Multi-Port MIDI Mapper

A fast hardware-based Midi mapper that controls multiple output Ports.

[TODO]

* Foot peddle automation?
* Multiple output midi commands (like my voice controlled midi app)

## Mappings and Filters

A filter removes certain Midi messages.   
A map changes certain Midi messages.  
A sequence performs stateful orchestration

* Channel Filter  
  Only let messages through that are sent on a specific midi channel (several filter functions).
* Velocity Filter  
  Only let message through that are within a certain velocity value range (key board splits).
* Message Filter  
  Only lets messages through that are of a certain type (channel/real-time/sysex).
* Program Change Map  
  Map the received Program Change to a Program Change value that will be sent to the output.
* Control Change Map  
  Map the received Control Change to a Control Change value that will be sent to the output.
* Note Map  
  Map the received Note (On/Off) message to a Note message that will be sent to the output.
* Velocity Map (advanced)  
  Map the received Note-on velocities to other midi messages. The value of the midi message can be generated by a function based on the incoming velocity.
* SysEx Map  
  Map received midi messages to SysEx message replacing some parameters in the SysEx message.
* Hybrid Map  
  Map a received Midi message (Program Change, Control Change or Note message) to another Midi message that will be sent to the output.
* Trigger Sequence  
  Triggers sending a number of configured Midi messages.
* Step Sequence  
  Send each configured Midi message one at a time based on a trigger input.
* Data Slots  
  Allows a Midi message from anywhere in the chain to be stored for later reuse.
* TODO  
  Indicate if to continue processing after a filter.  
  Allow Maps and Sequences to be turned on and off by Midi input.  
  Power up state of a Map and Sequence.

Data:

Midi Message -> Midi Message  
Midi Message = Program Change | Control Change | Note On/Off | SysEx

Signal flow processing:

[Filter] -> [Detect] -> [Transform] -> [Dispatch]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filter** | **Detect** | **Transform** | **Dispatch** |
| Program Change | [Type]  Bank Select # (hi/lo) Program Change # | Bank Select # (hi/lo) Program Change # | Param value functions | Bank Select # (hi/lo) Program Change # |
| Control Change | [Type]  Control # (hi/lo) | Control # (hi/lo) | Param value functions | Control # (hi/lo) |
| Note On/Off | [Type] Channel Velocity Note # | Note # | Param value functions | Channel Velocity Note # |
| SysEx | [Type] | Templated | Param value functions | Templated |

### Filtering

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Data** |
| Message Type | Filters on midi message types. | Note | Controller | Program Change | Pitch Wheel | Real Time | Common | SysEx |
| Channel | Filters on Midi Channel numbers | 1-16 (multiple) |
| Note # Range | Filters on Note Number ranges | 1-128 (begin-end) |
| Value range | Filters on Note Velocity, Control Change and Program Change values | 1-128 (begin-end) |
|  |  |  |

An incoming midi message must pass all filters before it is allowed to continue. Each type of filter can also indicate if the filtered messages (that are not mapped) should be send to the Midi output unchanged. Those pass-through messages will never be delivered to the Detect stage. When no filters are defined, all midi messages are passed through to the Detect stage.

### Detection

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Data** |
| Note On/Off | Detect Note messages. |  |
| - Note # |  | Specific, Range or Any |
| - Velocity |  | Specific, Range or Any |
| Control Change | Detect Control Changes |  |
| - Control # |  | Specific, Range or Any |
| - Value |  | Specific, Range or Any |
| Program Change | Detect Program Changes |  |
| - Program # |  | Specific, Range or Any |
| - Value |  | Specific, Range or Any |
| Pitch Wheel | Detect Pitch bends |  |
| SysEx | Detect SysEx based on a template |  |
| Common | Detect system common messages | Specific, Range or Any |
| Realtime | Detect System Realtime messages | Specific, Range or Any |

### Transformation

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Data** |
| Literal Value | Value is passed without changing it | 0-127 |
| Value Map | Maps a single value to another value | 0-127 |
| Linear Map | Maps a value range to another value range in a linear way | 0-127 |
| Log Map | Maps a value range to another value range in a logarithmic way | 0-127 |
| [other range maps] | Maps a value range to another value range in “a specific” way | 0-127 |
| Constant Value | Always provides the same value | 0-127 |
| Add Value | Adds a constant to the value | clipped |
| Sub Value | Subtracts a constant from a value | clipped |
|  |  |  |

Transformation functions can be parameterized. These parameters can also be sourced from other information in the same Midi message. Multiple transformations can be defined for one outgoing midi message.

