libremidi

Jean-Michaël Celerier

Mar 17, 2021

CONTENTS

1 libremidi API **1**

1.1 Class Hierarchy 1

1.2 File Hierarchy 1

1.3 Full API 1

2 Indices and tables **31**

Index **33**

CHAPTER

ONE

LIBREMIDI API

1.1 Class Hierarchy

1.2 File Hierarchy

1.3 Full API

1.3.1 Namespaces

Namespace libremidi

Contents

- *Namespaces*
- *Classes*
- *Enums*
- *Functions*
- *Typedefs*
- *Variables*

Namespaces

- *Namespace libremidi::util*

Classes

- *Struct chunking_parameters*
- *Struct driver_error*
- *Struct invalid_device_error*
- *Struct invalid_parameter_error*
- *Struct invalid_use_error*
- *Struct memory_error*
- *Struct message*
- *Struct meta_events*
- *Struct midi_exception*
- *Struct no_devices_found_error*
- *Struct observer::callbacks*
- *Struct system_error*
- *Struct thread_error*
- *Struct track_event*
- *Class midi_in*
- *Class midi_out*
- *Class observer*
- *Class reader*
- *Class writer*

Enums

- *Enum API*
- *Enum message_type*
- *Enum meta_event_type*
- *Enum midi_error*

Functions

- *Function libremidi::available_apis*
- *Function libremidi::clamp*
- *Template Function libremidi::for_all_backends*
- *Template Function libremidi::for_backend*
- *Function libremidi::get_version*
- *Template Function libremidi::make_tl*
- *Function libremidi::open_midi_in*

- *Function libremidi::open_midi_observer*
- *Function libremidi::open_midi_out*
- *Function libremidi::parseEvent*

Typedefs

- *Typedef libremidi::midi_bytes*
- *Typedef libremidi::midi_error_callback*
- *Typedef libremidi::midi_track*

Variables

- *Variable libremidi::available_backends*

Namespace libremidi::util

Contents

- *Functions*

Functions

- *Function libremidi::util::read_bytes*
- *Function libremidi::util::read_uint16_be*
- *Function libremidi::util::read_uint24_be*
- *Function libremidi::util::read_uint32_be*
- *Function libremidi::util::read_variable_length*
- *Function libremidi::util::write_double_be*
- *Function libremidi::util::write_float_be*
- *Function libremidi::util::write_int16_be*
- *Function libremidi::util::write_int32_be*
- *Function libremidi::util::write_uint16_be*
- *Function libremidi::util::write_uint32_be*
- *Function libremidi::util::write_variable_length*

1.3.2 Classes and Structs

Struct chunking_parameters

- Defined in file_include_libremidi_libremidi.hpp

Struct Documentation

struct libremidi::**chunking_parameters**

Used to determine how large sent messages will be chunked.

Public Members

std::chrono::milliseconds **interval** = {}

int32_t **size** = {}

std::function<bool (std::chrono::microseconds, int) > **wait** = *chunking_parameters*::*default_wait*

Will be called by the chunking code to allow the API user to wait.

By default just calls sleep. Arguments are: the time that must be waited, the bytes currently written. Return false if you want to abort the transfer, and true otherwise.

Public Static Functions

static inline bool **default_wait** (std::chrono::microseconds *time_to_wait*, int *written_bytes*)

Struct driver_error

- Defined in file_include_libremidi_libremidi.hpp

Inheritance Relationships

Base Type

- public libremidi::midi_exception (*Struct midi_exception*)

Struct Documentation

struct libremidi::**driver_error** : **public** libremidi::*midi_exception*

Public Functions

inline ~driver_error() override

Public Static Attributes

static constexpr auto **code** = *midi_error*::*DRIVER_ERROR*

Struct invalid_device_error

- Defined in file_include_libremidi_libremidi.hpp

Inheritance Relationships

Base Type

- `public libremidi::midi_exception` (*Struct midi_exception*)

Struct Documentation

struct `libremidi::`**invalid_device_error** : **public** libremidi::*midi_exception*

Public Functions

inline ~invalid_device_error() override

Public Static Attributes

static constexpr auto **code** = *midi_error*::*INVALID_DEVICE*

Struct invalid_parameter_error

- Defined in file_include_libremidi_libremidi.hpp

Inheritance Relationships

Base Type

- `public libremidi::midi_exception` (*Struct midi_exception*)

Struct Documentation

struct libremidi::**invalid_parameter_error** : **public** libremidi::*midi_exception*

Public Functions

inline **~invalid_parameter_error**() **override**

Public Static Attributes

static **constexpr** auto **code** = *midi_error*::*INVALID_PARAMETER*

Struct invalid_use_error

- Defined in file_include_libremidi_libremidi.hpp

Inheritance Relationships

Base Type

- public libremidi::midi_exception (*Struct midi_exception*)

Struct Documentation

struct libremidi::**invalid_use_error** : **public** libremidi::*midi_exception*

Public Functions

inline **~invalid_use_error**() **override**

Public Static Attributes

static **constexpr** auto **code** = *midi_error*::*INVALID_USE*

Struct memory_error

- Defined in file_include_libremidi_libremidi.hpp

Inheritance Relationships

Base Type

- `public libremidi::midi_exception` (*Struct midi_exception*)

