Package: openmpt (via r-universe)

January 26, 2025

Title Open 'ModPlug' Tracker Port

Version 0.1.4

Description Tracker music uses audio samples and pattern tables to organise musical compositions. Such music is stored in module files. This package reads, renders and plays module files using the 'libopenmpt' library <https://lib.openmpt.org/>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Imports av

Suggests httr2, kableExtra, knitr, rmarkdown, testthat (>= 3.0.0), xml2

LinkingTo cpp11

SystemRequirements openmpt: libopenmpt-devel (rpm) or libopenmpt-dev (deb), portaudio-devel (rpm) or portaudio19-dev (deb).

URL https://pepijn-devries.github.io/openmpt/, https://github.com/pepijn-devries/openmpt, https://lib.openmpt.org/

BugReports https://github.com/pepijn-devries/openmpt/issues

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

Author Pepijn de Vries [aut, cre] (<https://orcid.org/0000-0002-7961-6646>), Jeroen Ooms [ctb] (<https://orcid.org/0000-0002-4035-0289>), Jester [cph, dtc] (Copyright holder of cyberrid.mod), OpenMPT Project Developers and Contributors [cph]

Maintainer Pepijn de Vries <pepijn.devries@outlook.com>

Date/Publication 2025-01-11 16:00:02 UTC

Additional_repositories https://cranhaven.r-universe.dev

Config/pak/sysreqs libavfilter-dev libopenmpt-dev portaudio19-dev

Repository https://cranhaven.r-universe.dev

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/openmpt

RemoteSha 4a6773251aed2d9e482eae7bb6c21470cd475b29

RemoteSubdir openmpt

Contents

`control_keys` *Get or set OpenMPT module controls*

Description

Each individual module has its own set of control parameters. Use these functions to obtain or set the state of these parameters.

Usage

```
control_keys(mod, ...)

control(mod, key, ...)

control(mod, key, ...) <- value
```

Arguments

mod	A tracker module object of class openmpt.
...	Ignored
key	A character string representing a specific control you whish to get or set. Use control_keys() to list all available keys.
value	A replacement value for the specified control key. Check the libopenmpt documentation for the appropriate replacement types and values for each of the key values.

Value

control_keys() returns a vector of strings containing all available control keys for mod. control() returns the value for the specified key'. In case of an assign operator (<-) an updated version of mod‘ is returned, where the control key has been set if successful.

Examples

```
mod <- demo_mod()
control_keys(mod)

## get a specific control value
control(mod, "play.at_end")

## set a number of control values
control(mod, "play.at_end") <- "stop"
control(mod, "play.pitch_factor") <- 2
control(mod, "load.skip_plugins") <- TRUE
control(mod, "dither") <- 2L

## Show all control settings
all_keys <- control_keys(mod)
structure(
  lapply(all_keys, control, mod = mod),
  names = all_keys
)
```

convert_mod *Convert a ModPlug Tracker module to an audio file*

Description

Renders ModPlug Tracker music to an audio file and encodes it to a desired output format (e.g. .mp3, .ogg, etc) using `av::av_audio_convert()`.

Usage

```
convert_mod(
  mod,
  file,
  start_order = 0L,
  start_row = 0L,
  sample_rate = 44100L,
  duration = NA_real_,
  verbose = FALSE,
  ...
)
```

Arguments

`mod`	A tracker module object of class `openmpt`
`file`	Output audio file where the rendered audio is stored. The file name extension is used to determine the type of encoding to be applied.
`start_order`	Starting position (`integer` index starting at 0) in the pattern sequence table.
`start_row`	Starting row (`integer` index starting at 0) of the pattern table.
`sample_rate`	Output sample rate in Hz (samples per seconds).
`duration`	Duration in seconds. Rendered sample will not be longer than this duration. if set to `NA_real_` it is ignored and the module keeps rendering conform the specified `control()`.
`verbose`	Passed on to `av::av_audio_convert()`.
`...`	Ignored

Value

Returns `NULL` invisibly

Examples

```
mod <- demo_mod()

destination <- tempfile(fileext = ".mp3")

convert_mod(mod, destination, duration = 2)
```

demo_mod *Loads demo module included with the package*

Description

Reads the file `cyberrid.mod`. It is a ProTracker file create by Jester. It is redistributed under the Attribution Non-commercial Share Alike license. The music was part of an Amiga computer demo named 'Extension' and was originally released in 1993.

