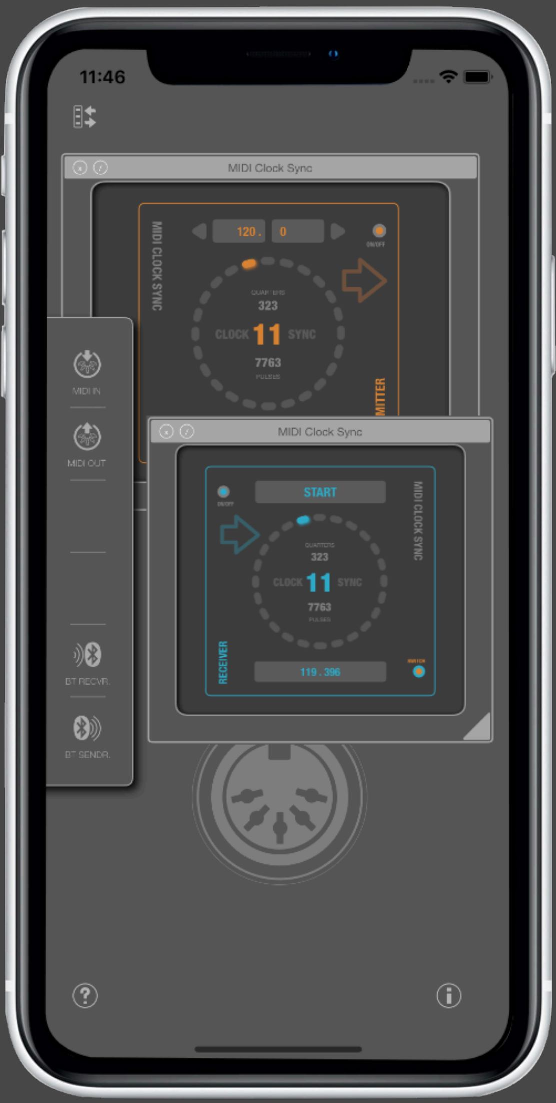
MIDI SWEET SERIES

MIDI SWEET : MIDI Clock Sync [AUDIO UNIT]

MANUAL

CREATE YOUR OWN MIDI CLOCK TRIGGERS AND CONDUCT OTHER APPS AND EXTERNAL MIDI DEVICES FOR TEMPO-SYNCHRONIZED PLAYBACK.



Preface

There seems to be much confusion, what MIDI Clock is and how it works. Now, the CLOCK signal is just a MIDI realtime message that is sent 24 times a quarter note for being able to calculate the current tempo of the sender at the location of the receiver.

It is by no means a song positioning signal, nor does it transmit any other information than its pulsing signal. Although there can be a possible combination with START, STOP and CONTINUE signals, but this is very different than for instance a concrete MIDI SPP (Song Position Pointer) signal, which is much more complex and thought for bar based song positioning.

It is also often mismatched with MTC (MIDI Time Code). this signal is in contradiction to the MC completely independent from the song tempo. The time code is exclusively expressed in hours, minutes, seconds and frames. It is correctly clocked to the current real world time starting from a certain point.

We often see people talking about these things in complete misunderstanding and mismatching of these facts. Especially some users are asking for a MTC clock synchronized to a song tempo and song position. This is complete nonsense, as these two things have nothing in common. The SPP however can deliver something that is desired here. It is synchronized to the song tempo and transmits discrete song positions regarding the musical information of a song.

MIDI Clock sync so does nothing different than just sending intervals of impulses. There is no timing information other than the time between these generated impulses. Device Synchronisation requires allot more than this.

Another fact is the widely spread misunderstanding of timing precision and jittering. It is just not true, that specialized external atom clock precision hardware devices are required to generate a good MC signal. Even inside the digital domain absolutely perfect signals and synchronisation can be created with very simple methods. The reason is that audio and MIDI on a digital device are 100% synchronized if implemented correctly, with the precision of a single audio sample at certain samplerate. The samplerate gives the resolution of i.e. 48000 or 96000 Hz, this is pulses per second, by the way.

MIDI Clock Sync is always 100% synchronous to the devices native audio clock, meaning it is absolutely sample accurate, more accurate than any external hardware can ever be. However, jittering may occur, if transmission to external equipment is used. This kind of imperfection is then caused by the transmitters, not the digital MC signal. It also can occur, if the receivers do implement the reading algorithms incorrectly per sample block instead of the given precise sample offset positions.

MIDI SWEET: MIDI Clock Sync

MIDI Clock Sync is the first product of our MIDI SWEET series and one of those tools, providing essential or experimental but useful functions for MIDI connectivity in form of Audio Units.

MIDI Clock

The classic MIDI Clock (MC) is a normed MIDI status timer signal, that is traditionally sent and received with nearly every hardware device, that relies on musical tempo information. From drum machines, arrangers, synthesizers, sequencers up to effect units, DJ mixers and lighting machines, many hard- and software devices are using this simple impulse signal to synchronize internal timing processes to the musical tempo of a selected master source.