Struct Documentation

struct `libremidi::memory_error` : **public** libremidi::*midi_exception*

Public Functions

inline **~memory_error**() **override**

Public Static Attributes

static **constexpr** auto **code** = *midi_error*::*MEMORY_ERROR*

Struct message

- Defined in file_include_libremidi_message.hpp

Struct Documentation

struct `libremidi::message`

Public Functions

message() **noexcept** = default

inline **message**(**const** *midi_bytes* &*src_bytes*, double *src_timestamp*)

template<typename ...**Args**>
inline **message**(*Args*... *args*) **noexcept**

inline bool **uses_channel**(int *channel*) **const**

inline int **get_channel**() **const**

inline bool **is_meta_event**() **const**

inline *meta_event_type* **get_meta_event_type**() **const**

inline *message_type* **get_message_type**() **const**

inline bool **is_note_on_or_off**() **const**

inline auto **size**() **const**

inline auto &**front**() **const**

inline auto &**back**() **const**

inline auto &**operator[]**(int *i*) **const**

inline auto &**front**()

inline auto &**back**()

inline auto &**operator[]**(int *i*)

template<typename ...**Args**>
inline auto **assign**(*Args*&&... *args*)

template<typename ...**Args**>
inline auto **insert**(*Args*&&... *args*)

inline auto **clear**()

inline auto **begin**() **const**

inline auto **end**() **const**

inline auto **begin**()

inline auto **end**()

inline auto **cbegin**() **const**

inline auto **cend**() **const**

inline auto **cbegin**()

inline auto **cend**()

inline auto **rbegin**() **const**

inline auto **rend**() **const**

inline auto **rbegin**()

inline auto **rend**()

Public Members

midi_bytes **bytes**

double **timestamp** = {}

Public Static Functions

static inline uint8_t **make_command**(**const** *message_type* *type*, **const** int *channel*) **noexcept**

static inline *message* **note_on**(uint8_t *channel*, uint8_t *note*, uint8_t *velocity*) **noexcept**

static inline *message* **note_off**(uint8_t *channel*, uint8_t *note*, uint8_t *velocity*) **noexcept**

static inline *message* **control_change**(uint8_t *channel*, uint8_t *control*, uint8_t *value*) **noexcept**

static inline *message* **program_change**(uint8_t *channel*, uint8_t *value*) **noexcept**

static inline *message* **pitch_bend**(uint8_t *channel*, int *value*) **noexcept**

static inline *message* **pitch_bend**(uint8_t *channel*, uint8_t *lsb*, uint8_t *msb*) **noexcept**

static inline *message* **poly_pressure**(uint8_t *channel*, uint8_t *note*, uint8_t *value*) **noexcept**

static inline *message* **aftertouch**(uint8_t *channel*, uint8_t *value*) **noexcept**

Struct meta_events

- Defined in file_include_libremidi_message.hpp

Struct Documentation

struct libremidi::**meta_events**

Public Static Functions

static inline *message* **end_of_track**()

static inline *message* **channel**(int *channel*)

static inline *message* **tempo**(int *mpqn*)

static inline *message* **time_signature**(int *numerator*, int *denominator*)

static inline *message* **key_signature**(int *keyIndex*, bool *isMinor*)

static inline *message* **song_position**(int *positionInBeats*) **noexcept**

Struct midi_exception

- Defined in file_include_libremidi_libremidi.hpp

Inheritance Relationships

Base Type

- public runtime_error

Derived Types

- public libremidi::driver_error (*Struct driver_error*)
- public libremidi::invalid_device_error (*Struct invalid_device_error*)
- public libremidi::invalid_parameter_error (*Struct invalid_parameter_error*)
- public libremidi::invalid_use_error (*Struct invalid_use_error*)
- public libremidi::memory_error (*Struct memory_error*)
- public libremidi::no_devices_found_error (*Struct no_devices_found_error*)
- public libremidi::system_error (*Struct system_error*)
- public libremidi::thread_error (*Struct thread_error*)

Struct Documentation

struct libremidi::**midi_exception** : **public** runtime_error

Base exception class for MIDI problems.

Subclassed by *libremidi::driver_error*, *libremidi::invalid_device_error*, *libremidi::invalid_parameter_error*, *libremidi::invalid_use_error*, *libremidi::memory_error*, *libremidi::no_devices_found_error*, *libremidi::system_error*, *libremidi::thread_error*

Public Functions

inline ~midi_exception() **override**

Struct no_devices_found_error

- Defined in file_include_libremidi_libremidi.hpp

Inheritance Relationships

Base Type

- public libremidi::midi_exception (*Struct midi_exception*)

Struct Documentation

struct libremidi::**no_devices_found_error** : **public** libremidi::*midi_exception*

Public Functions

inline ~no_devices_found_error() **override**

Public Static Attributes

static constexpr auto **code** = *midi_error*::*NO_DEVICES_FOUND*

Struct observer::callbacks

- Defined in file_include_libremidi_libremidi.hpp

Nested Relationships

This struct is a nested type of *Class observer*.

Struct Documentation

struct libremidi::*observer*::**callbacks**

Public Members

std::function<void (int, std::string) > **input_added**

std::function<void (int, std::string) > **input_removed**

std::function<void (int, std::string) > **output_added**

std::function<void (int, std::string) > **output_removed**

Struct system_error

- Defined in file_include_libremidi_libremidi.hpp

Inheritance Relationships

Base Type

- public libremidi::midi_exception (*Struct midi_exception*)

Struct Documentation

struct libremidi::**system_error** : **public** libremidi::*midi_exception*

Public Functions

inline **~system_error**() **override**

Public Static Attributes

static **constexpr** auto **code** = *midi_error*::*SYSTEM_ERROR*

Struct thread_error

- Defined in file_include_libremidi_libremidi.hpp

Inheritance Relationships

Base Type

- `public libremidi::midi_exception` (*Struct midi_exception*)

Struct Documentation

struct `libremidi::`**thread_error** : **public** libremidi::*midi_exception*

Public Functions

inline **~thread_error**() **override**

Public Static Attributes

static **constexpr** auto **code** = *midi_error*::*THREAD_ERROR*

Struct track_event

- Defined in file_include_libremidi_message.hpp

Struct Documentation

struct `libremidi::`**track_event**

Public Members

int **tick** = 0

int **track** = 0

message **m**

Class midi_in

- Defined in file_include_libremidi_libremidi.hpp

Class Documentation

class libremidi::**midi_in**

A realtime MIDI input class.

