

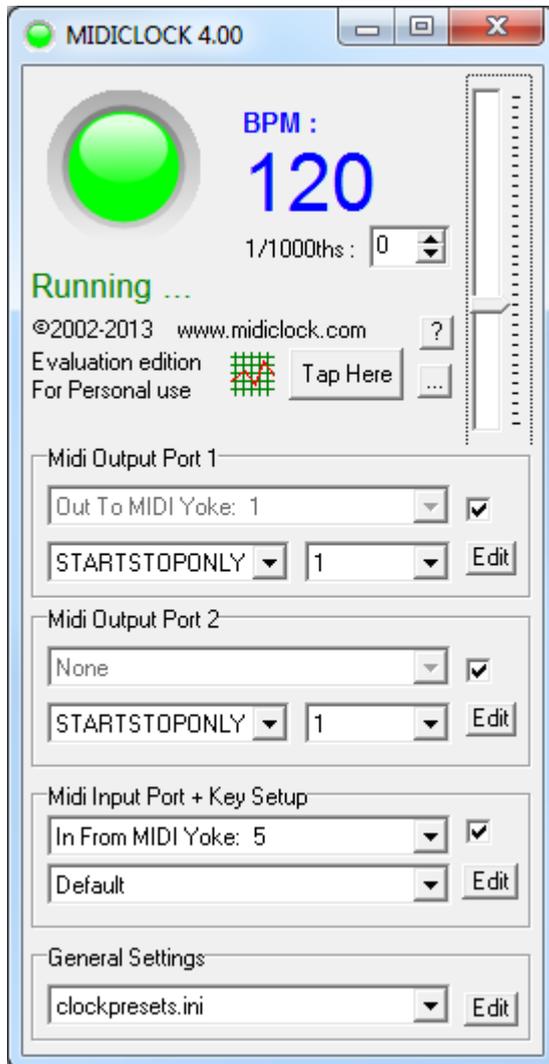
# MIDIClock 4.01

Quick Guide by Serge

Rev 4

12 apr 2014

[www.midiclock.com](http://www.midiclock.com)



# Introduction

- What is MIDIClock ?
  - A program that can be used to synchronize synths, effects, sequencers, arpeggiators, ...
  - Its main function is generating a midi beat clock signal on a midi output device at a selectable bpm rate. Other output signals are also possible through custom output drivers
  - The software has no installer, simply unzip to where you need it. It is portable : it can be run from a removable disk/stick without installing anything.
  - The latest trial version of MIDIClock can be downloaded from [midiclock.com](http://midiclock.com).

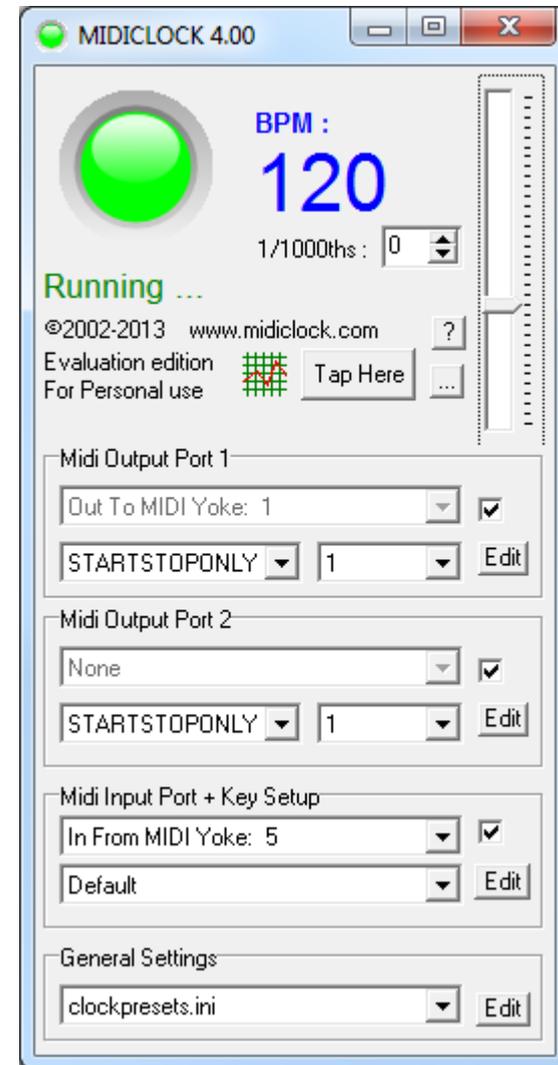
# What is the midi clock protocol?

