ostream& operator<< (::std::ostream& out, const value_mappings& mappings);
185
186 } /* namespace line */
187
188 } /* namespace gpio */
189
190 #endif /* __LIBGPIOD_CXX_LINE_HPP__ */

```

7.23 gpiodcxx/misc.hpp File Reference

```
#include <string>
```

Functions

- bool [gpiod::is_gpiochip_device](#) (const ::std::filesystem::path &path)
Check if the file pointed to by path is a GPIO chip character device.
- const ::std::string & [gpiod::api_version](#) ()
Get the human readable version string for libgpiod API.

7.23.1 Function Documentation

7.23.1.1 [api_version\(\)](#)

```
const ::std::string & gpiod::api_version ( )
```

Get the human readable version string for libgpiod API.

Returns

String containing the library version.

7.23.1.2 is_gpiochip_device()

```
bool gpiod::is_gpiochip_device (
    const ::std::filesystem::path & path )
```

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

Parameters

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

Returns

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

7.24 misc.hpp

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

```
1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2021 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4 #ifndef __LIBGPIOD_CXX_MISC_HPP__
5 #define __LIBGPIOD_CXX_MISC_HPP__
6
7 #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
8 #error "Only gpiod.hpp can be included directly."
9 #endif
10
11 #include <string>
12
13 namespace gpiod {
14
15 bool is_gpiochip_device(const ::std::filesystem::path& path);
16
17 const ::std::string& api_version();
18
19 } /* namespace gpiod */
20
21 #endif /* __LIBGPIOD_CXX_MISC_HPP__ */
```

7.25 gpiodcxx/request-builder.hpp File Reference

```
#include <memory>
#include <ostream>
```

Classes

- class [gpiod::request_builder](#)
Intermediate object storing the configuration for a line request.

Functions

- `::std::ostream & gpiod::operator<< (::std::ostream &out, const request_builder &builder)`
Stream insertion operator for GPIO request builder objects.

7.25.1 Function Documentation

7.25.1.1 operator<<()

```
::std::ostream & gpiod::operator<< (
    ::std::ostream & out,
    const request_builder & builder )
```

Stream insertion operator for GPIO request builder objects.

Parameters

<i>out</i>	Output stream to write to.
<i>builder</i>	Request builder object to insert into the output stream.

Returns

Reference to out.

7.26 request-builder.hpp

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

```
1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4
5 #ifndef __LIBGPIOD_CXX_REQUEST_BUILDER_HPP__
6 #define __LIBGPIOD_CXX_REQUEST_BUILDER_HPP__
7
8 #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
9 #error "Only gpiod.hpp can be included directly."
10 #endif
11
12 #include <memory>
13 #include <ostream>
14
15 namespace gpiod {
16
17     class chip;
18     class line_config;
19     class line_request;
20     class request_config;
21
22     class request_builder final
23     {
24     public:
25         request_builder(const request_builder& other) = delete;
26
27         request_builder(request_builder&& other) noexcept;
28
29         ~request_builder();
30
31         request_builder& operator=(const request_builder& other) = delete;
32
33         request_builder& operator=(request_builder&& other) noexcept;
34
35         request_builder& set_request_config(request_config& req_cfg);
36
37         const request_config& get_request_config() const noexcept;
38
39         request_builder& set_consumer(const ::std::string& consumer) noexcept;
40
41         request_builder& set_event_buffer_size(::std::size_t event_buffer_size) noexcept;
```

```

79
85     request_builder& set_line_config(line_config &line_cfg);
86
92     const line_config& get_line_config() const noexcept;
93
101    request_builder& add_line_settings(line::offset offset, const line_settings& settings);
102
110    request_builder& add_line_settings(const line::offsets& offsets, const line_settings& settings);
111
118    request_builder& set_output_values(const line::values& values);
119
124    line_request do_request();
125
126 private:
127
128     struct impl;
129
130     request_builder(chip& chip);
131
132     ::std::unique_ptr<impl> _m_priv;
133
134     friend chip;
135     friend ::std::ostream& operator<< (::std::ostream& out, const request_builder& builder);
136 };
137
144 ::std::ostream& operator<< (::std::ostream& out, const request_builder& builder);
145
146 } /* namespace gpiod */
147
148 #endif /* __LIBGPIOD_CXX_REQUEST_BUILDER_HPP__ */

```

7.27 gpiodcxx/request-config.hpp File Reference

