**Triggers**

**NoteOn**

trigger <Name> NoteOn <McSpec> <NoteSpec> <ProgramName> <status>

@ means: case insensitive

<Name> = <identifier>

<McSpec> = ( [@“Mc=” | “” ] <Mc>)

<Mc> = ( “1” | “2” .. | “15” | “16” )

<NoteSpec> = ( [@“Notes=” | “”] (@“ALL” | <NoteRange>)) |

( [@“Note=” | “”] <Note>)

<NoteRange> = ( [<Note>] “~” [<Note>] )| where Note1 <= Note2, Note >= “C1”

(first 12 notes are reserved), Note <” G-10”,

“~” means ALL

<Note> = (“C” | “D” | “E” | “F” | “G” | “A” | “B” ) (“#” | “b” | “”) (“-1” | “0” .. “7” | “8”], check value 1<=note <=127

<ProgramName> = <identifier>

<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> | “\_” )

<status> = (@“enabled” | @“disabled”)

trigger NoteOn MC=1~4 Note=C1~G#5 Transpose5 disabled;

**NoteOff**

will be similar

**Cc Trigger**

trigger <name> Cc <McSpec> <CcSpec> <ProgramName> <status>

<BnkPrgCheck> = ( @”BnkPrg” | “” ) ( <Mc> “.” <Bank> “.” <Prg> )

<Bank> = <digit> 0..127

<Prg> = <digit>0..127

**Translation to MC**

Always one MC

If note range = ALL or ~: create trigger in ALL-notes trigger table

Else if note range: create trigger in OCTAVE\_notes trigger table and specific NOTE trigger table \*

Else if single note: create trigger in single NOTE trigger table

\* example Note range F4~G7 results in 7 single-note triggers for F4 to B4 + 2 octave triggers for octave 5 and 6 and 8 single note triggers for C7-G7, total: 7 + 2 + 8 + 17.

The trigger table is defined by a hash key depending on MC, Note. Note can be:

Single note triggers: 0-127

Octave note triggers: 128-137 (octave 1..10)

All note trigger: 255

**When a note is received**

3 hash keys are calculated:

* All notes trigger table (using also MC)
* Octave trigger table (using also MC, note (octave))
* Single note trigger table (using also MC)

For all three tables, all commands are executed (and checked if the trigger is enabled and condition is met since other triggers can end up in the same trigger table).

**Performance**

To loop through 100 triggers within 3 tables, every trigger taking 10 instructions to check, taking 50 clock cycles. These are 50,00 clock cycle. Assuming 168 MHz this will cost 5,000/168,000,000 = 0.029 ms, thus very less.

The execution of the programs (assume 5 on average high), cost 500 instructions of 4 clock cycli each, resulting in 10,000 clock cycli, which results in 0.059 ms (assuming 168 MHz), also vey less.

**Memory usage**

Assuming there are 256 hash keys / trigger tables, and per entry the following information is stored:

* Type (MIDI/PedSw): 3 bits
* Enabled 1 bit
* For e.g. note on: MC 4 bits
* For e.g. note on: Note 8 bits (for oct)
* Program index 16 bits

Total: 32 bits -> 4 bytes

Thus total storage:

* Table start offsets: 256 \* 2 bytes (start) = 512 bytes
* Tables itself: 1,000 (entries) \* 4 bytes = 4,000 bytes

**Send DMX Instructions**

**Code examples**

send DMX Channel 128 Value 10

Send DMX Channel 128 Var V1

Send DMX Channel 128 Prop NoteNumber // Eg from MIDI

send DMX Channel 128 Values 10 20 30 40 50 60 70 80

send DMX Scenes 0 1 2 3 5 7 Mult 16 Value 10

send DMX OffsetChannel 128 Scenes 0 1 2 3 5 7 Mult 16 Value 10

send DMX OffsetChannel 128 Scenes 0 1 2 3 5 7 Mult 16 Values 10 20 30 40 50 60 70 80

**Code instructions**

0: Single channel

“send” “DMX” <channels\_str> <channel> <values\_str> <value>

1: Multiple channels

“send” “DMX” <channels\_str> <channel> <values\_str> (<value>)+

2: Multiple scenes, single value

“send” “DMX” <offset\_channel\_str> <offset\_channel> <scenes\_str> <scene>+ [<mult\_str> (8 | 16)] <values\_str> <value>

3: Multiple scenes, multiple values

“send” “DMX” <offset\_channel\_str> <offset\_channel> <scenes\_str> <scene>+ [<mult\_str> (8 | 16)] <values\_str> <value>+

<channels\_str> = (“channels” | “ch”), case independent

<channel> = <integer 0..255>

<value\_str> = (“values” | “val” | “v”), case independent

<value> = <integer 0..255>

<scenes\_str> = (“scenes” | “sc”), case independent

<scene> = <integer 0..6>

<mult\_str> = (“mult” | “m”), case independent

<mult> = <”8” | “16”>

<offset\_channel> = (“offset\_channel” | “offset\_ch” | “offch” | “och” | “oc”), case independent

<offset\_channel = <integer 0..2040 or 4080>, multiple of <mult>

**Instruction opcodes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Byte** | **Bits** | **Meaning** | **Values** | **Meaning** | **Description/remarks** |
| 0 | 7~4 | Instruction Type | ???? | Send DMX |  |
| 0 | 3~2 | Sub command | 00 | Single Channel |  |
|  |  |  | 01 | Multiple Channels |  |
|  |  |  | 10 | Multiple scenes, single value |  |
|  |  |  | 11 | Multiple scenes, multiple values |  |
| 0 | 1~0 |  |  |  | Depending on Sub command |