This class provides a common, platform-independent API for realtime MIDI input. It allows access to a single MIDI input port. Incoming MIDI messages are either saved to a queue for retrieval using the getMessage() function or immediately passed to a user-specified callback function. Create multiple instances of this class to connect to more than one MIDI device at the same time. With the OS-X, Linux ALSA, and JACK MIDI APIs, it is also possible to open a virtual input port to which other MIDI software clients can connect.

by Gary P. Scavone, 2003-2017.

Public Types

using **message_callback** = std::function<void (**const** *message* &message) >

User callback function type definition.

Public Functions

inline **midi_in** (libremidi::*API* *api* = *API*::*UNSPECIFIED*, std::string_view *clientName* = "RtMidi Input Client", unsigned int *queueSizeLimit* = 100)

Default constructor that allows an optional api, client name and queue size.

An exception will be thrown if a MIDI system initialization error occurs. The queue size defines the maximum number of messages that can be held in the MIDI queue (when not using a callback function). If the queue size limit is reached, incoming messages will be ignored.

If no API argument is specified and multiple API support has been compiled, the default order of use is ALSA, JACK (Linux) and CORE, JACK (OS-X).

Parameters

- `api`: An optional API id can be specified.
- `clientName`: An optional client name can be specified. This will be used to group the ports that are created by the application.
- `queueSizeLimit`: An optional size of the MIDI input queue can be specified.

inline **~midi_in** ()

If a MIDI connection is still open, it will be closed by the destructor.

inline libremidi::*API* **get_current_api** () **const** **noexcept**

Returns the MIDI API specifier for the current instance of RtMidiIn.

inline void **open_port** (unsigned int *portNumber*, std::string_view *portName*)

Open a MIDI input connection given by enumeration number.

Parameters

- `portNumber`: A port number greater than 0 can be specified. Otherwise, the default or first port found is opened.
- `portName`: A name for the application port that is used to connect to portId can be specified.

inline void **open_port** ()

inline void **open_port**(unsigned int *port*)

inline void **open_virtual_port**(std::string_view *portName*)

Create a virtual input port, with optional name, to allow software connections (OS X, JACK and ALSA only).

This function creates a virtual MIDI input port to which other software applications can connect. This type of functionality is currently only supported by the Macintosh OS-X, any JACK, and Linux ALSA APIs (the function returns an error for the other APIs).

Parameters

- `portName`: An optional name for the application port that is used to connect to portId can be specified.

inline void **open_virtual_port**()

inline void **set_callback**(*message_callback callback*)

Set a callback function to be invoked for incoming MIDI messages.

The callback function will be called whenever an incoming MIDI message is received. While not absolutely necessary, it is best to set the callback function before opening a MIDI port to avoid leaving some messages in the queue.

Parameters

- `callback`: A callback function must be given.
- `userData`: Optionally, a pointer to additional data can be passed to the callback function whenever it is called.

inline void **cancel_callback**()

Cancel use of the current callback function (if one exists).

Subsequent incoming MIDI messages will be written to the queue and can be retrieved with the *getMessage* function.

inline void **close_port**()

Close an open MIDI connection (if one exists).

inline bool **is_port_open**() **const noexcept**

Returns true if a port is open and false if not.

Note that this only applies to connections made with the openPort() function, not to virtual ports.

inline unsigned int **get_port_count**()

Return the number of available MIDI input ports.

Return This function returns the number of MIDI ports of the selected API.

inline std::string **get_port_name**(unsigned int *portNumber* = 0)

Return a string identifier for the specified MIDI input port number.

Return The name of the port with the given Id is returned. An empty string is returned if an invalid port specifier is provided. User code should assume a UTF-8 encoding.

inline void **ignore_types**(bool *midiSysex* = true, bool *midiTime* = true, bool *midiSense* = true)
Specify whether certain MIDI message types should be queued or ignored during input.

By default, MIDI timing and active sensing messages are ignored during message input because of their relative high data rates. MIDI sysex messages are ignored by default as well. Variable values of "true" imply that the respective message type will be ignored.

inline *message* **get_message**()
Fill the user-provided vector with the data bytes for the next available MIDI message in the input queue and return the event delta-time in seconds.

This function returns immediately whether a new message is available or not. A valid message is indicated by a non-zero vector size. An exception is thrown if an error occurs during message retrieval or an input connection was not previously established.

inline bool **get_message**(*message*&)

inline void **set_error_callback**(*midi_error_callback errorCallback*)
Set an error callback function to be invoked when an error has occured.

The callback function will be called whenever an error has occured. It is best to set the error callback function before opening a port.

inline void **set_client_name**(std::string_view *clientName*)

inline void **set_port_name**(std::string_view *portName*)

Class midi_out

- Defined in file_include_libremidi_libremidi.hpp

Class Documentation

class libremidi::**midi_out**
A realtime MIDI output class.

This class provides a common, platform-independent API for MIDI output. It allows one to probe available MIDI output ports, to connect to one such port, and to send MIDI bytes immediately over the connection. Create multiple instances of this class to connect to more than one MIDI device at the same time. With the OS-X, Linux ALSA and JACK MIDI APIs, it is also possible to open a virtual port to which other MIDI software clients can connect.

by Gary P. Scavone, 2003-2017.

Public Functions

inline **midi_out**(libremidi::*API api*, std::string_view *clientName*)
Default constructor that allows an optional client name.