## From wikipedia :

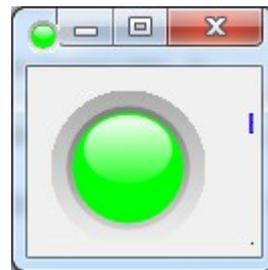
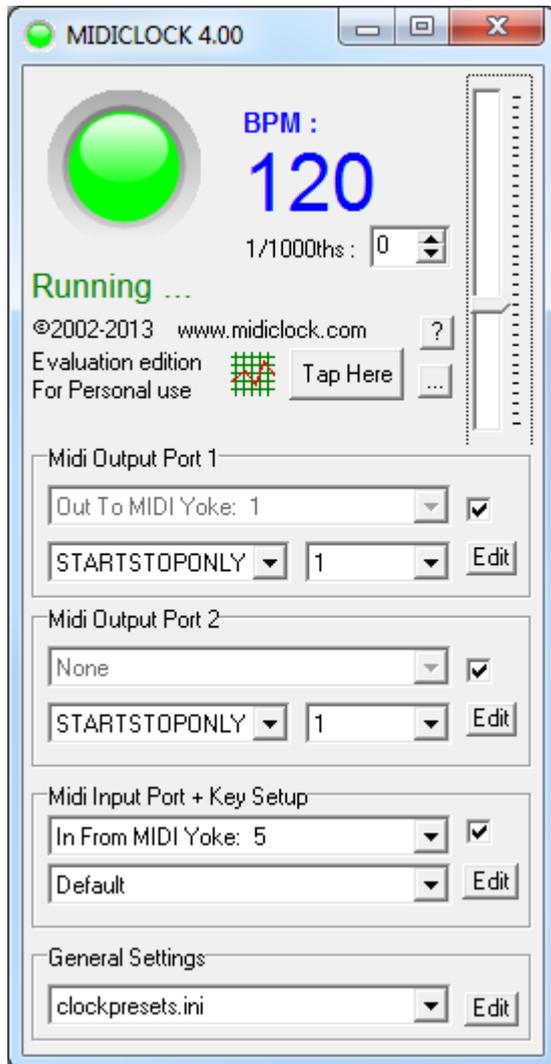
- MIDI beat clock is a clock signal that is broadcast via MIDI to ensure that several synthesizers stay in synchronization. It is not MIDI timecode.
- Unlike MIDI timecode, MIDI beat clock is sent at a rate that represents the current tempo, at 24 ppqn (pulses per quarter note). It is used to maintain a synchronized tempo for synthesizers that have BPM-dependent voices and also for arpeggiator synchronization. It does not transmit any location information (bar number or time code) and so must be used in conjunction with a positional reference (such as timecode) for complete sync.
- MIDI beat clock defines the following real time messages:
  - \* clock (decimal 248, hex 0xF8)
  - \* start (decimal 250, hex 0xFA)
  - \* continue (decimal 251, hex 0xFB)
  - \* stop (decimal 252, hex 0xFC)
- All the above real-time messages are supported by MIDIClock

# MIDIClock Features :

- 2 MIDI output drivers supported
- Up to 32 midi simultaneous MIDI input action mappings with MIDI learn function
- Fractional BPM rates from 0.001 BPM to 1000 BPM
- Highly customizable through INI files.
- Stable, usable for gigs, studio, live shows,...
- Registration & Support : [midiclock.com/upgrade](http://midiclock.com/upgrade)