```

#include <cstdint>
#include <iostream>
#include <memory>
#include <string>
#include "line.hpp"

```

Classes

- class [gpiod::request_config](#)
Stores a set of options passed to the kernel when making a line request.

Functions

- [::std::ostream & gpiod::operator<< \(::std::ostream &out, const request_config &config\)](#)
Stream insertion operator for [request_config](#) objects.

7.27.1 Function Documentation

7.27.1.1 operator<<()

```

::std::ostream & gpiod::operator<< (
    ::std::ostream & out,
    const request_config & config )

```

Stream insertion operator for request_config objects.

Parameters

<i>out</i>	Output stream to write to.
<i>config</i>	request_config to insert into the output stream.

Returns

Reference to out.

7.28 request-config.hpp

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

```

1  /* SPDX-License-Identifier: LGPL-2.1-or-later */
2  /* SPDX-FileCopyrightText: 2021-2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4  #ifndef __LIBGPIOD_CXX_REQUEST_CONFIG_HPP__
5  #define __LIBGPIOD_CXX_REQUEST_CONFIG_HPP__
6
7  #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
8  #error "Only gpiod.hpp can be included directly."
9  #endif
10
11 #include <cstdlib>
12 #include <iostream>
13 #include <memory>
14 #include <string>
15
16 #include "line.hpp"
17
18 namespace gpiod {
19
20 class chip;
21
22 class request_config final
23 {
24 public:
25     request_config();
26
27     request_config(const request_config& other) = delete;
28
29     request_config(request_config&& other) noexcept;
30
31     ~request_config();
32
33     request_config& operator=(request_config&& other) noexcept;
34
35     request_config& set_consumer(const ::std::string& consumer) noexcept;
36
37     ::std::string consumer() const noexcept;
38
39     request_config& set_event_buffer_size(::std::size_t event_buffer_size) noexcept;
40
41     ::std::size_t event_buffer_size() const noexcept;
42
43 private:
44     struct impl;
45
46     ::std::shared_ptr<impl> _m_priv;
47
48     request_config& operator=(const request_config& other);
49
50     friend request_builder;
51 };
52
53 ::std::ostream& operator<< (::std::ostream& out, const request_config& config);
54
55 } /* namespace gpiod */
56
57 #endif /* __LIBGPIOD_CXX_REQUEST_CONFIG_HPP__ */

```

7.29 gpiodcxx/timestamp.hpp File Reference

```
#include <chrono>
#include <cstdint>
```

Classes

- class [gpiod::timestamp](#)

Stores the edge and info event timestamps as returned by the kernel and allows to convert them to `std::chrono::time_point`.

7.30 timestamp.hpp

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