An exception will be thrown if a MIDI system initialization error occurs.

If no API argument is specified and multiple API support has been compiled, the default order of use is ALSA, JACK (Linux) and CORE, JACK (OS-X).

inline **midi_out**()

inline **~midi_out**()
The destructor closes any open MIDI connections.

inline libremidi::*API* **get_current_api**() **noexcept**
Returns the MIDI API specifier for the current instance of RtMidiOut.

inline void **open_port**(unsigned int *portNumber*, std::string_view *portName*)
Open a MIDI output connection.

An optional port number greater than 0 can be specified. Otherwise, the default or first port found is opened. An exception is thrown if an error occurs while attempting to make the port connection.

inline void **open_port**()

inline void **open_port**(unsigned int *port*)

inline void **close_port**()
Close an open MIDI connection (if one exists).

inline bool **is_port_open**() **const noexcept**
Returns true if a port is open and false if not.

Note that this only applies to connections made with the openPort() function, not to virtual ports.

inline void **open_virtual_port**(std::string_view *portName*)
Create a virtual output port, with optional name, to allow software connections (OS X, JACK and ALSA only).

This function creates a virtual MIDI output port to which other software applications can connect. This type of functionality is currently only supported by the Macintosh OS-X, Linux ALSA and JACK APIs (the function does nothing with the other APIs). An exception is thrown if an error occurs while attempting to create the virtual port.

inline void **open_virtual_port**()

inline unsigned int **get_port_count**()
Return the number of available MIDI output ports.

inline std::string **get_port_name**(unsigned int *portNumber* = 0)
Return a string identifier for the specified MIDI port type and number.

Return The name of the port with the given Id is returned. An empty string is returned if an invalid port specifier is provided. User code should assume a UTF-8 encoding.

inline void **send_message**(**const** std::vector<unsigned char> &*message*)
Immediately send a single message out an open MIDI output port.

An exception is thrown if an error occurs during output or an output connection was not previously established.

inline void **send_message**(**const** libremidi::*message* &*message*)

inline void **send_message**(**const** unsigned char **message*, size_t *size*)
Immediately send a single message out an open MIDI output port.

An exception is thrown if an error occurs during output or an output connection was not previously established.

Parameters

- `message`: A pointer to the MIDI message as raw bytes
- `size`: Length of the MIDI message in bytes

inline void **set_error_callback**(*midi_error_callback errorCallback*) **noexcept**
Set an error callback function to be invoked when an error has occured.

The callback function will be called whenever an error has occured. It is best to set the error callback function before opening a port.

inline void **set_client_name**(std::string_view *clientName*)

inline void **set_port_name**(std::string_view *portName*)

inline void **set_chunking_parameters**(std::optional<*chunking_parameters*> *parameters*)
For large messages, chunk their content and wait. Setting a null optional will disable chunking.

Class observer

- Defined in file_include_libremidi_libremidi.hpp

Nested Relationships

Nested Types

- *Struct observer::callbacks*

Class Documentation

class libremidi::**observer**
The callbacks will be called whenever a device is added or removed for a given API.

Public Functions

inline observer(libremidi::*API*, *callbacks*)

inline ~observer()

struct callbacks

Public Members

std::function<void(int, std::string)> **input_added**

std::function<void(int, std::string)> **input_removed**

std::function<void(int, std::string)> **output_added**

std::function<void(int, std::string)> **output_removed**

Class reader

- Defined in file_include_libremidi_reader.hpp

Class Documentation

class libremidi::**reader**

Public Functions

inline **reader**(bool *useAbsolute* = false)

inline **~reader**()

inline void **parse**(**const** std::vector<uint8_t> &*buffer*)

inline double **get_end_time**()

Public Members

float **ticksPerBeat** = {}

float **startingTempo** = {}

std::vector<*midi_track*> **tracks**

Class writer

- Defined in file_include_libremidi_writer.hpp

Class Documentation

class libremidi::**writer**

Public Functions

inline **writer**(int *ticks*)

inline **~writer**()

inline size_t **get_num_tracks**()

inline void **add_event**(int *tick*, int *track*, *message* *m*)

inline void **add_event**(int *track*, *track_event* *m*)

inline void **add_track**()

inline void **write**(std::ostream &*out*)

inline **const** std::vector<*midi_track*> &**get_tracks**()

1.3.3 Enums

Enum API

- Defined in file_include_libremidi_libremidi.hpp

Enum Documentation

enum libremidi::**API**
MIDI API specifier arguments.

Values:

enumerator UNSPECIFIED
Search for a working compiled API.

enumerator MACOSX_CORE
Macintosh OS-X Core Midi API.

enumerator LINUX_ALSA
The Advanced Linux Sound Architecture API.

enumerator LINUX_ALSA_SEQ

enumerator LINUX_ALSA_RAW
Raw ALSA API.

enumerator UNIX_JACK
The JACK Low-Latency MIDI Server API.

enumerator WINDOWS_MM
The Microsoft Multimedia MIDI API.

enumerator WINDOWS_UWP
The Microsoft WinRT MIDI API.

enumerator EMSCRIPTEN_WEBMIDI
Web MIDI API through Emscripten

enumerator DUMMY
A compilable but non-functional API.

