

Owen's Tech Log

Laptop ensembles, midi
clock synchronization, rewire,
MIDIClock, Reason,
Max/MSP, live performance

Archives

3.29.2004

More MTC and Cubase

The MTC output from Cubase is identical for different tempi. I captured the data using Max/MSP for 5 seconds at 120, 153 and 200 bpm. Please read my [report](#) from last week for more information.

MTC output from my MIDI Timecode generator is slightly faster than Cubase's MTC output.

As of now, Cubase will repond only to Full Frame Messages sent by my generator. It will not respond to Quarter Frame Messages.

[Download MTC Generator](#)

- posted by Owen @ 6:42 PM

3.13.2004

From [MIDI Time Concepts](#):

SMPTE and MIDI:

Song Position Pointer is a System Common status byte F2, that is generated every six MIDI clocks. It is used as a reference to find a location in a song by counting the Song Position Pointer total and diving by 16 (4 per quarter note in a 4/4 time signature) to find the measure.

A **Tempo Map** is used by a sequencer to change tempo at certain locations in a sequence.

In order to synchronize with Song Position Pointer, a **sync device** is used to decide tempo and tempo

changes in a song and the amount of measures in a song, using SPP and tempo map. The sync device can then figure out the amount of SMPTE time that passes at certain "hits".

SMPTE and MTC:

MIDI Time Code is a digital conversion of the SMPTE Time Code that allows MIDI devices to lock to the SMPTE Time Code in real time. Every frame of the SMPTE Time Code is broken down into 4 frames of MIDI Time Code with an average of 120 times a second.

It takes a total of 8 Quarter-Frame Messages to address one SMPTE Frame. That means that SMPTE is updated every two frames.

The use of MIDI Time Code in a sequencer allows the user to make changes with the tempo of a composition and all the timing information is automatically updated to follow the new tempo marking.

MIDI Clock and MTC:

The value of the tempo change from MIDI Clock would map out a new chart (tempo map) for placement of information along the MIDI Time Code.

- posted by Owen @ 2:40 PM

3.7.2004

[Cubase and MIDI Timecode](#) - Directions on how to make Cubase the slave device to another source using MIDI

[Sync Cubase to MIDI or SMPTE/MTC](#)

[MIDI Timecode Specification](#)

[MIDI Timecode](#) - Detailed Description

- posted by Owen @ 7:35 PM

3.1.2004

MIDI Timecode

[MTC and Max/MSP](#)

[FTP site](#) with a good tutorial on Max/MSP and MTC

[Brief Overview of MTC](#)

[More Detailed Description of MTC](#)

More Info

MTC

- does not transmit tempo data.
- uses MIDI System Messages
- provides an absolute time reference from the master device.

Do not transmit other midi data on the same ports and cables as MTC because of the volume of data and especially since you need uninterrupted flow of sync signal for timing.

New Setup: use Max/MSP as master -> generates both MTC and MIDI clock messages. Cubase can sync to MTC while Reason syncs to MIDI clock. Each user/slave has a Max patch which will allow them to send specific messages to the master if they want to change the tempo or stop/start the song.

Questions:

-> is this possible with MTC, as it doesn't transmit tempo data?

-> it seems that MTC is better suited to syncing tape and video. is it really applicable when syncing two pieces of software?

- posted by Owen @ 6:18 PM

2.28.2004

Some Interesting Links:

[Lots of MIDI Info](#)

[Kromozone](#) - A piece of software built in Max/MSP that uses TCP/IP to sync multiple computers over a network.

[Laptop DJ's](#) - A discussion on the current software situation (well... two years ago).

[GDAM](#) - Apparently a nice piece of unix-based djing software (link may be broken).

[More on GDAM](#)

[John Paul Young's Portfolio of Research](#) - A good source of info on interactive music over the web.

[Experimental Music in the Bay Area](#)

- posted by Owen @ 9:30 PM

2.15.2004

About Cubase and MIDI clock:

[Limitations of Cubase SX](#)

Cubase can only act as a master, though two version of Cubase on different computers can be synced using the 'VST System Link'

->VST System link is transmitted over digital audio cable (S/PDIF, ADAT, TDIF, AES)

Cubase can send MIDI clock messages to Max/MSP on the virtual ports, as well as to external MIDI devices.

->same accuracy as MCC and MidiClock

If Cubase is open, and MidiClock is opened afterwards, Cubase will crash.

MIDI Machine Control a protocol for controlling tape transports with instructions like Start, Stop and Locate (useful???)

When the Master button is selected in Cubase's transport window, the tempo is locked.