**00: Single Channel**

0: Single channel

“send” “DMX” <channels\_str> <channel> <values\_str> <value>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Byte** | **Bits** | **Meaning** | **Values** | **Meaning** | **Description/remarks** |
| 0 | 1~0 | Value Type | 00 | Value |  |
|  |  |  | 01 | Variable |  |
|  |  |  | 01 | Property |  |
| 1 |  | Channel | 0-255 | Channel |  |
| 2 |  | Value / Property / Variable | 0-255 | Depending on b0.1~0 |  |

**01: Multiple channels**

1: “send” “DMX” <channels\_str> <channel> <values\_str> (<value>)+

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Byte** | **Bits** | **Meaning** | **Values** | **Meaning** | **Description/remarks** |
| 0 | 0~1 | MSB Channel | 00-11 | MSB Channel) | 0, 256, 512 or 1024 |
|  |  |  | 1 | MSB Channel (255-511) |  |
| 1 |  | Channel | 0-255 | Channel |  |
| 2 |  | NrOfValues | 0-255 | Number of values |  |
| 3..NrOfValues |  | Values | 0-255 |  |  |

**10: Multiple channels, single value**

2: “send” “DMX” <offset\_channel\_str> <offset\_channel> <scenes\_str> <scene>+ [<mult\_str> (8 | 16)] <values\_str> <value>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Byte** | **Bits** | **Meaning** | **Values** | **Meaning** | **Description/remarks** |
| 0 | 1~0 | Value Type | 00 | Value |  |
|  |  |  | 01 | Variable |  |
|  |  |  | 01 | Property |  |
| 1 | 7 | Multiplication factor | 0 | 8 |  |
|  |  |  | 1 | 16 |  |
| 1 | 6 | Offset channel or scenes | 0 | Byte 1 are scenes |  |
|  |  |  | 1 | Byte 1 is offset channel |  |
| 1 | 5~0 | If bit 1.6=0: Scenes |  | 6 scenes | (first) 6 scenes |
|  |  | If bit 1.6=1: Offset Channel | 0-255 | Offset channel multiplied by multiplication factor | If mult factor = 8: 0-512  If mult factor = 16: 0-1024 |
| 2 |  | Scenes |  | (LSB) scenes | (last) 8 scenes |
| 3 |  | Value | 0-255 | Value |  |

**11: Multiple channels, multiple values**

3: “send” “DMX” <offset\_channel\_str> <offset\_channel> <scenes\_str> <scene>+ [<mult\_str> (8 | 16)] <values\_str> <value> +

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Byte** | **Bits** | **Meaning** | **Values** | **Meaning** | **Description/remarks** |
| 0 | 1 | Multiplication factor | 0 | 8 |  |
|  |  |  | 1 | 16 |  |
| 0 | 0 | Offset channel or scenes | 0 | Byte 1 are scenes |  |
|  |  |  | 1 | Byte 1 is offset channel |  |
| 1 |  | If 0.0=0: Scenes |  | 8 scenes | (first) 8 scenes |
|  |  | If 0.0=1: Offset Channel | 0-255 | Offset channel multiplied by multiplication factor | If mult factor = 8: 0-2048  If mult factor = 16: 0-4096 |
| 2 |  | Scenes |  | (LSB) scenes | (last) 8 scenes |
| 3 |  | Number of Values | 0-255 | Number of values |  |
| 4..x |  | Values | 0-255 | Values |  |

* + N

**Hardware (Simple)**

Components

* STM32F103C8T6, 512 KB Flash, 20 KB SRAM
* SD Card
* 2x NRF24L01+

Inputs/outputs

* USB (no MIDI)
* SD slot
* 2 x MIDI In
* 1 x MIDI Out
* DMX Out
  + 2 x green: MIDI In
  + 1 x red: MIDI Out
  + 1 x red: DMX Out
  + 1 x blue: power

**Hardware (Max)**

Components

* STM32F446 (?), 1 GB Flash, 192 KB SRAM
* SD card (reader)
* CAN or RS485
* 2x NRF24L01+

Inputs/Outputs

* CAN/RS485
* USB (for MIDI/updates)
* SD slot
* 4x MIDI In
* 2x MIDI Out
* 2x DMX Out
* LEDS:
  + 4 x green: MIDI In
  + 2 x red: MIDI Out
  + 2 x red: DMX Out
  + 1 x blue: power
  + 6 x yellow/red: RF/comm
  + 1 x yellow/red: CAN/RS485

**Setup**

Kronos MIDI OUT -> Mestra MIDI IN 1

Studiologic MIDI OUT -> Mestra MIDI IN 2

Behringer MIDI OUT -> Mestra MIDI IN 3

(KeyTar -> Mestra MIDI IN 4)

Mestra MIDI OUT 1 -> Kronos MIDI IN

Mestra DMX OUT 5 -> DMX Chain

OR

Kronos MIDI OUT -> Mestra MIDI IN 1

Studiologic MIDI OUT -> Behringer MIDI IN

Behringer MIDI OUT -> Mestra MIDI IN 2 (merged with Studiologic)

Mestra MIDI OUT 1 -> Kronos MIDI IN

Mestra DMX OUT 3 -> DMX Chain

Memory for DMX:

128 channels -> 128 bytes

**Messages for DMX:**

Type