Usage

```
demo_mod()
```

Value

Returns the demo module tracker object of class `openmpt`

Examples

```
mod <- demo_mod()
```

get_current_channel_vu_left *Get the state of specific aspects of an* `openmpt` *class object*

Description

While playing with `play()` or rendering with `convert_mod()`, the state of the module can change continuously (volume, speed, order index, etc.). These functions return the current state of an `openmpt` class object.

Usage

```
get_current_channel_vu_left(mod, channel, ...)

get_current_channel_vu_mono(mod, channel, ...)

get_current_channel_vu_rear_left(mod, channel, ...)

get_current_channel_vu_rear_right(mod, channel, ...)

get_current_channel_vu_right(mod, channel, ...)

get_current_estimated_bpm(mod, ...)

get_current_order(mod, ...)
```

```
get_current_pattern(mod, ...)

get_current_playing_channels(mod, ...)

get_current_row(mod, ...)

get_current_speed(mod, ...)

get_current_tempo(mod, ...)
```

Arguments

mod	A tracker module object of class openmpt.
channel	An integer channel index (starting at 0), for which to get the current state.
...	Ignored

Value

Return numeric or integer values of the requested state. Function names are pretty self-explanatory. Note that tempo and speed values are tracker dependent, their meaning depend on the originating tracker.

Examples

```
mod <- demo_mod()

get_current_channel_vu_left(mod, 0L)
get_current_channel_vu_mono(mod, 0L)
get_current_channel_vu_rear_left(mod, 0L)
get_current_channel_vu_rear_right(mod, 0L)
get_current_channel_vu_right(mod, 0L)
get_current_estimated_bpm(mod)
get_current_order(mod)
get_current_pattern(mod)
get_current_playing_channels(mod)
get_current_row(mod)
get_current_speed(mod)
get_current_tempo(mod)
```

---

get_duration_seconds *Get ModPlug Tracker module duration*

Description

Get the duration of the song from a openmpt class module object in seconds.

Usage

```
get_duration_seconds(mod, ...)
```

Arguments

mod	A tracker module object of class openmpt.
...	Ignored

Value

Returns a numeric value indicating the song duration in seconds.

Examples

```
mod <- demo_mod()
get_duration_seconds(mod)
```

---

get_instrument_names *Get* openmpt *module element names*

Description

Get names of elements in an openmpt class object. Use get_metadata() to get a module's name.

Usage

```
get_instrument_names(mod, ...)

get_sample_names(mod, ...)

get_channel_names(mod, ...)

get_pattern_names(mod, ...)

get_order_names(mod, ...)

get_subsong_names(mod, ...)
```

Arguments

mod	A tracker module object of class openmpt.
...	Ignored

Value

A vector of strings with names

Examples

```
mod <- demo_mod()
get_subsong_names(mod)
get_channel_names(mod)
get_pattern_names(mod)
get_order_names(mod)
get_instrument_names(mod)
get_sample_names(mod)[1:8]
```

`get_metadata` *Get ModPlug Tracker module meta data*

Description

Get meta data of a tracker module such as its `"type"`, `"title"` and `"tracker"`. Use `get_metadata_keys()` to get the available keys for a module object.

Usage

```
get_metadata(mod, key = "title", ...)

get_metadata_keys(mod, ...)
```

Arguments

`mod`	A tracker module object of class `openmpt`.
`key`	A key as listed by `get_metadata_keys()`.
`...`	Ignored

Value

A list of available keys in case of `get_metadata_keys()`, the requested information in case of `get_metadata()`.

Examples

```
mod <- demo_mod()
get_metadata_keys(mod)

get_metadata(mod, "tracker")
```

get_num_instruments *Get element counts from an* openmpt *module*

Description

Functions that count specific elements in openmpt class objects and returns the resulting number.

Usage

```
get_num_instruments(mod, ...)

get_num_samples(mod, ...)

get_num_channels(mod, ...)

get_num_orders(mod, ...)

get_num_patterns(mod, ...)

get_num_subsongs(mod, ...)

get_pattern_num_rows(mod, pattern)
```

Arguments

mod	A tracker module object of class openmpt.
...	Ignored
pattern	An integer pattern index (starting at 0) for which to get count details.

Value

Returns an integer count of the requested information.