Enum message_type

- Defined in file_include_libremidi_message.hpp

Enum Documentation

enum libremidi::**message_type**
Values:

enumerator INVALID

enumerator NOTE_OFF

enumerator NOTE_ON

enumerator POLY_PRESSURE

enumerator CONTROL_CHANGE

enumerator PROGRAM_CHANGE

enumerator AFTERTOUCH

enumerator PITCH_BEND

enumerator SYSTEM_EXCLUSIVE

enumerator TIME_CODE

enumerator SONG_POS_POINTER

enumerator SONG_SELECT

enumerator RESERVED1

enumerator RESERVED2

enumerator TUNE_REQUEST

enumerator EOX

enumerator TIME_CLOCK

enumerator RESERVED3

enumerator START

enumerator CONTINUE

enumerator STOP

enumerator RESERVED4

enumerator ACTIVE_SENSING

enumerator SYSTEM_RESET

Enum meta_event_type

- Defined in file_include_libremidi_message.hpp

Enum Documentation

enum libremidi::meta_event_type

Values:

enumerator SEQUENCE_NUMBER

enumerator TEXT

enumerator COPYRIGHT

enumerator TRACK_NAME

enumerator INSTRUMENT

enumerator LYRIC

enumerator MARKER

enumerator CUE

enumerator PATCH_NAME

enumerator DEVICE_NAME

enumerator CHANNEL_PREFIX

enumerator MIDI_PORT

enumerator END_OF_TRACK

enumerator TEMPO_CHANGE

enumerator SMPTE_OFFSET

enumerator TIME_SIGNATURE

enumerator KEY_SIGNATURE

enumerator PROPRIETARY

enumerator UNKNOWN

Enum midi_error

- Defined in file_include_libremidi_libremidi.hpp

Enum Documentation

enum libremidi::**midi_error**

Defines various error types.

Values:

enumerator WARNING
A non-critical error.

enumerator UNSPECIFIED
The default, unspecified error type.

enumerator NO_DEVICES_FOUND
No devices found on system.

enumerator INVALID_DEVICE
An invalid device ID was specified.

enumerator MEMORY_ERROR
An error occured during memory allocation.

enumerator INVALID_PARAMETER
An invalid parameter was specified to a function.

enumerator INVALID_USE
The function was called incorrectly.

enumerator DRIVER_ERROR
A system driver error occured.

enumerator SYSTEM_ERROR
A system error occured.

enumerator THREAD_ERROR
A thread error occured.

1.3.4 Functions

Function libremidi::available_apis

- Defined in file_include_libremidi_libremidi.cpp

Function Documentation

Warning: doxygenfunction: Unable to resolve function "libremidi::available_apis" with arguments () in doxygen xml output for project "libremidi" from directory: ./doxyoutput/xml. Potential matches:

```
- std::vector<libremidi::API> available_apis() noexcept
```

Function libremidi::clamp

- Defined in file_include_libremidi_message.hpp

Function Documentation

inline constexpr uint8_t libremidi::**clamp**(uint8_t *val*, uint8_t *min*, uint8_t *max*)

Template Function libremidi::for_all_backends

- Defined in file_include_libremidi_libremidi.cpp

Function Documentation

template<typename **F**>
auto libremidi::**for_all_backends**(*F* &&*f*)

Template Function libremidi::for_backend

- Defined in file_include_libremidi_libremidi.cpp

Function Documentation

template<typename **F**>
auto libremidi::**for_backend**(libremidi::*API api*, *F* &&*f*)

Function libremidi::get_version

- Defined in file_include_libremidi_libremidi.cpp

Function Documentation

Warning: doxygenfunction: Unable to resolve function "libremidi::get_version" with arguments () in doxygen xml output for project "libremidi" from directory: ./doxyoutput/xml. Potential matches:

```
- std::string get_version() noexcept
```

Template Function libremidi::make_tl

- Defined in file_include_libremidi_libremidi.cpp

Function Documentation

template<typename **unused**, typename ...**Args**>
constexpr auto libremidi::**make_tl**(*unused*, *Args*...)

Function libremidi::open_midi_in

- Defined in file_include_libremidi_libremidi.cpp

Function Documentation

inline std::unique_ptr<midi_in_api> libremidi::**open_midi_in**(libremidi::*API* *api*, std::string_view *clientName*, unsigned int *queueSizeLimit*)

Function libremidi::open_midi_observer

- Defined in file_include_libremidi_libremidi.cpp

Function Documentation

inline std::unique_ptr<observer_api> libremidi::**open_midi_observer**(libremidi::*API* *api*, *observer*::*callbacks* &&*cb*)

Function libremidi::open_midi_out

- Defined in file_include_libremidi_libremidi.cpp

Function Documentation

inline std::unique_ptr<midi_out_api> libremidi::**open_midi_out**(libremidi::*API* *api*, std::string_view *clientName*)

Function libremidi::parseEvent

- Defined in file_include_libremidi_reader.cpp

Function Documentation

inline *track_event* libremidi::**parseEvent**(int *tick*, int *track*, uint8_t **const** *&*dataStart*, *message_type* *lastEventTypeByte*)

Function libremidi::util::read_bytes

- Defined in file_include_libremidi_reader.cpp

Function Documentation

inline void libremidi::util::**read_bytes**(*midi_bytes* &*buffer*, uint8_t **const** *&*data*, int *num*)

Function libremidi::util::read_uint16_be

- Defined in file_include_libremidi_reader.cpp

Function Documentation

inline uint16_t libremidi::util::**read_uint16_be**(uint8_t **const** *&*data*)

Function libremidi::util::read_uint24_be

- Defined in file_include_libremidi_reader.cpp

Function Documentation

inline uint32_t libremidi::util::**read_uint24_be**(uint8_t **const** *&*data*)

Function libremidi::util::read_uint32_be

- Defined in file_include_libremidi_reader.cpp

Function Documentation

inline uint32_t libremidi::util::**read_uint32_be**(uint8_t **const** *&*data*)

Function libremidi::util::read_variable_length

- Defined in file_include_libremidi_reader.cpp

Function Documentation

inline uint32_t libremidi::util::**read_variable_length**(uint8_t **const** *&*data*)