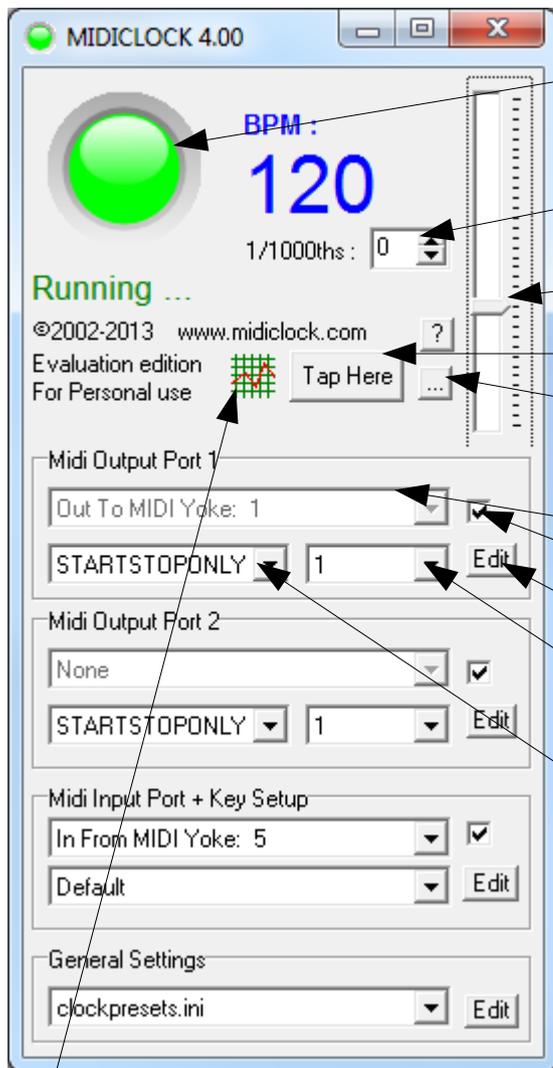


# MIDIClock Shapes



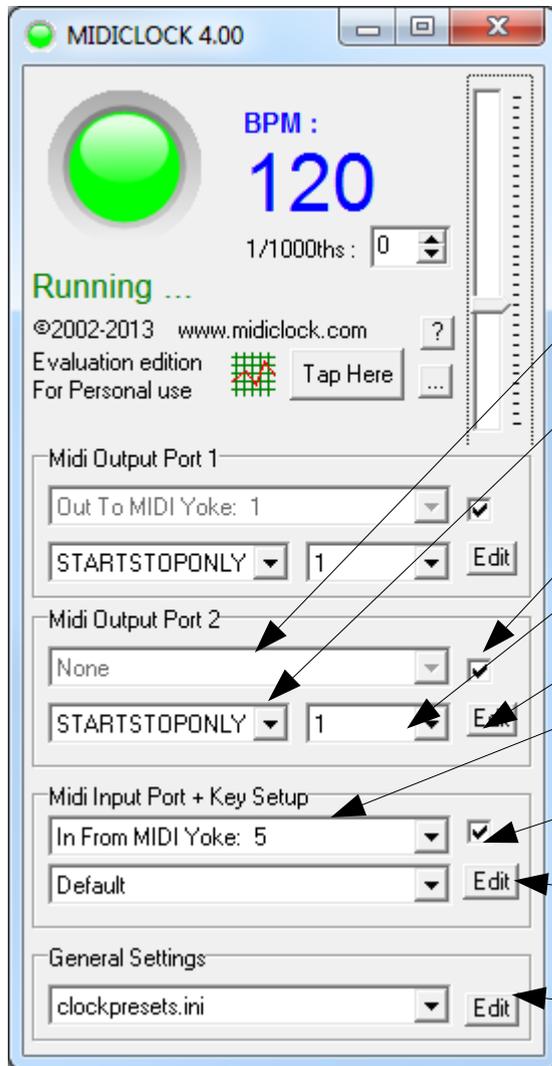
The form size of your midiclock can be resized to reduce the claimed screen real estate.

# Main Screen functionality (1/2)



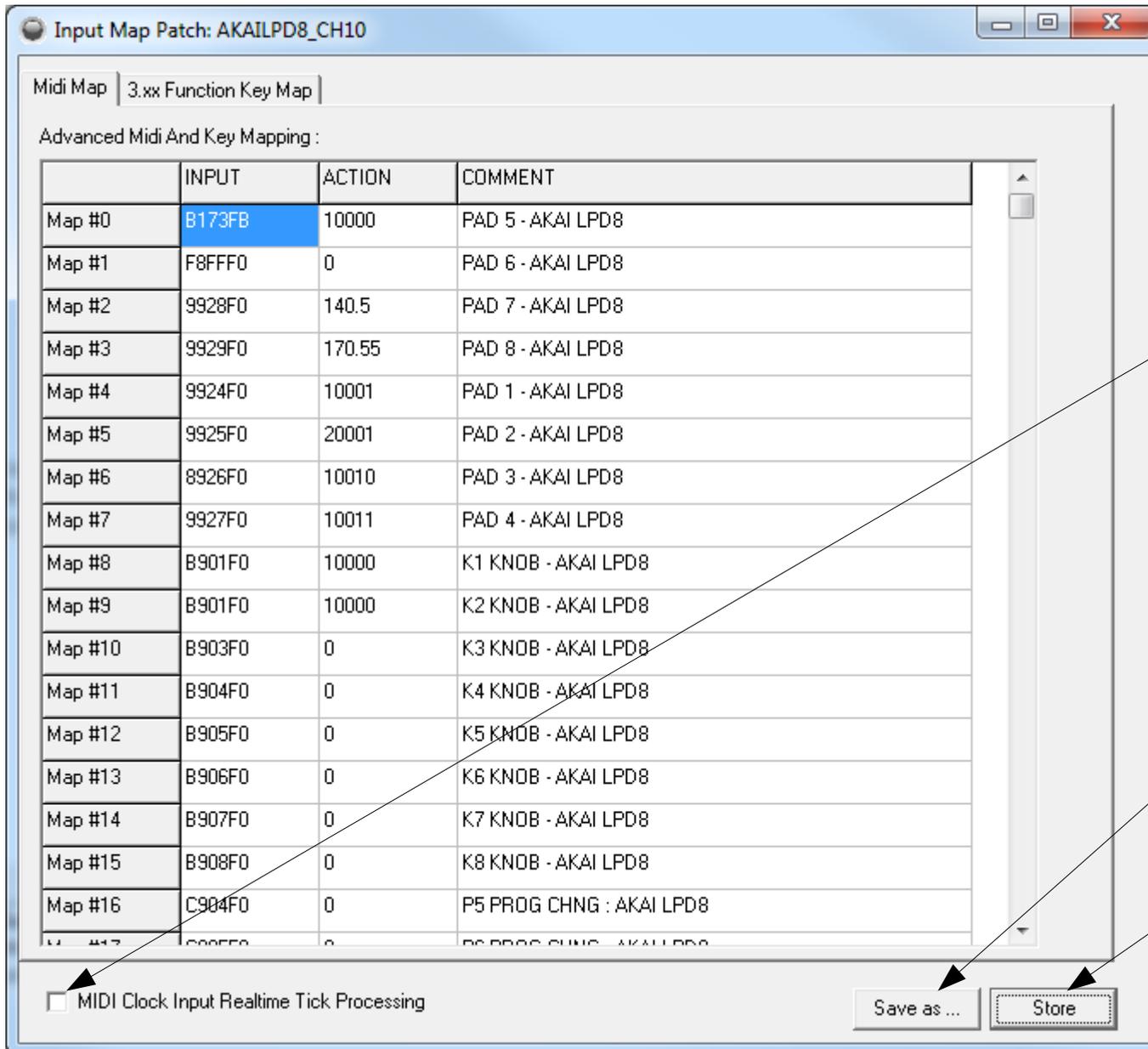
- Start /stop the timer
- Fractional part of the BPM
- Main BPM rate slider
- Tap tempo input button
- Small form factor
- Device #1 Selection
- Device #1 Enable/Disable
- Device #1 Config editing
- Device #1 Tempo divider
- Device #1 Output driver
- Tempo Chart Visualisation

