**Mestra**

**Combined Controller**

Triggers

Michel Keijzers, © 2017

# History

Table : History

|  |  |
| --- | --- |
| **Date/period** | **Actions** |
| Nov 8, 2017 | Initial version |

# Table of Contents

Contents

[History 1](#_Toc498126522)

[Table of Contents 2](#_Toc498126523)

[List of Tables 2](#_Toc498126524)

[List of Figures 2](#_Toc498126525)

[1 Introduction 3](#_Toc498126526)

[2 Folder Structure 4](#_Toc498126527)

# List of Tables

[Table 1: History 1](#_Toc498126528)

# List of Figures

# Introduction

This document describes the trigger for the combined Controller and MIDI/DMX slave.

# MIDI Triggers

## Examples

trigger MIDI NoteOn MC=1 Note=C1 Transpose5;

trigger MIDI NoteOn MC=1 Note=C1~ Transpose5;

trigger MIDI NoteOn MC=1 ~C3 Transpose5;

trigger MIDI NoteOn MC=1 Note=C1 Transpose5 disabled;

trigger MIDI NoteOn MC=1 Notes=C1~G#5 Transpose5 disabled;

trigger MIDI NoteOn MC=1 Notes=C1~G#5 Transpose5 disabled;

trigger NoteOn MC=1 All Transpose5 disabled;

trigger NoteOn 1 C1 DoubleOctaveHigher enabled;

trigger NoteOff 1 C1 DoubleOctaveHigher enabled;

trigger PolyAftertouch 1 All DoubleOctaveHigher enabled;

trigger PolyAftertouch 1 10~120 DoubleOctaveHigher enabled;

trigger ControlChange 1 10~120 DoubleOctaveHigher enabled;

trigger ProgramChange 10 DoubleOctaveHigher enabled;

No MC present

trigger Aftertouch 1 All DoubleOctaveHigher enabled;

trigger PitchBend 1 10~120 DoubleOctaveHigher enabled;

Only the most 8 most significant bits are used as value

trigger Sysex DoubleOctaveHigher enabled;

No filtering is used (depends on sysex data bytes

## **Syntax**

<trigger> [<midi >] <trigger\_name> <note\_on> <mc\_spec> <note\_spec> <program\_name> <enabled>;

<trigger> [<midi>] <trigger\_name> <note\_off> <mc\_spec> <note\_spec> <program\_name> <enabled>;

<trigger> [<midi> ]<trigger\_name> <poly\_aftertouch > <val\_spec> <value\_spec> <program\_name> <enabled>;

<trigger> [<midi>] <trigger\_name> <control\_change> <val\_spec> <value\_spec> <program\_name> <enabled>;

<trigger> [<midi>] <trigger\_name> <program\_change> <val\_spec> <value\_spec> <program\_name> <enabled>;

<trigger> [<midi>] <trigger\_name> <after\_touch> <val\_spec> <value\_spec> <program\_name> <enabled>;

<trigger> [<midi>] <trigger\_name> <pitch\_bend > <val\_spec> <value\_spec> <program\_name> <enabled>;

<trigger> [<midi>] <trigger\_name> <sysex> <program\_name> <enabled>;

@ means: case insensitive

<trigger>: @”trigger”

<midi>: @”MIDI”

<note\_on>: @”NoteOn” | @“On”

<note\_off>: @”NoteOn” | @“Off”

<poly\_aftertouch>: @”PolyAftertouch” | @“PolyAft”

<control\_change>: @”ControlChange” | @“Cc”

<program\_change>: @”ProgramChange” | @“PrgChg”

<aftertouch>: @”Aftertouch” | @“Aft”

<pitchbend>: @”Pitchbend” | @”PB”

<sysex> @”SystemExclusive” | @”Sysex”

<note\_spec>: ( [@“Notes=” | “”] (@“ALL” | <note\_range>)) |

( [@“Note=” | “”] <note>)