Examples

```
mod <- demo_mod()
get_num_instruments(mod)
get_num_samples(mod)
get_num_channels(mod)
get_num_orders(mod)
get_num_patterns(mod)
get_num_subsongs(mod)
get_pattern_num_rows(mod, 0L)
```

get_order_pattern *Get the pattern index of an* openmpt *module at a specific order index*

Description

A module contains a sequence table describing the order in which to play patterns. This function returns the index of the patter at specific position in the sequence table.

Usage

```
get_order_pattern(mod, order, ...)
```

Arguments

mod	A tracker module object of class openmpt.
order	Index of the position in the pattern sequence table (starts at 0).
...	Ignored

Value

Returns the integer index (starting at 0) of the pattern at the indicated order position.

Examples

```
mod <- demo_mod()
get_order_pattern(mod, 3L)
```

has_audio_device *Test if there is an audio device*

Description

Tests if an audio device is present on the system.

Usage

```
has_audio_device()
```

Value

Returns a logical value.

modarchive_info *Functions to interact with modArchive*

Description

ModArchive is one of the largest online archives of module files. These functions will assist in accessing this archive. For mor information see `vignette("modarchive")`.

Usage

```
modarchive_info(mod_id, api = modarchive_api())

modarchive_search_mod(
  text,
 where = c("filename_or_songtitle", "filename_and_songtitle", "filename", "songtitle",
    "module_instruments", "module_comments"),
 format = c("unset", "669", "AHX", "DMF", "HVL", "IT", "MED", "MO3", "MOD", "MTM",
    "OCT", "OKT", "S3M", "STM", "XM"),
  size,
  channels,
  page,
  api = modarchive_api()
)

modarchive_search_genre(
  genre = c("unset", modarchive_genres()),
 format = c("unset", "669", "AHX", "DMF", "HVL", "IT", "MED", "MO3", "MOD", "MTM",
    "OCT", "OKT", "S3M", "STM", "XM"),
  size,
  channels,
  page,
  api = modarchive_api()
)

modarchive_search_hash(text, api = modarchive_api())

modarchive_random(
  genre = modarchive_genres(),
 format = c("unset", "669", "AHX", "DMF", "HVL", "IT", "MED", "MO3", "MOD", "MTM",
    "OCT", "OKT", "S3M", "STM", "XM"),
  size,
  page,
  api = modarchive_api()
)

modarchive_search_artist(text, page, api = modarchive_api())
```

```
modarchive_view_by(
  text,
  by = c("view_by_list", "view_by_rating_comments", "view_by_rating_reviews",
    "view_modules_by_artistid", "view_modules_by_guessed_artist"),
 format = c("unset", "669", "AHX", "DMF", "HVL", "IT", "MED", "MO3", "MOD", "MTM",
    "OCT", "OKT", "S3M", "STM", "XM"),
  size,
  page,
  api = modarchive_api()
)

modarchive_download(mod_id, read_fun = read_mod, ...)

modarchive_api()

modarchive_requests(api = modarchive_api())

modarchive_genres()
```

Arguments

mod_id	An integer code used as module identifier in the ModArchive database. A mod_id can be obtained by performing a search with for instance modarchive_search_mod().
api	Most ModArchive functions require a personal secret API key. This key can be obtained from the ModArchive forum. See vignette("modarchive") for more details.
text	Text (character) used for searching. In some functions the asterisk symbol * can be used as a wildcard in the search.
where	A character string specifying where to search. See the 'usage' section for allowed values.
format	A character string specifying to which file format the search should be limited. See 'usage' section for allowed values.
size	A vector of two integer values, specifying a filter to apply to the search results. It filters the results to the file size range specified here in kB. When omitted, all file sizes are returned.
channels	A vector of two integer values, specifying a filter to apply to the search results. It filters the results to the specified range of number of channels in the module. When omitted, modules with any number of channels are returned.
page	Many of the ModArchive functions return paginated tables. When this argument is omitted, the first page is returned. Use an integer value to return a specific page. The total number of pages of a search or view is returned as an attribute to the returned data.frame.
genre	A genre of music to limit your search to. See modarchive_genres() for a list of available values.
by	A character string specifying which aspect to view the results by. See the 'usage' section for allowed values.

read_fun	Function that accepts an URL first argument. By default it is read_mod() and is used to read the file. You can replace it with other functions such as ProTrackR2::pt2_read_mod().
...	Arguments passed on to read_fun

Value

Most functions documented here return a data.frame with information about one or more modules, or an artist. NULL is returned in case a search has no results.

modarchive_download() returns the result of calling read_fun on the requested module.

modarchive_requests() returns the number of requests that you made this month using the API key, and how many are available.

modarchive_api() returns your API key, when you have set it as environmental variable ("MODARCHIVE_API") or session option ("modarchive_api"). When it is not set it will return "".

modarchive_genres() returns a vector of character strings, listing the music genres specified by ModArchive.