# Main Screen functionality (2/2)



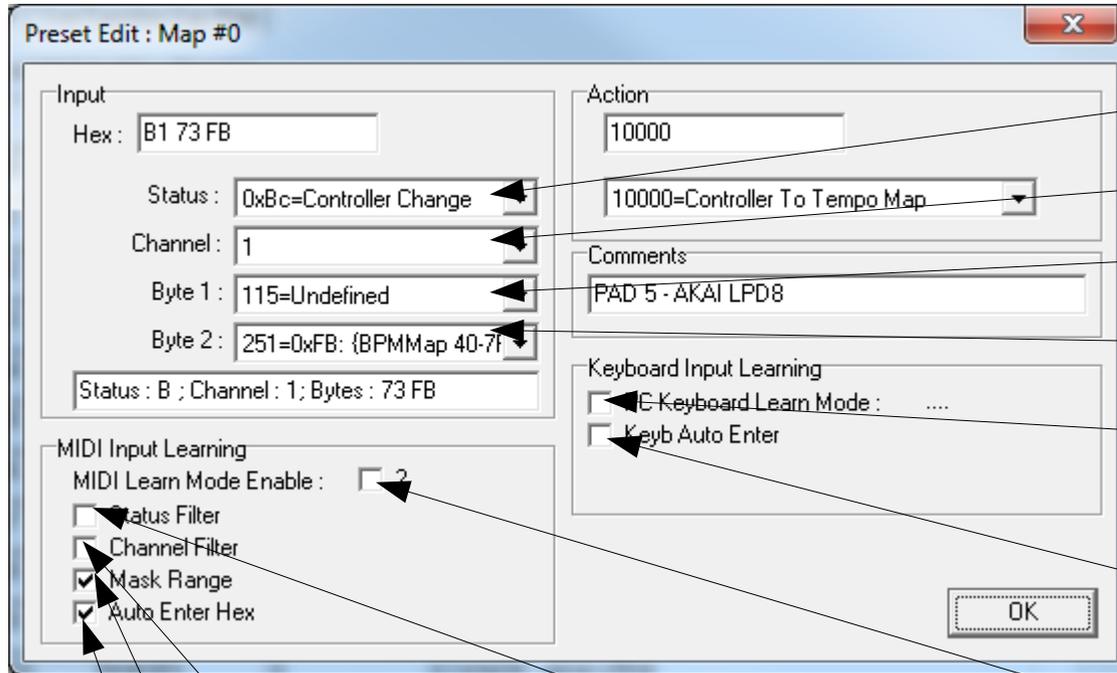
- Device #2 Selection
- Device #2 Output driver
- Device #2 Enable/Disable
- Device #2 Tempo divider
- Device #2 Config editing
- Input MIDI Device Selection
- Input MIDI Device Enable/Disable
- Input Map Editing
- Preset patch selection/edit

# Input Patch Editor



- MIDI command to action mapping
- Click on any line to edit the mapping
- Enable Realtime MIDI Clock Input processing : this makes midiclock respond to an incoming midi clock signal
- Save as a new input driver
- Store as current input driver

# Editing Input Map #0 (1/2)



Status byte

Midi channel

Parameter 1 of midi message

Parameter 2 of midi message

**PC Keyboard Learning function** : → shows hex code for the key pressed

Keyb Auto Enter : enters key code in fields automatically

**MIDI Learning function** --> updates map to incoming midi date. Use this for quick mapping.

Filter on status byte in learning mode

Filter on selected channel for incoming data in learning mode

Apply the mask range when data comes in, in learning mode

Automatically update the hex code in the editor on input

# Editing Input Map #0 (2/2)

Preset Edit : Map #0

Input  
Hex : B1 73 FB  
Status : 0xBc=Controller Change  
Channel : 1  
Byte 1 : 115=Undefined  
Byte 2 : 251=0xFB: {BPMMMap 40-7f  
Status : B ; Channel : 1 ; Bytes : 73 FB

Action  
10000  
10000=Controller To Tempo Map

Comments  
PAD 5 - AKAI LPD8

Keyboard Input Learning  
 PC Keyboard Learn Mode : ....  
 Keyb Auto Enter

MIDI Input Learning  
MIDI Learn Mode Enable :  ?  
 Status Filter  
 Channel Filter  
 Mask Range  
 Auto Enter Hex

