Consecutive MIDI messages sent to same channel are garbled

Asked 2 years, 2 months ago Active 1 year ago Viewed 115 times

0

I'm writing a simple C++ command line utility to send MIDI messages from my Mac (MacOS 10.12) to a Behringer Powerplay P16-M audio mixer via a Roland UM-One Mk 2 USB interface. I'm using the `rtmidi` realtime MIDI API.

When I send two consecutive MIDI messages to the same channel, the two messages seem to get garbled by the mixer. When I send two consecutive messages to alternate channels, it works fine.

For example, in order to set channel 1 level to midi 0 (-51dB) and channel 1 pan to midi 64 (centered), I understand that I should send these two messages: `0xB0 0x07 0x0` and `0xB0 0x0A 0x40` . When I send those messages one after the other, the LEDs on the mixer unexpectedly indicate that channel 1 level is about 64 and the panning is unchanged -- just as if I'd sent the single message `0xB0 0x07 0x40` . If instead I intersperse the two messages with a message for another channel, or with a ridiculously long sleep (anything over about 900 milliseconds does it), the LEDs indicate the expected settings: level at zero, pan centered.

What am I doing wrong? Why can't I send two consecutive messages to the same channel? Am I sending the messages too fast? Is this is an `rtmidi` issue? Is it an issue with the mixer?

Here's a minimal but complete working example that demonstrates the problem. (I compile it in XCode against the CoreMIDI, CoreAudio, and CoreFoundation frameworks.)

```
#include <iostream>
#include <vector>
#include <unistd.h> // (for usleep)
#include "RtMidi.h"

// define some midi messages
static std::vector<unsigned char> ch1vol {0xB0 , 0x07 , 0x0}; //  set ch 1 volume to
midi 0 (-51dB)
static std::vector<unsigned char> ch1pan {0xB0 , 0x0A , 0x40}; // set ch 1 pan to 64
static std::vector<unsigned char> ch2vol {0xB1 , 0x07 , 0x0}; // set ch 2 volume to midi
0 (-51dB)
static std::vector<unsigned char> ch2pan {0xB1 , 0x0A , 0x40}; // set ch 2 pan to 64

#define SLEEPMSEC( milliseconds ) usleep( (unsigned long) (milliseconds * 1000.0) )

int main(int argc, const char * argv[]) {

    RtMidiOut * midiOut = new RtMidiOut(); // create an RtMidiOut

    // are there any midi ports available?
    if (!midiOut->getPortCount()) {
        std::cout << "*** No midi ports found. Goodbye." << std::endl;
        exit(1);
    }

    // open the desired midi device
    std::string portName = midiOut->getPortName(0);
    if (portName != "Powerplay 16") {
        std::cout << "*** Can't find requested midi device. Goodbye." << std::endl;
        exit(1);
    }
    midiOut->openPort(0); // open the midi device to receive output

    switch(argc) {
        case 2:
            // This works as expected
            // Result: channels 1 and 2 volumes are set to 0; the panning in both
channels is centered)
            midiOut->sendMessage( &ch1vol ); // set ch 1 volume to 0
            midiOut->sendMessage( &ch2vol ); // set ch 2 volume to 0
            midiOut->sendMessage( &ch1pan ); // set ch 1 pan to center
            midiOut->sendMessage( &ch2pan ); // set ch 2 pan to center
            break;
        case 3:
            // This does NOT work as expected
            // Result: channels 1 and 2 volumes are set to 64; the panning in both
channels is  unchanged
            midiOut->sendMessage( &ch1vol ); // set ch 1 volume to 0
            midiOut->sendMessage( &ch1pan ); // set ch 1 pan to center
            midiOut->sendMessage( &ch2vol ); // set ch 2 volume to 0
            midiOut->sendMessage( &ch2pan ); // set ch 2 pan to center
            break;
        case 4:
            // Introduce a sleep in between same-channel calls.
            // This DOES work as expected (but with an unacceptable delay)
            midiOut->sendMessage( &ch1vol ); // set ch 1 volume to 0
            SLEEPMSEC(900);
            midiOut->sendMessage( &ch1pan ); // set ch 1 pan to center
            midiOut->sendMessage( &ch2vol ); // set ch 2 volume to 0
            SLEEPMSEC(900);
            midiOut->sendMessage( &ch2pan ); // set ch 2 pan to center
            break;
        default:
            std::cout << "Usage: testmidi ARG [ARG [ARG]]" << std::endl;
            std::cout << "Perform a midi test according to the number of arguments." <<
std::endl;
            std::cout << "Be sure to reset the device manually after each run." <<
std::endl;
            break;
    }
}
```

c++ midi coremidi mixer

asked Mar 18 '18 at 16:20

jt bullitt
240 3 11

Ahh MIDI. Thanks for the flashbacks :D – Lightness Races in Orbit Mar 18 '18 at 16:45

2 I'm not familiar with rtmidi, but your code looks reasonable. I think it's most likely to be a bug in the mixer, but theoretically it could be anything in the chain from rtmidi to the mixer (that is, CoreMIDI, the UM-ONE's driver, the Mac USB stack, the UM-ONE itself, then the MIDI cable). Try connecting the MIDI OUT of the UM-ONE to another computer's MIDI IN (or make a loopback cable to connect to the MIDI IN of the same UM-ONE) and see if the MIDI on the physical cable is what you expect. If it's good, it's the mixer's fault. – Kurt Revis Mar 18 '18 at 21:36

This sounds as if some component does not handle running status correctly. To rule out your software and rtmidi, can you put just these two messages into a .mid file and play that with some normal MIDI player? – CL. Mar 18 '18 at 22:18

1 Answer

Active Oldest Votes

0

As @Kurt Revis suggested, the problem is in the mixer.

What I did: I used MIDI Monitor to verify that my outgoing messages are all correct. So my code is OK. I connected my cables to a different MIDI device (a Roland Juno synth) and sent it a bunch of repeated messages: no problem. I conclude that the Mac, my code, the UM-ONE interface, and cables, are all OK. That leaves the mixer. I tried a second Behringer Powerplay P16-M, and got exactly the same results. It therefore seems likely to be a design/manufacture problem with the mixer.

What I conclude: The Behringer Powerplay P16-M mixer doesn't correctly process incoming MIDI messages that are sent sequentially to the same channel, unless they are all of the same control change type.

A Workaround: After lots of experimentation, here's what I've come up with:

1. When sending consecutive (but different) control change messages to the same channel, insert a delay of at least 500 milliseconds between messages. For example, if you send a bunch of volume messages to a channel, followed by a bunch of pan messages to that channel, there's no problem. If you want to send a bunch of interleaved volume and pan messages to the same channel, then you need to insert a delay between each message.
2. When resetting the mixer, channel 0 ("main") requires special handling. Messages sent to it need to be "insulated" from messages sent to other channels by inserting a delay of at least 500 milliseconds.

[EDIT] Better solution: Instead of inserting a delay, send a MIDI pitch bend message to the device instead (e.g., send `0xE0 0x0 0x0`). Even though the P16-M doesn't have any documented use for the pitch bend message, it's apparently enough to shake things up and get the device back in sync with the message stream. Works for me, anyway.

edited May 21 '19 at 22:33

answered Mar 22 '18 at 2:15
jt bullitt
240 3 11