```
1 /* SPDX-License-Identifier: LGPL-2.1-or-later */
2 /* SPDX-FileCopyrightText: 2022 Bartosz Golaszewski <brgl@bgdev.pl> */
3
4 #ifndef __LIBGPIOD_CXX_TIMESTAMP_HPP__
5 #define __LIBGPIOD_CXX_TIMESTAMP_HPP__
6
7 #if !defined(__LIBGPIOD_GPIOD_CXX_INSIDE__)
8 #error "Only gpiod.hpp can be included directly."
9 #endif
10
11 #include <chrono>
12 #include <cstdint>
13
14 namespace gpiod {
15
16     class timestamp final
17     {
18     public:
19         using time_point_monotonic = ::std::chrono::time_point<::std::chrono::steady_clock>;
20         using time_point_realtime = ::std::chrono::time_point<::std::chrono::system_clock,
21                                     ::std::chrono::nanoseconds>;
22
23         timestamp(::std::uint64_t ns) : _m_ns(ns) {}
24
25         timestamp(const timestamp& other) noexcept = default;
26         timestamp(timestamp&& other) noexcept = default;
27         timestamp& operator=(const timestamp& other) noexcept = default;
28         timestamp& operator=(timestamp&& other) noexcept = default;
29         ~timestamp() = default;
30
31         operator ::std::uint64_t() noexcept
32         {
33             return this->ns();
34         }
35
36         ::std::uint64_t ns() const noexcept
37         {
38             return this->_m_ns;
39         }
40
41         time_point_monotonic to_time_point_monotonic() const
42         {
43             return time_point_monotonic(::std::chrono::nanoseconds(this->ns()));
44         }
45
46         time_point_realtime to_time_point_realtime() const
47         {
48             return time_point_realtime(::std::chrono::nanoseconds(this->ns()));
49         }
50
51     private:
52         ::std::uint64_t _m_ns;
53     };
54 #endif
```