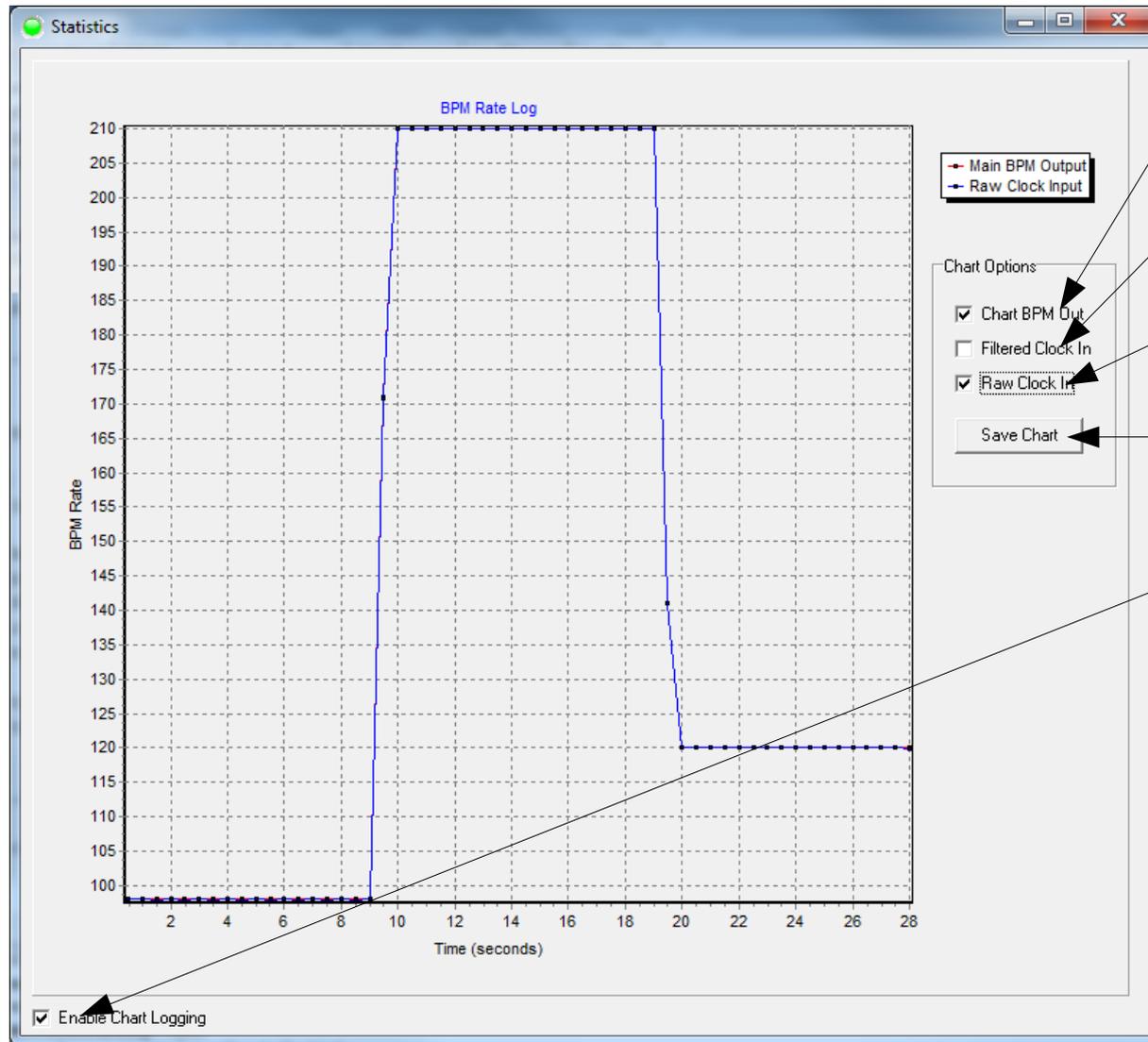
OK

Action mapped to the defined input midi code (for a list of actions see the special commands description at the end of this document)

Descriptive comment for this map

Press ok to apply the settings

# Tempo Chart Visualisation



• Enable Output BPM rate display

• Enable display of Filtered Clock Input Values

• Enable display of Raw Clock Input Values

• Save chart as BMP

• Enable/Disable chart logging

# Keyboard Shortcuts

- CTRL+ALT+S : save current settings to clockpresets.ini. It creates the file if it does not yet exist.
- F1/RETURN : Start/Stop timer toggle
- F2 : Pause timer
- F3 : Continue/Resume timer
- F5 : set BPM to Preset #5
- CTRL+F5 : Store current BPM rate at preset #5
- F6/F7/F8 : Change BPM to Preset #6/7/8
- CTRL+F6/F7/F8 : Store BPM at preset #6/7/8
- SPACE : tap tempo trigger
- Any custom key can be mapped through the input map !!

# File Overview

- **MIDIClock.exe** : The application binary. Clicking this application starts up midiclock.
- **clockpresets.ini** : Startup application presets. These are saved by pressing CTRL+ALT+S.
- **MidiclockRegistration.ini** (optional) : contains the key for registered users.
- **\*.mcp** files : alternative application preset patches. Same file format as clockpresets.ini.
- **outdrv subdirectory** : contains output drivers, such as :
  - GMBASSDRUM.ini : a GM bass metronome
  - GMCLICHEROCK1.ini : a typical GM pop-rock drum rhythm
  - G71UT.ini : interfacing to a Zoom G7.1ut effect pedal
- **indrv subdirectory** : contains input drivers

# Clockpresets.ini layout (1/3)

- **[APPLICATION\_PRESETS]** ; Section indicating general application presets
  - MAINSCREENPOSITIONTOP=155 ; Main screen top pixel position at program startup
  - MAINSCREENPOSITIONLEFT=565 ; Main screen left pixel position at program startup
  - DEFAULTSPEED=120 ; BPM Rate used at program startup
  - SHOWFLASHING=1 ; 0 = disable window color flashing ; 1=enable window flashing at ¼ beat
  - FULLFLASHING=0 ; 1 = enable full window flashing ; 0 = disable
  - FLASHINGDURATION=1000 ; Color duration (unit = 1/24<sup>th</sup> of a beat). Set to 1000 for normal use.
  - FLASHINGPERIOD=24 ; Cycle period for the flashing, define in 1/24<sup>th</sup> of a beat units
  - IGNOREMOUSEWHEEL=0 ; 1=Disable mouse wheel influence on bpm rate
  - IGNOREKEYBOARDTAP=1 ; 1=Ignore keyboard 'SPACE' for tapping and 'RETURN' for start/stop
  - ALWAYSONTOP=0 ; 1= keep application window always in front of other applications
  - AUTOSTARTONTAP=0 ; 1= automatically start the midi clock signal at beat 1 after tapping beat 4
  - TAPTEMPOTIMEOUT = 5000 ; Tap tempo input state reset timeout (in milliseconds)
  - TAPTEMPOCOUNT=2 ; number of taps needed to trigger a tempo change (2, 3 or 4)
  - BPMROUNDING=0.001 ; rounding step taken into account when determining the BPM rate
  - BPMMINIMUM=0.001 ; the minimal BPM rate selectable in the midi clock GUI
  - BPMMAXIMUM=330 ; the maximal BPM rate selectable in the midi clock GUI
  - DONTASKSAVESETTINGS=0 ; 1= Shows “do you want to save settings”
  - CLOSEMINIMIZESWHENRUNNING=0 ; 1 = Closing the application minimizes it when the clock is running