See Also

modland_search()

Examples

```
elekfunk <- modarchive_download(41529)

## Check how many API requests are left this month
reqs <- modarchive_requests()
if (length(reqs) > 0) {
  reqs <- 1 - reqs$current / reqs$maximum
} else {
  reqs <- 0
}

## The examples below will only work with a valid
## API key for modArchive and if more than 25%
## of the monthly requests are left:
if (modarchive_api() != "" && reqs > 0.25) {

  mod_info <- modarchive_info(41529)
  if (nrow(mod_info) > 0) mod_info$url[[1]]
  info_search <- modarchive_search_mod("*intro.mod",
                                       size = c(8L, 10L),
                                       channels = c(1L, 4L))
  info_genre  <- modarchive_search_genre("Chiptune", "IT")
  info_hash   <- modarchive_search_hash("8f80bcab909f700619025bd7f2975749")
  info_artist <- modarchive_search_artist("89200")
  info_list   <- modarchive_view_by("A", "view_by_list", "XM",
                                    page = 2)
  info_random <- modarchive_random("Comedy")
}
```

`modland_search` *Functions to interact with modLand*

Description

ModLand is an online archive containing over 400,000 module files. These functions allow you to search in and download from this archive.

Usage

```
modland_search(text, ...)

modland_download(
  format,
  author,
  title,
  mirror = c("modland.com", "ftp.modland.com", "antarctica.no", "ziphoid.com",
    "exotica.org.uk"),
  read_fun = read_mod,
  ...
)
```

Arguments

`text`	Search text, to look for on modland.
`...`	Arguments passed on to `read_fun`
`format`	A single length `character` vector, indicating the tracker file format. Can be obtained from a `modland_search()`.
`author`	A single length `character` vector, indicating the module author name. Can be obtained from a `modland_search()`.
`title`	A single length `character` vector, indicating the module title. Can be obtained from a `modland_search()`.
`mirror`	A single length `character` vector. Should contain one of the mirrors listed in the 'usage' section. Select a mirror site from which the module file needs to be downloaded.
`read_fun`	Function that accepts an URL first argument. By default it is `read_mod()` and is used to read the file. You can replace it with other functions such as `ProTrackR2::pt2_read_mod()`.

Value

In case of `modland_search()` a `data.frame` with search results are returned (or `NULL` if there are no results).

`modland_download()` will return the result of the function specified by `read_fun`. By default it will return an `openmpt` class object.

See Also

modarchive_search_mod()

Examples

```
search_result <- modland_search("elekfunk mod")

## The URL in the search results will download a rendered
## ogg file. If you want to download te original mod file,
## use this:
if (length(search_result) > 0) {
  mod <- modland_download(search_result$format[[1]],
                          search_result$author[[1]],
                          search_result$title[[1]])
}
```

---

openmpt_info *Retrieve information about the OpenMPT library*

Description

A wrapper for the get function in libopenmpt (see API documentation)

Usage

```
openmpt_info(key = "library_version", ...)
```

Arguments

key	A key character string indicating which information to retrieve. can be "library_version", "library_features" and many others. See API documentation) for all possible values.
...	Ignored

Value

Returns a character string with the requested information.

Examples

```
openmpt_info("library_version")
openmpt_info("library_features")
openmpt_info("url")
```

pattern *Get a specific* openmpt *pattern table or its cells*

Description

Collects a specific pattern table from a tracker module and presents it as a matrix of formatted character strings in case of pattern(). The other functions return a single string for a specific cell inside the pattern table.

Usage

```
pattern(mod, pattern = 0L, width = 0L, pad = TRUE, ...)

format_pattern_row_channel(mod, pattern, row, channel, width, pad, ...)

format_pattern_row_channel_command(mod, pattern, row, channel, command, ...)

highlight_pattern_row_channel(mod, pattern, row, channel, width, pad, ...)

highlight_pattern_row_channel_command(mod, pattern, row, channel, command, ...)
```

Arguments

mod	A tracker module object of class openmpt.
pattern	The pattern index (starting at 0) of the pattern to get.
width	The maximum number of characters the string should contain. 0 means no limit.
pad	If TRUE, the string will be resized to the exact length provided in the width parameter.
...	Ignored
row	A row index (starting at 0) for the row inside the pattern table.
channel	a channel (i.e., column) index (starting at 0) inside the pattern table.
command	One of "note", "instrument", "volumeffect", "effect", "volume", or "parameter".