```
110 };  
111  
112 } /* namespace gpiod */  
113  
114 #endif /* __LIBGPIOD_CXX_TIMESTAMP_HPP__ */
```


Index

- ACTIVE
 - gpiod::line, 12
- active_low
 - gpiod::line_info, 46
 - gpiod::line_settings, 60
- add_line_settings
 - gpiod::line_config, 43
 - gpiod::request_builder, 71
- api_version
 - misc.hpp, 104
- AS_IS
 - gpiod::line, 10, 11
- bad_mapping
 - gpiod::bad_mapping, 18
- begin
 - gpiod::edge_event_buffer, 36
- bias
 - gpiod::line, 10
 - gpiod::line_info, 47
 - gpiod::line_settings, 60
- BOTH
 - gpiod::line, 12
- capacity
 - gpiod::edge_event_buffer, 36
- chip
 - gpiod::chip, 21
- chip-info.hpp
 - operator<<, 86
- chip.hpp
 - operator<<, 88
- chip_closed
 - gpiod::chip_closed, 26
- chip_info
 - gpiod::chip_info, 28
- chip_name
 - gpiod::line_request, 52
- clock
 - gpiod::line, 10
- close
 - gpiod::chip, 21
- consumer
 - gpiod::line_info, 47
 - gpiod::request_config, 77
- debounce_period
 - gpiod::line_info, 47
 - gpiod::line_settings, 61
- debounced
 - gpiod::line_info, 47
- direction
 - gpiod::line, 11
 - gpiod::line_info, 48
 - gpiod::line_settings, 61
- DISABLED
 - gpiod::line, 10
- do_request
 - gpiod::request_builder, 72
- drive
 - gpiod::line, 11
 - gpiod::line_info, 48
 - gpiod::line_settings, 61
- edge
 - gpiod::line, 11
- edge-event-buffer.hpp
 - operator<<, 89
- edge-event.hpp
 - operator<<, 91
- edge_detection
 - gpiod::line_info, 48
 - gpiod::line_settings, 61
- edge_event
 - gpiod::edge_event, 32
- edge_event_buffer
 - gpiod::edge_event_buffer, 35, 36
- end
 - gpiod::edge_event_buffer, 36
- event_buffer_size
 - gpiod::request_config, 77
- event_clock
 - gpiod::line_info, 48
 - gpiod::line_settings, 62
- event_type
 - gpiod::edge_event, 31
 - gpiod::info_event, 39
- FALLING
 - gpiod::line, 12
- FALLING_EDGE
 - gpiod::edge_event, 31
- fd
 - gpiod::chip, 21
 - gpiod::line_request, 52
- get_event
 - gpiod::edge_event_buffer, 36
- get_info
 - gpiod::chip, 22

- get_line_config
 - gpiod::request_builder, 72
- get_line_info
 - gpiod::chip, 22
 - gpiod::info_event, 40
- get_line_offset_from_name
 - gpiod::chip, 22
- get_line_settings
 - gpiod::line_config, 43
- get_request_config
 - gpiod::request_builder, 72
- get_value
 - gpiod::line_request, 52
- get_values
 - gpiod::line_request, 53, 54
- global_seqno
 - gpiod::edge_event, 32
- gpiod.hpp, 85
- gpiod::bad_mapping, 17
 - bad_mapping, 18
 - operator=, 18, 19
- gpiod::chip, 19
 - chip, 21
 - close, 21
 - fd, 21
 - get_info, 22
 - get_line_info, 22
 - get_line_offset_from_name, 22
 - operator bool, 23
 - operator=, 23
 - path, 23
 - prepare_request, 23
 - read_info_event, 24
 - unwatch_line_info, 24
 - wait_info_event, 24
 - watch_line_info, 25
- gpiod::chip_closed, 25
 - chip_closed, 26
 - operator=, 27
- gpiod::chip_info, 27
 - chip_info, 28
 - label, 29
 - name, 29
 - num_lines, 29
 - operator=, 29, 30
- gpiod::edge_event, 30
 - edge_event, 32
 - event_type, 31
 - FALLING_EDGE, 31
 - global_seqno, 32
 - line_offset, 32
 - line_seqno, 33
 - operator=, 33
 - RISING_EDGE, 31
 - timestamp_ns, 34
 - type, 34
- gpiod::edge_event_buffer, 34
 - begin, 36
 - capacity, 36
 - edge_event_buffer, 35, 36
 - end, 36
 - get_event, 36
 - num_events, 38
 - operator=, 38
- gpiod::info_event, 38
 - event_type, 39
 - get_line_info, 40
 - info_event, 40
 - LINE_CONFIG_CHANGED, 39
 - LINE_RELEASED, 39
 - LINE_REQUESTED, 39
 - operator=, 40, 41
 - timestamp_ns, 41
 - type, 41
- gpiod::line, 9
 - ACTIVE, 12
 - AS_IS, 10, 11
 - bias, 10
 - BOTH, 12
 - clock, 10
 - direction, 11
 - DISABLED, 10
 - drive, 11