# Clockpresets.ini layout (2/3)

- **[MIDIOUTPUTPORT1]** ; section containing the midi port #1 application settings
  - MIDIPORTINDEX=1 ; Selection Order number of the midi port #1
  - MIDIPORTNAME=Out To MIDI Yoke: 1 ; Device name of the midi port #1
  - DEVICETYPE=MidiClock ; Output driver selected for this midi port #1
  - CLOCKDIVIDER=1 ; Divider used on the clock signal on port #1
- **[MIDIOUTPUTPORT2]** ; section containing the midi port #2 application settings (same structure)
  - MIDIPORTINDEX=9 ; Selection Order number of the midi port #2
  - MIDIPORTNAME=E-MU Xmidi1X1 ; Device name of the midi port #2
  - DEVICETYPE=MidiClock ; Output driver selected for this midi port #2
  - CLOCKDIVIDER=1 ; Divider used on the clock signal on port #2
- **[MIDIINPUTPORT1]** ; section containing the midi input port application settings
  - MIDIPORTINDEX=2 ; Selection Order number of the midi input port
  - MIDIPORTNAME=In From MIDI Yoke: 3 ; Device name of the midi input port
  - DEVICETYPE=AKAILPD8\_CH10 ; Input driver selected for this midi port

# Clockpresets.ini layout (3/3)

- **[BPM\_PRESETS]** ; section containing BPM rate preset values.
  - F5BPM=100 ; pressing F5 will set the BPM rate to 100
  - F6BPM=120 ; pressing F6 will set the BPM rate to 120
  - F7BPM=140 ; pressing F7 will set the BPM rate to 140
  - F8BPM=160 ; pressing F8 will set the BPM rate to 160

Note : since 4.00, custom keymaps can be set up in the input map editor. The F5/F6/F7/F8 presets are supported for backwards compatibility

# Input drivers : General Layout

- **[SPECIAL\_COMMANDS]** ; Powerful description of remote controlling midi clock by MIDI messages
  - INPUT0=933C40 ; MIDI Message to be parsed. (replace 933C40 with your midi message in hex)
  - ACTION0=150 ; Action to be taken when receiving that MIDI message.
  - ...up to INPUT31 : 32 INPUT/ACTION pairs can be defined.
- INPUT specifier :
  - Specifies the 3 midi bytes in hex (status byte + 2 data bytes) on which midiclock reacts
  - When the status byte is 0x60, the input is a PC keyboard keypress. The 2nd data byte determines the key code.
  - For example the midi message 933C40
    - The keyboard plays middle C (note number 60, hexadecimal 3C) on channel 4 (the zero nibble of 94), at half the full velocity (velocity 64, hexadecimal 40).
    - INPUT0=933C40 ; this would define the above midi message as input signal
    - ACTION0= 150 ; the action would be to set the bpm rate to 140 bpm
  - What about velocity sensitive keyboards ?
    - Use masking : using F0 as data byte 2 will make midiclock trigger on all non zero velocities
    - Example : INPUT0=933CF0 : for every middle C played (note on event), the ACTION0 will be run
  - ACTION0=130 ; example : set BPM RATE to 130 when the INPUT0 MIDI Message is received
  - More advanced options and usage for INPUT/ACTION will be described on the next pages.

# Input drivers : Actions

- The following target action codes can be executed on incoming midi commands :
  - 0 = do nothing
  - 1 --> 200 = set this specific BPM value
  - 10000 = perform midi byte 2 to tempo transformation (see further on for the input masking options)
  - 10000.1 = perform midi byte 1 to tempo transformation (see further on for the input masking options)
  - 10001 = start-stop
  - 10002 = pause
  - 10003 = continue
  - 10004 = start
  - 10005 = stop
  - 10006 = decrease tempo
  - 10007 = increase tempo
  - 10008 = tempo - 2
  - 10009 = tempo + 2
  - 10010 = double tempo
  - 10011 = halve tempo
  - 10016 = tempo – rounding step
  - 10017 = tempo + rounding step
  - 10101 = beat reset
  - 20001 = tap tempo trigger
  - 20001.1 = single byte special tap tempo trigger : ignores byte 2 and uses byte 1 only for range limits  
(can be used for 'any key' tap tempo)