- posted by Owen @ 5:06 PM

2.8.2004

Debugging MIDIClock Clone:

In order to use the virtual ports "to Max/MSP 1/2" and "from Max/MSP 1/2" within Max itself you need

to route the MIDI path using another application. MIDI Patchbay is one such application, available at [Pete Yandell's Site](#).

so with that accomplished, i was able to use my MidiClock Test patch to debug my MidiClock Clone. here's the data:

5 bpm = 498-499 ms
10 bpm = 249 ms
20 bpm = 124 ms between each pulse
40 bpm = 61 ms
60 bpm = 41 ms (similar value to that of the original MidiClock)
80 bpm = 30 ms
84 bpm = 29 ms
100 bpm = 29 ms (there's a problem here)

so the problem could either be with Max itself (can't send the data out fast enough), or with the 'tempo' object.

for MIDI tempo conversion (ms to bpm), also some interesting MIDI stuff:

[MIDI File Format: Tempo and Timebase](#)
[The Max Factor](#)
[Maximum MIDI](#)
[DJ'ing with Live](#)

mission accomplished. using the metro object, with a conversion from bpm to milliseconds, solved the flakiness and limitation of my initial MidiClock Clone. the problem must be with the 'tempo' object and its maximum output rate. now MidiClock and MidiClock Clone output the same data (using the MidiClock Test patch).

now let's try it out with Reason...

- the minimum tempo that Reason will receive from any external midi clock source is 20 bpm. if you send anything lower, Reason will stay at 20 bpm. The maximum bpm that Reason will accept is 500.
- though there is now a greater tempo range than MidiClock (up to 500 bpm), it is less accurate - 120 bpm from MidiClock Clone = anywhere from 120 to 126 bpm in Reason.

- lower tempi are more accurate than higher
- MidiClock seems to have some small tempo variations, as well, but not as large as my clone.

MidiClock Clone Conclusions

Download: [MCC \(version 2\)](#)

- posted by Owen @ 6:49 PM

2.1.2004

findings up to 01/27:

about the MIDIClock application (from [Granted Software](#)):

limit = 60 to 240 bpm

syncs well internally with reason (no noticeable delay)

could be more compatible with reason - only has reset, start, stop, continue

->makes it hard to sync with non-midi sources (mp3 player, etc.), need more accuracy, better restart options

->possible way of detecting tempo from an audio file?

i attempted to control the tap tempo of the midiclock application using max/msp through midi cables, to test remote control of the tempo (that is the only way to do it)

communication problems - no way for slave machines to communicate with master (for tempo change, start, stop, etc.), except with tap tempo (but this is not inaccurate or the best solution).

hardware limitations - each computer needs a usb/midi interface (with at least one free input), and master needs 2 free outputs (for a 3 computer setup).

reason seems to recognize max/msp for use with remote control and midi clock sync options
->need to transmit MIDI Clock signals to the MIDI Out that is connected to the computer running Reason

i should email cycling 74 re: rewire compatibility for OS X (what kind of priority is it?).

some links:

[MIDI quiz with answers](#)

[Sparkler: An Audio-Driven Interactive Live Computer Performance for Symphony Orchestra](#)

[CodeBLUE: A Bluetooth Interactive Dance Club System](#)

[Dekstasy](#) - DJ'ing software

- posted by Owen @ 6:52 PM

so here's some things that i discovered during the week of **01/20**. hopefully things will be a little more organized when i get the hang of using blogger and use it on a regular basis.

> Find out the **programmability and clocking capability of Reason**.

there is an option for MIDI Clock Sync and Remote Control through MIDI

Here's some info from the Reason manual:

"Using MIDI Clock, you can slave (synchronize) Reason to hardware devices (tape recorders, drum machines, stand alone sequencers, workstations etc.) and other computer programs running on the same or another computer. MIDI Clock is a very fast metronome that can be transmitted in a MIDI Cable. As part of the MIDI Clock concept there are also instructions for Start, Stop and locating to sixteenth note positions."

Reason always acts as the slave allows for adjustment for latency (compensation)

About Tempo Changes:

Again, due to the latency phenomenon, Reason needs a bit of time to adjust to changes in tempo. If there are abrupt changes in the MIDI Clock, due to drastic tempo changes in the master, you will note that Reason will require up to one measure to adjust itself to the change. How long this actually takes also depends

on the precision of the incoming MIDI Clock. The more precise it is, the faster Reason can adjust to it. If this adjustment is a problem, try to use gradual tempo changes rather than immediate ones.

When Reason is synchronized to MIDI Clock, there is no Tempo readout.

[Some software developed to work with Reason](#)

> Be an expert on **MIDI clocks** (a web page?)

From the MIDI Specification Sheets:
5 messages - clock/tempo, start, stop, continue, song position pointers

Status Data Byte(s) Description
D7----D0 D7----D0

System Real-Time Messages:
11111000 Timing Clock.
Sent 24 times per quarter note when synchronization is required (see text).