Function libremidi::util::write_double_be

- Defined in file_include_libremidi_writer.cpp

Function Documentation

inline std::ostream &libremidi::util::**write_double_be**(std::ostream &*out*, double *value*)

Function libremidi::util::write_float_be

- Defined in file_include_libremidi_writer.cpp

Function Documentation

inline std::ostream &libremidi::util::**write_float_be**(std::ostream &*out*, float *value*)

Function libremidi::util::write_int16_be

- Defined in file_include_libremidi_writer.cpp

Function Documentation

inline std::ostream &libremidi::util::**write_int16_be**(std::ostream &*out*, int16_t *value*)

Function libremidi::util::write_int32_be

- Defined in file_include_libremidi_writer.cpp

Function Documentation

inline std::ostream &libremidi::util::**write_int32_be**(std::ostream &*out*, int32_t *value*)

Function libremidi::util::write_uint16_be

- Defined in file_include_libremidi_writer.cpp

Function Documentation

inline std::ostream &libremidi::util::**write_uint16_be**(std::ostream &*out*, uint16_t *value*)

Function libremidi::util::write_uint32_be

- Defined in file_include_libremidi_writer.cpp

Function Documentation

inline std::ostream &libremidi::util::**write_uint32_be**(std::ostream &*out*, uint32_t *value*)

Function libremidi::util::write_variable_length

- Defined in file_include_libremidi_writer.cpp

Function Documentation

inline void libremidi::util::**write_variable_length**(uint32_t *aValue*, std::vector<uint8_t> &*outdata*)

1.3.5 Variables

Variable libremidi::available_backends

- Defined in file_include_libremidi_libremidi.cpp

Variable Documentation

static constexpr auto libremidi::**available_backends** = *make_tl*(0, dummy_backend{})

1.3.6 Defines

Define LIBREMIDI_DUMMY

- Defined in file_include_libremidi_libremidi.cpp

Define Documentation

Warning: doxygendefine: Cannot find define "LIBREMIDI_DUMMY" in doxygen xml output for project "libremidi" from directory: ./doxyoutput/xml

Define LIBREMIDI_EXPORT

- Defined in file_include_libremidi_libremidi.hpp

Define Documentation

LIBREMIDI_EXPORT

Define LIBREMIDI_INLINE

- Defined in file_include_libremidi_message.hpp

Define Documentation

LIBREMIDI_INLINE

Define LIBREMIDI_VERSION

- Defined in file_include_libremidi_libremidi.hpp

Define Documentation

LIBREMIDI_VERSION

1.3.7 Typedefs

Typedef libremidi::midi_bytes

- Defined in file_include_libremidi_message.hpp

Typedef Documentation

using libremidi**::midi_bytes** = std::vector<unsigned char>

Typedef libremidi::midi_error_callback

- Defined in file_include_libremidi_libremidi.hpp

Typedef Documentation

using libremidi**::midi_error_callback** = std::function<void (*midi_error* type, std::string_view errorText) >

Error callback function.

Note that class behaviour is undefined after a critical error (not a warning) is reported.

Parameters

- `type`: Type of error.
- `errorText`: Error description.

Typedef libremidi::midi_track

- Defined in file_include_libremidi_message.hpp

Typedef Documentation

typedef std::vector<*track_event*> libremidi::**midi_track**

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

L

libremidi::API (*C++ enum*), 19
libremidi::API::DUMMY (*C++ enumerator*), 19
libremidi::API::EMSCRIPTEN_WEBMIDI (*C++ enumerator*), 19
libremidi::API::LINUX_ALSA (*C++ enumerator*), 19
libremidi::API::LINUX_ALSA_RAW (*C++ enumerator*), 19
libremidi::API::LINUX_ALSA_SEQ (*C++ enumerator*), 19
libremidi::API::MACOSX_CORE (*C++ enumerator*), 19
libremidi::API::UNIX_JACK (*C++ enumerator*), 19
libremidi::API::UNSPECIFIED (*C++ enumerator*), 19
libremidi::API::WINDOWS_MM (*C++ enumerator*), 19
libremidi::API::WINDOWS_UWP (*C++ enumerator*), 19
libremidi::available_backends (*C++ member*), 27
libremidi::chunking_parameters (*C++ struct*), 4
libremidi::chunking_parameters::default_wait (*C++ function*), 4
libremidi::chunking_parameters::interval (*C++ member*), 4
libremidi::chunking_parameters::size (*C++ member*), 4
libremidi::chunking_parameters::wait (*C++ member*), 4
libremidi::clamp (*C++ function*), 22
libremidi::driver_error (*C++ struct*), 4
libremidi::driver_error::~driver_error (*C++ function*), 5
libremidi::driver_error::code (*C++ member*), 5
libremidi::for_all_backends (*C++ function*), 22
libremidi::for_backend (*C++ function*), 22
libremidi::invalid_device_error (*C++ struct*), 5
libremidi::invalid_device_error::~invalid_device_er (*C++ function*), 5
libremidi::invalid_device_error::code (*C++ member*), 5
libremidi::invalid_parameter_error (*C++ struct*), 6
libremidi::invalid_parameter_error::~invalid_parame (*C++ function*), 6
libremidi::invalid_parameter_error::code (*C++ member*), 6
libremidi::invalid_use_error (*C++ struct*), 6
libremidi::invalid_use_error::~invalid_use_error (*C++ function*), 6
libremidi::invalid_use_error::code (*C++ member*), 6
libremidi::make_tl (*C++ function*), 23
libremidi::memory_error (*C++ struct*), 7
libremidi::memory_error::~memory_error (*C++ function*), 7
libremidi::memory_error::code (*C++ member*), 7
libremidi::message (*C++ struct*), 7
libremidi::message::aftertouch (*C++ function*), 8
libremidi::message::assign (*C++ function*), 8
libremidi::message::back (*C++ function*), 7, 8
libremidi::message::begin (*C++ function*), 8
libremidi::message::bytes (*C++ member*), 8
libremidi::message::cbegin (*C++ function*), 8
libremidi::message::cend (*C++ function*), 8
libremidi::message::clear (*C++ function*), 8
libremidi::message::control_change (*C++ function*), 8
libremidi::message::end (*C++ function*), 8
libremidi::message::front (*C++ function*), 7
libremidi::message::get_channel (*C++ function*), 7
libremidi::message::get_message_type (*C++ function*), 7