# Input drivers : Tempo Transformation Input Masks (1/3)

- In case of an action = 10000 --> several midi value to tempo transformation can be selected by input midi data byte 2
- In case of an action = 10000.1 --> several midi value to tempo transformation can be selected by input midi data byte 1
- This is useful for mapping a specific controller range or patch onto a tempo range of choice.
- For byte E0 – EF, a custom transformation can be defined in the input driver. For example, for E0 :
  - [CUSTOMRANGE\_TRANSFORMS]
  - RANGEMIN\_E0=00
  - RANGEMAX\_E0=7F
  - RANGEMULTIPLIER\_E0=1.5
  - RANGEOFFSET\_E0=30
- Specific values can also be overwritten by an explicit map, for example E1 :
  - [CUSTOMRANGE\_TRANSFORMS]
  - RANGEMIN\_E1=00
  - RANGEMAX\_E1=7F
  - RANGEMULTIPLIER\_E1=1
  - RANGEOFFSET\_E1=0
  - RANGEMAP\_E1=CUSTOMRANGE\_MAP2

# Input drivers : Tempo

## Transformation Input Masks (2/3)

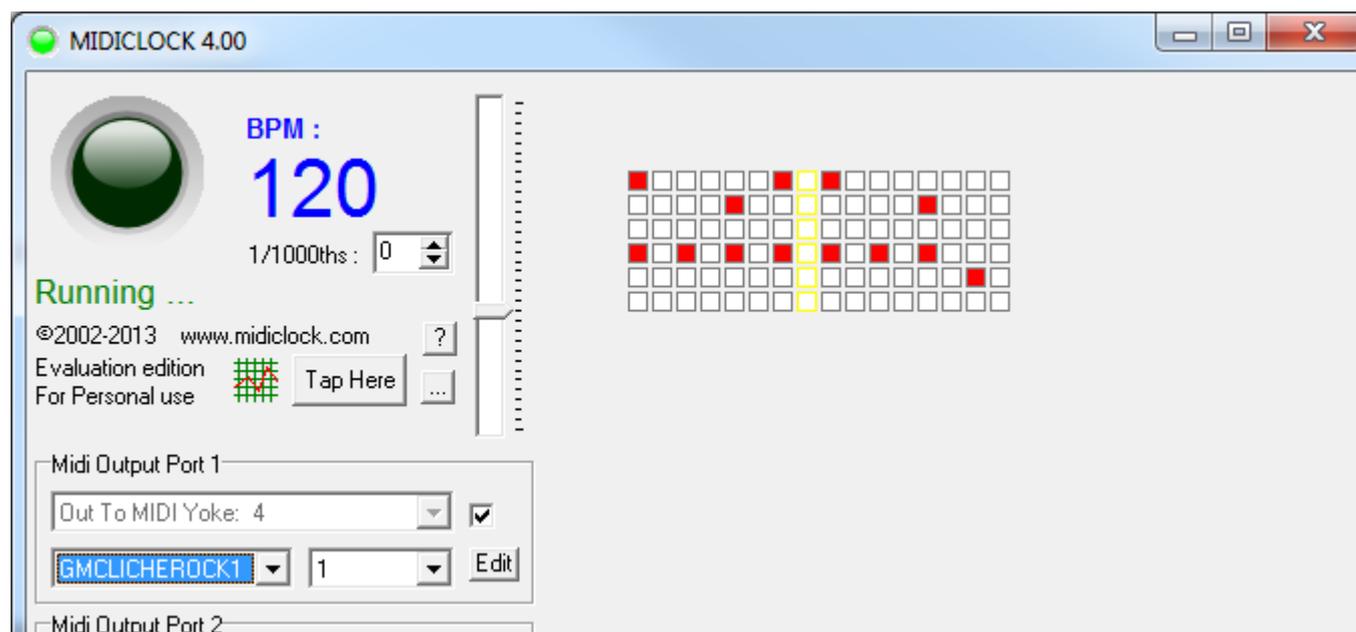
- This custom map can then be defined in a separate section. You can design your own mapping curve from a spreadsheet this way.
- For example :
  - [CUSTOMRANGE\_MAP2]
  - 0=60
  - 1=60.72
  - 2=61.45
  - 3=62.19
  - 4=62.93
  - 5=63.69
  - 6=64.45
  - 7=65.23
  - 8=66.01
  - 9=66.8
  - 10=67.6
  - 11=68.41
  - 12=69.23

# Input drivers : Tempo Transformation Input Masks (3/3)

- Midiclock will also transform the incoming data byte according to these fixed formulas :
  - Mask byte 0xF0: From 0x00 to 0x7F --> multiplier = 1.58 and offset = 0
  - Mask byte 0xF1: From 0x00 to 0x7F --> multiplier = 1 and offset = 0
  - Mask byte 0xF2: From 0x00 to 0x7F --> multiplier = 2 and offset = 0
  - Mask byte 0xF3: From 0x00 to 0x7F --> multiplier = 3 and offset = 0
  - Mask byte 0xF4: From 0x00 to 0x7F --> multiplier = 4 and offset = 0
  - Mask byte 0xF5: From 0x00 to 0x7F --> multiplier = 1 and offset = 40
  - Mask byte 0xFA: From 0x00 to 0x3F --> multiplier = 2 and offset = 0
  - Mask byte 0xFB: From 0x40 to 0x7F --> multiplier = 2 and offset = 0
  - Mask byte 0xFC: From 0x00 to 0x3F --> multiplier = 2 and offset = 50
  - Mask byte 0xFD: From 0x40 to 0x7F --> multiplier = -2 and offset = 306
  - Mask byte 0xFF: From 0x01 to 0x7F --> multiplier = 2 and offset = 0
- These mappings cannot be changed, and are kept in 3.xx releases for backwards compatibility

# Output drivers : General

- Midiclock contains a framework for custom MIDI output protocols. Those are located in the outdrv subdirectory.
- Midiclock includes a tiny tight sequencer for each output driver.
- Visual display of the beat position is possible
- Currently beat editing occurs through ini editing (see next page)



# Output drivers : Configuration

The subdirectory outdrv contains the selectable output drivers.