<note\_range>: <note\_1> “~“ <note\_2>)| @”All”

where note\_1 = note, note\_2 = note, note1 <= note\_2

<note>: (“C” | “D” | “E” | “F” | “G” | “A” | “B” ) (“#” | “b” | “”) (““0” .. “10”)

, check value 1<=note <=127, G10 = 127

<value\_spec>: ( [@“val=” | @”value=”| “”] (@“ALL” | <value\_range>)) |

( [@“CC=” | “”] <cc>)

<value\_range>: <value1> “~“ <value2>)| @”All” where value1 = value, value2 = value,

value1 <= value2

<value>: 0 <= value <= 127

<mc\_spec> : ( [@“mc=” | “” ] <mc\_nr>)

<mc\_nr>: ( “1” | “2” .. | “15” | “16” )

<identifier>: <alpha\_char> + <id\_char>\*

<alpha\_char>: (“A”| “B” | ..| “Y” | “Z” | “a” | “b” | .. | “y” | “z” | ‘\_’ )

<digit>: (“0” | “1” | .. | “8” | “9” )

<id\_char>: <alpha\_char> |<digit>)

<enabled\_status>: [ @”enabled” | @”disabled” ]

## Parse Structure

enum ELineType { Setup, Trigger, Program, Instruction } LineType;

enum EMidiType { NoteOn, NoteOff, Aftertouch, …. } MidiType;

int MidiChannel;

int MidiPropertyFirst;

int MidiPropertyLast;

int MidiValueMin;

int MidiValueMax;

void Parse()

{

String[] words = split to words;

case (words[0] == <trigger>

CurrentWord++;

ParseTrigger();

}

void ParseTrigger()

{

case (words[CurrentWord] == <MIDI>

CurrentWord++;

ParseTriggerMidi();

case (words(CurrentWord) == <note\_on>

CurrentWord++;

ParseTriggerMidiNoteOn();

Case (words(CurrentWord) == <note\_off>

CurrentWord++;

ParseTriggerMidiNoteOn();

}

void ParseTriggerMidiNoteOn()

{

}

## **Memory**

A trigger always has 5 bytes. The first three bytes depend on the type. Bytes 3 and 4 contains the program ID, 0-65535.

Table 1: Triggers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Byte 0** | **Byte 1** | **Byte 2** | **Name** | **Value(s)** | **Meaning/comment** |
| 1--- ---- |  |  | Enabled | 1/0 | Enabled/disabled |
| -000 ---- |  |  | Trigger Type | 0000 | MIDI |
| ---- 1000 |  |  | MIDI Type | 1000 | Note On |
| ---- 1001 |  |  | MIDI Type | 1001 | Note Off |
| ---- 1010 |  |  | MIDI Type | 1010 | Poly Aftertouch |
| ---- 1011 |  |  | MIDI Type | 1011 | Control Change |
| ---- 1100 |  |  | MIDI Type | 1100 | Program Change |
| ---- 1101 |  |  | MIDI Type | 1101 | Aftertouch |
| ---- 1110 |  |  | MIDI Type | 1110 | Pitch Bend |
|  | mmmm mmmm |  | MIDI Channel | 0~127 | MC 1 to 128 (reserved for Program Change and SysEx) |
|  |  | 0--- ---- | Elements Type | 0 | Single (non-group) (reserved for SysEx) |
|  |  | -nnn nnnn | Property Number | 0~127 | e.g. note, CC value, for pitch bend: 8 MSB bits |
|  |  | 1--- ---- | Elements Type | 1 | Group, (reserved for SysEx) |
|  |  | -1-- ---- | All values |  | e.g. all note/ccs |
|  |  | --## #### | Reserved |  |  |
|  |  | -0-- ---- | Group elements | 0 | Group |
|  |  | --gg gg-- | Group | 0~15 | 8 elements per group |
|  |  | ---- --## | Reserved |  |  |
| ---- 1111 |  |  |  | 1111 | System Common |
|  | 0000 ---- |  |  | 0000 | Sys Ex |
|  | ---- #### | #### #### | Reserved |  |  |
|  | pppp ---- |  |  |  | All other values ignored. |