Value

A matrix of pattern cells formatted as character strings in case of pattern(). Each column represents. All other methods return a single string for a specific cell. an audio channel.

Examples

```
mod <- demo_mod()
pattern(mod)
format_pattern_row_channel(mod, 0L, 1L, 2L, 0L, TRUE)
format_pattern_row_channel_command(mod, 0L, 1L, 2L, "parameter")
highlight_pattern_row_channel(mod, 0L, 1L, 2L, 0L, TRUE)
highlight_pattern_row_channel_command(mod, 0L, 1L, 2L, "note")
```

pitch-tempo *Control the pitch and tempo of a module*

Description

Functions to control the pitch and tempo of a module.

Usage

```
pitch_factor(mod, ...)

pitch_factor(mod, ...) <- value

tempo_factor(mod, ...)

tempo_factor(mod, ...) <- value
```

Arguments

mod	A tracker module object of class openmpt.
...	Ignored
value	Replacement value. A numeric factor with which to adjust the tempo or pitch of a module

Value

Returns current factor, or the updated object in case of an assign operation (<-).

Examples

```
mod <- demo_mod()

## Increase the module pitch with a factor 2
pitch_factor(mod) <- 2
pitch_factor(mod)

## Increase the module tempo with a factor 2
tempo_factor(mod) <- 2
tempo_factor(mod)
```

play *Play a ModPlug Tracker module*

Description

Renders a module tracker object of class openmpt and plays it instantaneously.

Usage

```
play(mod, sample_rate = 44100L, progress = "vu", duration = NA_real_, ...)
```

Arguments

mod	A tracker module object of class openmpt.
sample_rate	Output sample rate when playing the module.
progress	Progress printed to console while playing. Should be one of "vu" (indicative volume meter), "time" (shows timer) or "none" (don't show progress). If your audio is stuttering you might want to set this to "none" to save processing speed.
duration	Duration in seconds. Play routine will not last longer than this duration. if set to NA_real_ it is ignored and the module keeps rendering conform the specified control().
...	Ignored

Value

Returns NULL invisibly.

Examples

```
if (interactive() && has_audio_device()) {
  mod <- demo_mod()
  play(mod)
}
```

position_seconds *Get and set ModPlug Tracker module position*

Description

Get or set the position of the music player. rewind() moves the position to the start of the song.

Usage

```
position_seconds(mod, ...)

position_seconds(mod, ...) <- value

rewind(mod, ...)

set_position_order_row(mod, order, row, ...)
```

Arguments

mod	A tracker module object of class openmpt.
...	Ignored
value	Position in seconds to move the player to. The value is rounded to its nearest order and row position.
order	Index of the position in the pattern sequence table (starts at 0).
row	Index of the row in the current pattern table (starts at 0).

Value

Returns NULL invisibly, or the updated object in case of the assign operator (<-).

Examples

```
mod <- demo_mod()
position_seconds(mod)
position_seconds(mod) <- 10.2
set_position_order_row(mod, 1, 4)
rewind(mod)
```

---

print.openmpt *Implementation of basic S3 generics*

Description

Implementation of basic S3 generics such as print().

Usage

```
## S3 method for class 'openmpt'
print(x, ...)

## S3 method for class 'openmpt'
format(x, ...)
```

Arguments

x	Object to apply the method to.
...	Ignored.

Value

In case of `print` and `format` a formatted string with basic information about the module is returned.

---

read_mod *Read Open ModPlug module*

Description

Read any of the music tracker module file formats supported by libopenmpt: https://wiki.openmpt.org/Manual:_Module_formats.

Usage

```
read_mod(file, ...)
```

Arguments

file	File path or URL to read the file from. Binary connections are also supported.
...	Ignored

Value

A `modplug` class object. It is an external pointer, pointing to the module object in memory. It can be used for rendering audio.

Examples

```
## You can read from files
mod1 <- read_mod(system.file("cyberrid", "cyberrid.mod", package = "openmpt"))


## but also URLs
mod2 <- read_mod("https://api.modarchive.org/downloads.php?moduleid=41529#elektric_funk.mod")
```

render_param *Get or set render parameters for a specific module*

Description

Each individual module has its own set of render parameters. Use these functions to obtain or set the state of these parameters.