libremidi::message::get_meta_event_type (C++ function), 7
libremidi::message::insert (C++ function), 8
libremidi::message::is_meta_event (C++ function), 7
libremidi::message::is_note_on_or_off (C++ function), 7
libremidi::message::make_command (C++ function), 8
libremidi::message::message (C++ function), 7
libremidi::message::note_off (C++ function), 8
libremidi::message::note_on (C++ function), 8
libremidi::message::operator[] (C++ function), 7, 8
libremidi::message::pitch_bend (C++ function), 8
libremidi::message::poly_pressure (C++ function), 8
libremidi::message::program_change (C++ function), 8
libremidi::message::rbegin (C++ function), 8
libremidi::message::rend (C++ function), 8
libremidi::message::size (C++ function), 7
libremidi::message::timestamp (C++ member), 8
libremidi::message::uses_channel (C++ function), 7
libremidi::message_type (C++ enum), 19
libremidi::message_type::ACTIVE_SENSING (C++ enumerator), 20
libremidi::message_type::AFTERTOUCH (C++ enumerator), 20
libremidi::message_type::CONTINUE (C++ enumerator), 20
libremidi::message_type::CONTROL_CHANGE (C++ enumerator), 19
libremidi::message_type::EOX (C++ enumerator), 20
libremidi::message_type::INVALID (C++ enumerator), 19
libremidi::message_type::NOTE_OFF (C++ enumerator), 19
libremidi::message_type::NOTE_ON (C++ enumerator), 19
libremidi::message_type::PITCH_BEND (C++ enumerator), 20
libremidi::message_type::POLY_PRESSURE (C++ enumerator), 19
libremidi::message_type::PROGRAM_CHANGE (C++ enumerator), 20
libremidi::message_type::RESERVED1 (C++ enumerator), 20
libremidi::message_type::RESERVED2 (C++ enumerator), 20
libremidi::message_type::RESERVED3 (C++ enumerator), 20
libremidi::message_type::RESERVED4 (C++ enumerator), 20
libremidi::message_type::SONG_POS_POINTER (C++ enumerator), 20
libremidi::message_type::SONG_SELECT (C++ enumerator), 20
libremidi::message_type::START (C++ enumerator), 20
libremidi::message_type::STOP (C++ enumerator), 20
libremidi::message_type::SYSTEM_EXCLUSIVE (C++ enumerator), 20
libremidi::message_type::SYSTEM_RESET (C++ enumerator), 20
libremidi::message_type::TIME_CLOCK (C++ enumerator), 20
libremidi::message_type::TIME_CODE (C++ enumerator), 20
libremidi::message_type::TUNE_REQUEST (C++ enumerator), 20
libremidi::meta_event_type (C++ enum), 20
libremidi::meta_event_type::CHANNEL_PREFIX (C++ enumerator), 21
libremidi::meta_event_type::COPYRIGHT (C++ enumerator), 20
libremidi::meta_event_type::CUE (C++ enumerator), 20
libremidi::meta_event_type::DEVICE_NAME (C++ enumerator), 20
libremidi::meta_event_type::END_OF_TRACK (C++ enumerator), 21
libremidi::meta_event_type::INSTRUMENT (C++ enumerator), 20
libremidi::meta_event_type::KEY_SIGNATURE (C++ enumerator), 21
libremidi::meta_event_type::LYRIC (C++ enumerator), 20
libremidi::meta_event_type::MARKER (C++ enumerator), 20
libremidi::meta_event_type::MIDI_PORT (C++ enumerator), 21
libremidi::meta_event_type::PATCH_NAME (C++ enumerator), 20
libremidi::meta_event_type::PROPRIETARY (C++ enumerator), 21
libremidi::meta_event_type::SEQUENCE_NUMBER (C++ enumerator), 20
libremidi::meta_event_type::SMPTE_OFFSET (C++ enumerator), 21

