MIDI Functions

The Smart Multibox has quite a few functions that allow you to create, send and receive MIDI messages. MIDI messages can also be created manually - they are actually Python bytearrays, and can be manipulated with standard Python functions.

Message Creation Functions

midi_msg_make_1byte(msg_type)

Creates and returns a 1 byte MIDI message with the given message type. Message types can be found in the Constants section.

midi_msg_make_cc(channel, number, value)

Creates and returns a MIDI CC (continuous controller) message with the given MIDI channel (1-16), number (0-127) and value (0-127).

midi_msg_make_chan_pressure(channel, value)

Creates and returns a MIDI channel pressure message with the given MIDI channel (1-16) and value (0-127).

midi_msg_make_note(channel, note, velocity)

Creates and returns a MIDI note message with the given MIDI channel (1-16), note number (0-127) and velocity (0-127). Using velocity of 0 is equivalent to a "note off".

midi_msg_make_pc(channel, number)

Creates and returns a MIDI PC (program change) message with the given MIDI channel (1-16) and number (0-127).

midi_msg_make_pitch_bend(channel, value)

Creates and returns a MIDI pitch bend message with the given MIDI channel (1-16) and value (0-16383).

midi_msg_make_poly_pressure(channel, note, value)

Creates and returns a MIDI polyphonic pressure message with the given MIDI channel (1-16), note number (0-127) and value (0-127).

Message Manipulation Functions

midi_msg_get_channel(midi_msg)

Given a MIDI message, returns the MIDI channel the message is on (1-16). Returns -1 if the MIDI message has no channel associated with it.

midi_msg_set_channel(midi_msg, channel)

Given a MIDI message, sets the MIDI channel of the message (1-16). Raises a ValueError exception if the channel value is not valid, or if the message does not have a specific channel.

midi_msg_get_number(midi_msg)

Given a MIDI message, returns the PC, CC or note number from the message is (0-127). Returns -1 if the MIDI message has no number associated with it.

midi_msg_set_number(midi_msg, number)

Given a MIDI message, sets the PC, CC or Note number of the message (0-127). Raises a ValueError exception if the channel value is not valid, or if the message does not have a number associated with it.

midi_msg_get_type(midi_msg)

Given a MIDI message, returns the MIDI message type, which is the status byte with the MIDI channel value zeroed out (128-250). Constants defined for each message type are defined in the Constants section. Returns -1 if the MIDI message is not valid.

midi_msg_set_type(midi_msg, msg_type)

Given a MIDI message, sets the type of the message (128-250). Constants defined for each message type are defined in the Constants section. The existing MIDI channel of the message, if any, is left unchanged. Raises a ValueError exception if the type value or the MIDI message is not valid.

midi_msg_get_value(midi_msg)

Given a MIDI message, returns the CC, velocity or pressure value of the message (0-127). Returns -1 if the MIDI message has no value associated with it.

midi_msg_set_value(midi_msg, channel)

Given a MIDI message, sets the CC, velocity or pressure value of the message (0-127). Raises a ValueError exception if the value is not valid, or if the message does not have a value associated with it.

Other MIDI Related Functions

midi_allow_running_status(in_port, allow)

If allow is set to True, data bytes received on in_port that have no status byte in front of them will have the most recently received status byte prepended to them, per the Running Status feature found in the MIDI specification. If allow is set to False, any data bytes received without a preceding status byte will be ignored. The default setting is False.

midi_msg_is_valid(msg)

Returns True if the given value is a valid MIDI message, False if not.

midi_route_clear()

Removes all MIDI clock routes

midi_route_clock(in_port, out_port)

Adds a routing connection from in_port to out_port, where MIDI clock, start, stop and continue messages received at in_port are automatically forwarded to out_port. See the Constants section for port definitions.

midi_send(port, msg)

Sends a MIDI message to a specified MIDI port. The MIDI port values are defined in the Constants section. Returns the number of bytes sent, or -1 if an error occurred.

usb_route_input(port, enable)

Enables or disables routing from a MIDI input to USB. This is the same type of routing used in the USB MIDI mode of the Smart Multibox. For example, messages coming in to MIDI In 1 will appear on a connected computer on Smart Multibox MIDI In 1. By default, all MIDI to USB routes are disabled.

usb_route_output(port, enable)

Enables or disables routing from USB to a MIDI output. This is the same type of routing used in the USB MIDI mode of the Smart Multibox. For example, sending a message to Smart Multibox MIDI Out 1 on a connected computer will result in that message coming out of MIDI Out 1 on the Smart Multibox. By default, all USB to MIDI routes are disabled.

Revision #15
Created 12 January 2025 22:27:37 by RJM Music
Updated 7 March 2025 16:53:16 by RJM Music