Usage

```
render_param(mod, key, ...)

render_param(mod, key, ...) <- value
```

Arguments

mod	A tracker module object of class openmpt.
key	One of "MASTERGAIN", "STEREOSEPARATION", "INTERPOLATION", or "VOLUMERAMPING". details copied from libopenmpt documentation

Master Gain The related value represents a relative gain in milliBel. The default value is 0. The supported value range is unlimited.

Stereo Separation The related value represents the stereo separation generated by the libopenmpt mixer in percent. The default value is 100. The supported value range is from 0 up to 200.

Interpolation Filter The related value represents the interpolation filter length used by the libopenmpt mixer. The default value is 0, which indicates a recommended default value. The supported value range is from 0 up to infinity. Values greater than the implementation limit are clamped to the maximum supported value. Currently supported values:

- 0: internal default
- 1: no interpolation (zero order hold)
- 2: linear interpolation
- 4: cubic interpolation
- 8: windowed sinc with 8 taps

Volume Ramping Strength The related value represents the amount of volume ramping done by the libopenmpt mixer. The default value is -1, which indicates a recommended default value. The meaningful value range is from -1 up to 10. A value of 0 completely disables volume ramping. This might cause clicks in sound output. Higher values imply slower/softer volume ramps.

...	Ignored.
value	An integer replacement value for the render parameter selected with key

Value

Returns the current integer render parameter for the specified `key` and `mod`. In case of an assign operator (`<-`) `mod` with an updated set of render parameters is returned.

Examples

```
mod <- demo_mod()

render_param(mod, "STEREOSEPARATION") <- 50
render_param(mod, "STEREOSEPARATION")
render_param(mod, "MASTERGAIN")
render_param(mod, "INTERPOLATION")
render_param(mod, "VOLUMERAMPING")
```

---

repeat_count *Get or set the repeat count for an* `openmpt` *module*

Description

Tracker music can be composed such that it is intended to play in a continuous loop. With the repeat count you can affect the number of times a module is repeated when playing with `play()` or rendered with `convert_mod()`.

Usage

```
repeat_count(mod, ...)

repeat_count(mod, ...) <- value
```

Arguments

`mod`	A tracker module object of class `openmpt`.
`...`	Ignored
`value`	An `integer` value to assign to the repeat count.

Value

Returns the `integer` repeat count of an `openmpt` object. In case of an assign operator (`<-`) an updated `mod` is returned with the new repeat count.

Examples

```
mod <- demo_mod()
repeat_count(mod) <- 2
repeat_count(mod)
```

subsong *Get or set the current subsong in an* openmpt *module*

Description

Some openmpt modules may contain multiple subsongs. Use these functions to get the current subsong index, or select a different one.

Usage

```
subsong(mod, ...)

subsong(mod, ...) <- value
```

Arguments

mod	A tracker module object of class openmpt.
...	Ignored
value	An integer index of the subsong to select.

Value

Returns the integer index of the currently selected subsong. In case of the assign operator (<-) it returns a version of mod with an update selection for the subsong

Examples

```
mod <- demo_mod()
subsong(mod)
## a value of -1 plays all subsongs consecutively
subsong(mod) <- -1
```

volume-control *Control the volume of a module*

Description

Functions to control the global volume of a module, or that of specific channels in the module.

Usage

```
channel_mute_status(mod, channel, ...)

channel_mute_status(mod, channel, ...) <- value

channel_volume(mod, channel, ...)

channel_volume(mod, channel, ...) <- value

global_volume(mod, ...)

global_volume(mod, ...) <- value
```

Arguments

mod	A tracker module object of class openmpt.
channel	Channel index (integer starting at 0) for which to request or control the volume.
...	Ignored
value	Replacement value. In case of 'status' functions a logical value, in case of 'volume' functions a numeric value.

Value

Returns the volume (status), or the updated object in case of an assign operation (<-).

Examples

```
mod <- demo_mod()

channel_mute_status(mod, 0L)

## Mute the first channel in the module
channel_mute_status(mod, 0L) <- TRUE

## Second channel volume at 50%
channel_volume(mod, 1L) <- 0.5
channel_volume(mod, 1L)

## global volume at 90%
global_volume(mod) <- 0.9
global_volume(mod)
```

Index