The digital world unfortunately more and more has dropped or neglected native support for MC, which is very sad and also a really bad progression for entire music connectivity in general. Because MIDI Clock can make allot of sense, if you connect and sync external hardware devices to iOS via MIDI connections of any kind. It can even synchronize the tempo information between different playing iOS devices and internal apps via MIDI signalling in realtime. Unfortunately many iOS apps and even hosts and sequencers do not really support it or do send a jittering signal based on timer threads, which potentially introduces timing problems.

Audio Unit

We have developed a visual Audio Unit, that can create and handle user defined MIDI Clock signals and bring back this technology to your fingertips and control. Even if your host does not support MC, you can load the audio unit and generate your own.

Please note, that this Audio Unit is (intentionally) defined as MIDI enabled audio effect.

MIDI Clock Sync will register as an audio effect (or music effect) but it will not produce any sound. The audio part is required, because it needs a sample processing block to sync the generated impulses exactly to the running audio clock, which is always 100% stable. Our unit is therefore highly precise, that means sample accurate, because it is not based on any timers nor does it depend on the current processor load or such. It is always generated in absolute sync to the timing stable audio clock (sample stream) of a running iOS device. The audio engine must be continuously running, only this way the precise MIDI clock can be generated.

Please also note, that the UI updating thread is completely separated from the audio processing thread therefore, so that the clock impulses are always tight, even if the graphics performance may be overloaded or throttled down by the system for some reasons.

From a technical view of sight, MC will calculate and send 24 exactly timed MIDi status message bytes per (imaginary) quarter note with correct sample frame offsets. There is no time code transmitted nor is it indicated when a quarter note was reached or something like that. The anonymous impulses are sent continuously without any counting and it depends on the receiver, how fast it reacts to timing clock changes or whether these signals will be used or just ignored.

Transmitter versus Receiver

The MIDI Clock Sync Audio Unit has 2 basic modi. The first mode is a MIDI Clock generator (transmitter). It generates and sends the MC impulses with the precision of a user definable fixed floating point tempo value, that can be adjusted by sliding the tempo controls between 10,000 to 480,000 BPM (beats per minute). A visual running circular light will indicate the running state.

The second mode (receiver) is able to receive ANY valid incoming MC signals and then calculates and displays the resulting tempo (i.e. from external hardware devices or from other internal master sources). Thus, the transmitter mode can be seen as a master clock source and the receiver mode as a virtual slave device.

We implemented the slave mode for testing purposes merely, due to the fact, that MC signalling is not very well documented by most music apps, providing it. It is often even unclear, whether a signal is really sent or received or not. We have checked some hosts claiming to send the signal, but they actually didn’t. MIDI Clock Sync receiver can check the state easily and give you a visual feedback of the MC functionality and its progression. So the two modi do work completely independent of each other.

If the unit is switched to transmitter mode, it will not receive external signals and vice versa.

The distribution app, for instance, loads 2 instances of the audio unit into two virtual plugin windows onto the screen at same time, to demonstrate both modi of the audio unit at ones. The windows can be arranged and resized. This app is basically a minimal Audio Unit host, that can handle MIDI input (external, internal and those signals, coming from loaded audio units).

The Transmitter (orange) acts as the sender of a MIDI Clock signal, the Receiver (blue) acts as a slave device, receiving the signal via internal MIDI loopback and then calculates the resulting tempo. In other words, the MC stream is sent to the host app by the transmitter, the host just forwards the MIDI signal back to the receiver. This all should be sample accurate and not relay on additional timers but the audio clock (sample rate). Otherwise there is anything wrong in the hosts conception.

These both instances of MIDI Clock Sync can be switched to their inverted mode and thus change their roles on the fly. The demo app practically does not make any sense, it is merely for a demonstration, but it has advanced MIDI connectivity, so you could practically send and test received MC signals with some activated external MIDI ports.

Usage

The MIDI Clock Sync audio unit itself is easy to use and self-explaining. Please note, that these kind of plugins generally require a host application, that supports MIDI callbacks and actually is able to handle such MIDI loopback messages by forwarding to a useful destination. Such destinations can be internal and external MIDI devices and apps, connected to the MIDI ports.

Please note, that external connections may degrade the signal stability due to baud- and transmission rates and such.

Some users may ask, why they should need a MIDI Clock Sync generator as an audio unit, if their host app actually can do this. Well, most hosts implement one master clock, nothing more and nothing less. Some hosts even do not implement it at all. Imagine, you could connect and sync different devices with different tempi or you just want to start additional playback of external equipment, tempo synchronized in a live session or via automation and so on. The possibilities are complex and will not be exercised in detail here.

The MIDI Clock Sync transmitter mode additionally sends MIDI Start/Stop/Continue messages, if you press the Start/Stop button, which could be very useful for some external hardware stuff or to conduct a chain of connected apps and devices with the touch of a finger or via automation. It also can be switched off, effectively resetting the internal counters.

The receiver mode can show incoming MIDI Start/Stop/Continue and Reset messages and process automated parameter changes but it will not generate anything. other than visual information. Although, there is a MIDI Thru parameter implemented, which optionally loops all received MIDI messages back to the host app.

Please note, that only messages sent to a rendering process in receiver mode will be looped back. There is a certain danger of creating endless message loops this way.

The default state of a loaded MIDI Clock Sync audio unit is the receiver mode (blue). You can switch the mode by tapping the switch control at the bottom of the user interface.