libremidi::meta_event_type::TEMPO_CHANGE (C++ enumerator), 21
libremidi::meta_event_type::TEXT (C++ enumerator), 20
libremidi::meta_event_type::TIME_SIGNATURE (C++ enumerator), 21
libremidi::meta_event_type::TRACK_NAME (C++ enumerator), 20
libremidi::meta_event_type::UNKNOWN (C++ enumerator), 21
libremidi::meta_events (C++ struct), 9
libremidi::meta_events::channel (C++ function), 9
libremidi::meta_events::end_of_track (C++ function), 9
libremidi::meta_events::key_signature (C++ function), 9
libremidi::meta_events::song_position (C++ function), 9
libremidi::meta_events::tempo (C++ function), 9
libremidi::meta_events::time_signature (C++ function), 9
libremidi::midi_bytes (C++ type), 28
libremidi::midi_error (C++ enum), 21
libremidi::midi_error::DRIVER_ERROR (C++ enumerator), 21
libremidi::midi_error::INVALID_DEVICE (C++ enumerator), 21
libremidi::midi_error::INVALID_PARAMETER (C++ enumerator), 21
libremidi::midi_error::INVALID_USE (C++ enumerator), 21
libremidi::midi_error::MEMORY_ERROR (C++ enumerator), 21
libremidi::midi_error::NO_DEVICES_FOUND (C++ enumerator), 21
libremidi::midi_error::SYSTEM_ERROR (C++ enumerator), 21
libremidi::midi_error::THREAD_ERROR (C++ enumerator), 21
libremidi::midi_error::UNSPECIFIED (C++ enumerator), 21
libremidi::midi_error::WARNING (C++ enumerator), 21
libremidi::midi_error_callback (C++ type), 28
libremidi::midi_exception (C++ struct), 10
libremidi::midi_exception::~midi_exception (C++ function), 10
libremidi::midi_in (C++ class), 13
libremidi::midi_in::~midi_in (C++ function), 13
libremidi::midi_in::cancel_callback (C++ function), 14
libremidi::midi_in::close_port (C++ function), 14
libremidi::midi_in::get_current_api (C++ function), 13
libremidi::midi_in::get_message (C++ function), 15
libremidi::midi_in::get_port_count (C++ function), 14
libremidi::midi_in::get_port_name (C++ function), 14
libremidi::midi_in::ignore_types (C++ function), 15
libremidi::midi_in::is_port_open (C++ function), 14
libremidi::midi_in::message_callback (C++ type), 13
libremidi::midi_in::midi_in (C++ function), 13
libremidi::midi_in::open_port (C++ function), 13
libremidi::midi_in::open_virtual_port (C++ function), 14
libremidi::midi_in::set_callback (C++ function), 14
libremidi::midi_in::set_client_name (C++ function), 15
libremidi::midi_in::set_error_callback (C++ function), 15
libremidi::midi_in::set_port_name (C++ function), 15
libremidi::midi_out (C++ class), 15
libremidi::midi_out::~midi_out (C++ function), 15
libremidi::midi_out::close_port (C++ function), 16
libremidi::midi_out::get_current_api (C++ function), 15
libremidi::midi_out::get_port_count (C++ function), 16
libremidi::midi_out::get_port_name (C++ function), 16
libremidi::midi_out::is_port_open (C++ function), 16
libremidi::midi_out::midi_out (C++ function), 15
libremidi::midi_out::open_port (C++ function), 16
libremidi::midi_out::open_virtual_port (C++ function), 16
libremidi::midi_out::send_message (C++ function), 16
libremidi::midi_out::set_chunking_parameters (C++ function), 17

libremidi::midi_out::set_client_name (C++ function), 17
libremidi::midi_out::set_error_callback (C++ function), 17
libremidi::midi_out::set_port_name (C++ function), 17
libremidi::midi_track (C++ type), 29
libremidi::no_devices_found_error (C++ struct), 10
libremidi::no_devices_found_error::~no_devices_found_error (C++ function), 10
libremidi::no_devices_found_error::code (C++ member), 10
libremidi::observer (C++ class), 17
libremidi::observer::~observer (C++ function), 17
libremidi::observer::callbacks (C++ struct), 11, 17
libremidi::observer::callbacks::input_added (C++ member), 11, 17
libremidi::observer::callbacks::input_removed (C++ member), 11, 17
libremidi::observer::callbacks::output_added (C++ member), 11, 17
libremidi::observer::callbacks::output_removed (C++ member), 11, 17
libremidi::observer::observer (C++ function), 17
libremidi::open_midi_in (C++ function), 23
libremidi::open_midi_observer (C++ function), 23
libremidi::open_midi_out (C++ function), 24
libremidi::parseEvent (C++ function), 24
libremidi::reader (C++ class), 18
libremidi::reader::~reader (C++ function), 18
libremidi::reader::get_end_time (C++ function), 18
libremidi::reader::parse (C++ function), 18
libremidi::reader::reader (C++ function), 18
libremidi::reader::startingTempo (C++ member), 18
libremidi::reader::ticksPerBeat (C++ member), 18
libremidi::reader::tracks (C++ member), 18
libremidi::system_error (C++ struct), 11
libremidi::system_error::~system_error (C++ function), 11
libremidi::system_error::code (C++ member), 11
libremidi::thread_error (C++ struct), 12
libremidi::thread_error::~thread_error (C++ function), 12
libremidi::thread_error::code (C++ member), 12
libremidi::track_event (C++ struct), 12
libremidi::track_event::m (C++ member), 12
libremidi::track_event::tick (C++ member), 12
libremidi::track_event::track (C++ member), 12
libremidi::util::read_bytes (C++ function), 24
libremidi::util::read_uint16_be (C++ function), 24
libremidi::util::read_uint24_be (C++ function), 25
libremidi::util::read_uint32_be (C++ function), 25
libremidi::util::read_variable_length (C++ function), 25
libremidi::util::write_double_be (C++ function), 25
libremidi::util::write_float_be (C++ function), 25
libremidi::util::write_int16_be (C++ function), 26
libremidi::util::write_int32_be (C++ function), 26
libremidi::util::write_uint16_be (C++ function), 26
libremidi::util::write_uint32_be (C++ function), 26
libremidi::util::write_variable_length (C++ function), 26
libremidi::writer (C++ class), 18
libremidi::writer::~writer (C++ function), 18
libremidi::writer::add_event (C++ function), 18
libremidi::writer::add_track (C++ function), 18
libremidi::writer::get_num_tracks (C++ function), 18
libremidi::writer::get_tracks (C++ function), 18
libremidi::writer::write (C++ function), 18
libremidi::writer::writer (C++ function), 18
LIBREMIDI_EXPORT (C macro), 27
LIBREMIDI_INLINE (C macro), 27
LIBREMIDI_VERSION (C macro), 28