 - edge, 11
 - FALLING, 12
 - INACTIVE, 12
 - INPUT, 11
 - MONOTONIC, 11
 - NONE, 12
 - OPEN_DRAIN, 11
 - OPEN_SOURCE, 11
 - operator<<, 12–15
 - OUTPUT, 11
 - PULL_DOWN, 10
 - PULL_UP, 10
 - PUSH_PULL, 11
 - REALTIME, 11
 - RISING, 12
 - UNKNOWN, 10
 - value, 12
- gpiod::line::offset, 67
 - offset, 68
 - operator=, 69
- gpiod::line_config, 42
 - add_line_settings, 43
 - get_line_settings, 43
 - line_config, 42
 - operator=, 44
 - reset, 44
 - set_output_values, 44
- gpiod::line_info, 45
 - active_low, 46
 - bias, 47
 - consumer, 47
 - debounce_period, 47
 - debounced, 47

- direction, [48](#)
- drive, [48](#)
- edge_detection, [48](#)
- event_clock, [48](#)
- line_info, [46](#)
- name, [49](#)
- offset, [49](#)
- operator=, [49](#), [50](#)
- used, [50](#)
- gpio::line_request, [50](#)
 - chip_name, [52](#)
 - fd, [52](#)
 - get_value, [52](#)
 - get_values, [53](#), [54](#)
 - line_request, [52](#)
 - num_lines, [54](#)
 - offsets, [54](#)
 - operator bool, [54](#)
 - operator=, [55](#)
 - read_edge_events, [55](#)
 - reconfigure_lines, [56](#)
 - release, [56](#)
 - set_value, [56](#)
 - set_values, [57](#), [58](#)
 - wait_edge_events, [58](#)
- gpio::line_settings, [58](#)
 - active_low, [60](#)
 - bias, [60](#)
 - debounce_period, [61](#)
 - direction, [61](#)
 - drive, [61](#)
 - edge_detection, [61](#)
 - event_clock, [62](#)
 - line_settings, [60](#)
 - operator=, [62](#)
 - output_value, [63](#)
 - reset, [63](#)
 - set_active_low, [63](#)
 - set_bias, [64](#)
 - set_debounce_period, [64](#)
 - set_direction, [64](#)
 - set_drive, [65](#)
 - set_edge_detection, [65](#)
 - set_event_clock, [65](#)
 - set_output_value, [67](#)
- gpio::request_builder, [70](#)
 - add_line_settings, [71](#)
 - do_request, [72](#)
 - get_line_config, [72](#)
 - get_request_config, [72](#)
 - operator=, [72](#)
 - request_builder, [71](#)
 - set_consumer, [73](#)
 - set_event_buffer_size, [73](#)
 - set_line_config, [73](#)
 - set_output_values, [75](#)
 - set_request_config, [75](#)
- gpio::request_config, [76](#)
 - consumer, [77](#)
 - event_buffer_size, [77](#)
 - operator=, [77](#)
 - request_config, [76](#)
 - set_consumer, [78](#)
 - set_event_buffer_size, [78](#)
- gpio::request_released, [79](#)
 - operator=, [80](#)
 - request_released, [79](#), [80](#)
- gpio::timestamp, [81](#)
 - ns, [83](#)
 - operator=, [83](#)
 - timestamp, [82](#)
 - to_time_point_monotonic, [84](#)
 - to_time_point_realtime, [84](#)
- gpiodcxx/chip-info.hpp, [86](#)
- gpiodcxx/chip.hpp, [87](#), [88](#)
- gpiodcxx/edge-event-buffer.hpp, [89](#), [90](#)
- gpiodcxx/edge-event.hpp, [91](#), [92](#)
- gpiodcxx/exception.hpp, [93](#)
- gpiodcxx/info-event.hpp, [94](#)
- gpiodcxx/line-config.hpp, [95](#)
- gpiodcxx/line-info.hpp, [96](#), [97](#)
- gpiodcxx/line-request.hpp, [98](#), [99](#)
- gpiodcxx/line-settings.hpp, [100](#)
- gpiodcxx/line.hpp, [101](#), [103](#)
- gpiodcxx/misc.hpp, [104](#), [105](#)
- gpiodcxx/request-builder.hpp, [105](#), [106](#)
- gpiodcxx/request-config.hpp, [107](#), [108](#)
- gpiodcxx/timestamp.hpp, [109](#)
- INACTIVE
 - gpio::line, [12](#)
- info_event
 - gpio::info_event, [40](#)
- INPUT
 - gpio::line, [11](#)
- is_gpiochip_device
 - misc.hpp, [104](#)
- label
 - gpio::chip_info, [29](#)
- line-config.hpp
 - operator<<, [95](#)
- line-info.hpp
 - operator<<, [97](#)
- line-request.hpp
 - operator<<, [98](#)
- line_config
 - gpio::line_config, [42](#)
- LINE_CONFIG_CHANGED
 - gpio::info_event, [39](#)
- line_info
 - gpio::line_info, [46](#)
- line_offset