I will explain how this works, by explaining some examples :

Sample Midiclock.ini explained.

```
[DEVICECONFIGURATION]
DEVICETYPE=MidiClock           ; Name of the device
DEVICE_CLOCKOUTPUT_ENABLE=1   ; send out midiclock commands (F8)
```

Sample GMBOINKY.ini explained

```
[DEVICECONFIGURATION]
DEVICETYPE=GM BOINKY           ; Name of the device
DEVICE_SYSEXINIT_STRING=       ; sysex data to be sent on driver initialisation
DEVICE_CLOCKOUTPUT_ENABLE=0    ; 1 = sends out midiclock (F8) bytes, 0 = doesn't send F8
DEVICE_SYSEX_ENABLE=0         ; 1 = enables sysex output, 0 = no sysex output
DEVICE_SYSEXRELEASE_STRING=    ; sysex data to be sent on driver release
DEVICE_SYSEXTEMPOSET1_STRING=  ; sysex data to be sent before the tempo code (on change)
DEVICE_SYSEXTEMPOSET2_STRING=  ; sysex data to be sent after the tempo code (on change)
DEVICE_SYSEX_VALOFFSET=40      ; offset to be added to the sysex tempo code
DEVICE_TRACKS_ENABLE=1        ; 1= enables track sequencing, 0 = disables track sequencing
DEVICE_TRACKS_PROGRAMCHANGE=35 ; Default prog change nr. to be sent before starting playback
DEVICE_TRACKS_CHANNEL=0       ; Default channel to be used by tracks
DEVICE_CLOCKOUTPUT_NOSTARTATSTART=1 ; 1= send no start byte (FA) when the clock is started
DEVICE_CLOCKOUTPUT_NOSTOPATEND=1 ; 1= send no stop byte (FC) when the clock is stopped
```

# Output Drivers : Sequences

Beat sequences can be customized.  
This currently happens by creating/editing an output driver.  
You can find examples in the outdrv subdirectory.

GMBOINKY.INI is an example of a sequence.

```
[DEVICECONFIGURATION]
DEVICETYPE=GM BOINKY
DEVICE_SYSEXINIT_STRING=
DEVICE_CLOCKOUTPUT_ENABLE=0
DEVICE_SYSEX_ENABLE=0
DEVICE_SYSEXRELEASE_STRING=
DEVICE_SYSEXTEMPOSET1_STRING=
DEVICE_SYSEXTEMPOSET2_STRING=
DEVICE_TRACKS_ENABLE=1
DEVICE_TRACKS_PROGRAMCHANGE=35
DEVICE_TRACKS_CHANNEL=0
DEVICE_CLOCKOUTPUT_NOSTARTATSTART=1
DEVICE_CLOCKOUTPUT_NOSTOPATEND=1
[TRACKS]
C2247400=1000100010001000
C3307400=0010001000100100
C43C7400=0000000000000010
C5487400=0000000000000001
```

You can setup some parameters that are enforced on initializing the driver :

- DEVICE\_TRACKS\_CHANNEL : default midi channel used (when not specified)
- DEVICE\_TRACKS\_PROGRAMCHANGE : program nr used for default midi channel

Next up you can add TRACKS

C2247400=1000100010001000

So the key is built up like this :

track name = C2  
note number = 24  
note volume = 74  
track channel = 00

an optional X can be put behind that, to indicate no note-off should be sent.

the track events are simply  
1 = occurrence  
0 = no occurrence

This way you can create a complete sequence



# Output Drivers : CTRL output

The CTRL OUTPUT feature is intended to allow controlling effect devices by setting a delay value parameter through a midi controller. (requires midiclock 4.01 or higher)

- `DEVICE_CTRLLOUT_ENABLE=1` ; 1= enable CTRL driver
- `DEVICE_CTRLLOUT_CONVERT_BPM_TO_MS=1` ; 1 = convert BPM to millisecond, 0 = don't convert/
- `DEVICE_CTRLLOUT_FLOOR_OUTPUT_BYTE=0` ; 1 = floor the output byte : floor example 3.9 --> 3  
; 0 =rounding will be applied : round example : 3.9 --> 4
- `DEVICE_CTRLLOUT_BPM_OFFSET_VALUE=0` ; offset added to BPM value, this can be negative, fractional,...
- `DEVICE_CTRLLOUT_BPM_MULTIPLIER=1` ; multiplier for bpm, this can be fractional, should be > 0.0001
- `DEVICE_CTRLLOUT_MS_MULTIPLIER=0.1` ; multiplier applied to the millisecond value
- `DEVICE_CTRLLOUT_MIDICHANNEL=3` ; midi channel used for the control output
- `DEVICE_CTRLLOUT_MIDICTRL=5` ; controller value used for the output

**Note concerning midi channel notation : MIDIClock starts counting from 0, most synths start counting from 1. So when you set your midi channel to 1 on your synth, you probably use midi channel 0 in MIDIClock**

# Any Questions ??

- Visit [www.midiclock.com](http://www.midiclock.com) for more information...