11111010 Start.
Start the current sequence playing.
(This message will be followed with Timing Clocks).

11111011 Continue.
Continue at the point the sequence was Stopped.

11111100 Stop.
Stop the current sequence.

System Common Messages:
11110010 0llllll Song Position Pointer.
0mmmmmmm This is an internal 14 bit register that holds the number of MIDI beats (1 beat= six MIDI clocks) since the start of the song. l is the LSB, m the MSB.

Propellerhead's ID #: 00H 20H 3AH

System Real Time Messages:

The MIDI System Real Time messages are used to synchronize all of the MIDI clock-based equipment within a system, such as sequencers and drum machines. Most of the System Real Time messages are normally ignored by keyboard instruments and synthesizers. To help ensure accurate timing, System Real Time messages are given priority over other messages, and these single-byte messages may occur anywhere in the data stream (a Real Time message may appear between the status byte and data byte of some other MIDI message).

The System Real Time messages are the Timing Clock, Start, Continue, Stop, Active Sensing, and the System Reset message. The Timing Clock message is the master clock which sets the tempo for playback of a sequence. The Timing Clock message is sent 24 times per quarter note. The Start, Continue, and Stop messages are used to control playback of the sequence.

Sync

Most synths allow various parameters to be synchronized to the instrument's own internal clock, or to an external source of MIDI clock signals. MIDI clock is used to lock the tempo of two or more tempo-based devices, such as sequencers. To use MIDI clock, you need to make one device the clock source (also known as the master), and transmit the MIDI clock signals from the master to all of the other devices that you want to sync.

This can mean sending the clock messages down a physical MIDI cable, for instance, from a computer-based sequencer to a hardware synth. If you're running a software synth as a plug-in inside a sequencer, the sequencer may automatically send clock signals to the plug-in. (Or it may not. Some sequencer/plug-in combinations can't be synchronized. This was a problem, for instance, with older versions of Emagic Logic when VST plug-ins were used.)

Whether you're using hardware or only software, you need to instruct the receiving device to sync to the incoming clock signals. In a hardware synth, you'd

typically do this by going to the global, master, or utility area and choosing “MIDI” as the clock source rather than internal.” A synth whose clock source is set to internal will usually follow its own tempo, cheerfully ignoring any clock signals it receives. On some synths, you can sync the speed of each LFO or other parameter to MIDI clock individually, but on other synths, external sync may be an all-or-nothing proposition. Check your owner’s manual for details.

A MIDI clock message is transmitted 24 times for every quarter-note. The technical term is “ppq,” which stands for pulses per quarter-note. MIDI clock runs at 24 ppq, no matter what clock resolution the transmitting device is set to. Your sequencer may have an internal resolution of 960 ppq, for instance, but in that case, it will transmit one MIDI clock message for every 40 internal clock ticks.

If you speed up the tempo on the master, the clock messages will get closer together, and if you slow down the tempo, they’ll get further apart. The receiving synth synchronizes its operation to the MIDI clock simply by counting the messages. For instance, if an LFO is set to sync to sixteenth-notes, it will start a new LFO cycle every six clock pulses (because there are four sixteenth-notes in a quarter-note, and $4 \times 6 = 24$).

In many synths, MIDI clock sync is not implemented as well as it could be. Here’s the catch: When synced, your LFO will cycle nicely at the same tempo as your sequencer, but there’s no guarantee that all of the LFO cycles won’t start a little too early or a little too late in relation to where the sequencer thinks the beat is. MIDI defines not only clock, but start, stop, and continue messages. It would be nice if all the MIDI-syncable LFOs in the world were designed so they’d start a new cycle on receipt of a start message, but most of them don’t. The result is, the rhythm pattern coming out of your synth may sound different each time you start your sequencer to play the song, even though the two devices are in sync.

The way to solve this problem is simple, but not elegant: You can record the audio output of your

synth (playing at the correct tempo) into the sequencer as an audio track. Once you've captured the synth's rhythm pattern, you can treat it like any other audio loop: Synchronization is no longer needed, and the synth itself can be switched off.

You can use this method when you're in a hurry, or even with a synth that doesn't offer any type of synchronization. Turning the synth's own tempo knob and then recording an audio track is often easier than setting up sync.

> Look into **Open Sound Control, Rewire**, etc.

Reason uses ReWire for internal synchronization of audio and MIDI

ReWire is not currently supported in the OS X version of Max/MSP. The method used to provide ReWire support in OS 9 is not possible in OS X. We are working on a solution and eventually plan to include ReWire support in Max/MSP, but it will not be implemented for some time.

Max and Reason can communicate using MIDI through Max's 2 internal buses.

>>i would like to look into touch screen capabilities for controlling reason and using max/msp for remote control of reason.

- posted by Owen @ 6:34 PM

1.29.2004

so, i've created my blog. wow, that feels great.

- posted by Owen @ 4:25 PM