 - gpio::edge_event, [32](#)
- LINE_RELEASED
 - gpio::info_event, [39](#)
- line_request

- gpiod::line_request, 52
- LINE_REQUESTED
 - gpiod::info_event, 39
- line_seqno
 - gpiod::edge_event, 33
- line_settings
 - gpiod::line_settings, 60
- misc.hpp
 - api_version, 104
 - is_gpiochip_device, 104
- MONOTONIC
 - gpiod::line, 11
- name
 - gpiod::chip_info, 29
 - gpiod::line_info, 49
- NONE
 - gpiod::line, 12
- ns
 - gpiod::timestamp, 83
- num_events
 - gpiod::edge_event_buffer, 38
- num_lines
 - gpiod::chip_info, 29
 - gpiod::line_request, 54
- offset
 - gpiod::line::offset, 68
 - gpiod::line_info, 49
- offsets
 - gpiod::line_request, 54
- OPEN_DRAIN
 - gpiod::line, 11
- OPEN_SOURCE
 - gpiod::line, 11
- operator bool
 - gpiod::chip, 23
 - gpiod::line_request, 54
- operator<<
 - chip-info.hpp, 86
 - chip.hpp, 88
 - edge-event-buffer.hpp, 89
 - edge-event.hpp, 91
 - gpiod::line, 12–15
 - line-config.hpp, 95
 - line-info.hpp, 97
 - line-request.hpp, 98
 - request-builder.hpp, 106
 - request-config.hpp, 107
- operator=
 - gpiod::bad_mapping, 18, 19
 - gpiod::chip, 23
 - gpiod::chip_closed, 27
 - gpiod::chip_info, 29, 30
 - gpiod::edge_event, 33
 - gpiod::edge_event_buffer, 38
 - gpiod::info_event, 40, 41
 - gpiod::line::offset, 69
 - gpiod::line_config, 44
 - gpiod::line_info, 49, 50
 - gpiod::line_request, 55
 - gpiod::line_settings, 62
 - gpiod::request_builder, 72
 - gpiod::request_config, 77
 - gpiod::request_released, 80
 - gpiod::timestamp, 83
- OUTPUT
 - gpiod::line, 11
- output_value
 - gpiod::line_settings, 63
- path
 - gpiod::chip, 23
- prepare_request
 - gpiod::chip, 23
- PULL_DOWN
 - gpiod::line, 10
- PULL_UP
 - gpiod::line, 10
- PUSH_PULL
 - gpiod::line, 11
- read_edge_events
 - gpiod::line_request, 55
- read_info_event
 - gpiod::chip, 24
- REALTIME
 - gpiod::line, 11
- reconfigure_lines
 - gpiod::line_request, 56
- release
 - gpiod::line_request, 56
- request-builder.hpp
 - operator<<, 106
- request-config.hpp
 - operator<<, 107
- request_builder
 - gpiod::request_builder, 71
- request_config
 - gpiod::request_config, 76
- request_released
 - gpiod::request_released, 79, 80
- reset
 - gpiod::line_config, 44
 - gpiod::line_settings, 63
- RISING
 - gpiod::line, 12
- RISING_EDGE
 - gpiod::edge_event, 31
- set_active_low
 - gpiod::line_settings, 63
- set_bias
 - gpiod::line_settings, 64
- set_consumer
 - gpiod::request_builder, 73
 - gpiod::request_config, 78

- set_debounce_period
 - gpiod::line_settings, 64
- set_direction
 - gpiod::line_settings, 64
- set_drive
 - gpiod::line_settings, 65
- set_edge_detection
 - gpiod::line_settings, 65
- set_event_buffer_size
 - gpiod::request_builder, 73
 - gpiod::request_config, 78
- set_event_clock
 - gpiod::line_settings, 65
- set_line_config
 - gpiod::request_builder, 73
- set_output_value
 - gpiod::line_settings, 67
- set_output_values
 - gpiod::line_config, 44
 - gpiod::request_builder, 75
- set_request_config
 - gpiod::request_builder, 75
- set_value
 - gpiod::line_request, 56
- set_values
 - gpiod::line_request, 57, 58
- timestamp
 - gpiod::timestamp, 82
- timestamp_ns
 - gpiod::edge_event, 34
 - gpiod::info_event, 41
- to_time_point_monotonic
 - gpiod::timestamp, 84
- to_time_point_realtime
 - gpiod::timestamp, 84
- type
 - gpiod::edge_event, 34
 - gpiod::info_event, 41
- UNKNOWN
 - gpiod::line, 10
- unwatch_line_info
 - gpiod::chip, 24
- used
 - gpiod::line_info, 50
- value
 - gpiod::line, 12
- wait_edge_events
 - gpiod::line_request, 58
- wait_info_event
 - gpiod::chip, 24
- watch_line_info
 - gpiod::chip, 25