### Dispatcher

The dispatcher builds up a new midi message using the output(s) from the transformation step. The possible message builders are the opposite of the message detectors including parameters.

So for instance a note message can be constructed using a note # and a velocity.

For each midi message type a specific ‘builder’ can be used to make sure all parts of the midi message are provided. These parts can be supplied by transformation functions. Multiple builders –and therefor multiple output midi message- can be used.

## Data Structure

A pool of Map definitions is maintained by the device. Each Patch assigns a Map from the pool to the outputs that are available. If no map is assigned, no output will be generated.

System properties

|  |  |  |
| --- | --- | --- |
| **Property** | **Values** | **Description** |
| SysEx-Recv | Yes|No | Allow the unit to receive Midi SysEx messages. |
| SysEx-Id | 0-127 | A Midi SysEx Device Id. |
| SysExCh | 1-16 | The Midi Channel the SysEx messages are received on. |

## Schematic

The device will initially start out with 1 Midi input 4 Midi outputs.

## Midi Implementation

The Midi implementation for the device itself (not the mapper) is displayed in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Midi Message** | **Transmitted** | **Received** | **Remarks** |
| Note on/off | No | No | Ignored |
| Control Change | No | No | Ignored |
| Program Change | Yes | Yes | Patch select. Can be disabled. |
| Real Time | No | No | Ignored |
| SysEx | Yes | Yes | Bulk import/export. |

## Bill Of Materials

|  |  |  |
| --- | --- | --- |
| **#** | **Name** | **Link** |
| 6 | Midi Socket | <http://nl.farnell.com/pro-signal/psg03463/din-socket-pcb-5p-180deg/dp/1791756> |
| 1 | CNY17 Opto coupler | <http://nl.farnell.com/isocom/cny17f-4x/optocoupler-dip-6-tr-o-p/dp/1683185> |
| ~~1~~ | ~~LCD 2x20 Display~~ | [~~http://nl.farnell.com/powertip/pc2002lrs-awa-b-q/lcd-module-20x2-led-b-l/dp/1671504~~](http://nl.farnell.com/powertip/pc2002lrs-awa-b-q/lcd-module-20x2-led-b-l/dp/1671504) |
| 1 | LCD 2x24 Display | <http://nl.farnell.com/midas/mc22405c6wk-sptly/lcd-2x24-stn-ylw-green-b-l-5mm/dp/2063254> |
| 2 | 74LS07N | <http://nl.farnell.com/texas-instruments/sn74ls07n/ic-buffer-driver-receiver/dp/1470756> |
| 2 | 74HC595N | <http://nl.farnell.com/nxp/74hc595n/ic-74hc-cmos-74hc595-dip16-5v/dp/3166028> |
| 12 | 1N4148 | <http://nl.farnell.com/multicomp/1n4148/diode-do-35-100v-150ma/dp/9565124> |
| 18 | 220 ohm | <http://nl.farnell.com/multicomp/mcf-0-25w-220r/resistor-220r-0-25w-5/dp/9339299> |
| 1 | Trimmer 10k | <http://nl.farnell.com/te-connectivity-citec/cb10lh103m/trimmer-side-adjust-10k/dp/1227524> |
| 4 | 100nF capacitor | <http://nl.farnell.com/vishay/a104k15x7rh5taav/capacitor-axial-100v-100nf/dp/1902257> |
| 1 | 24LC256 I2C EEPROM | <http://nl.farnell.com/microchip/24lc256-i-p/eeprom-serial-256k-24lc256-dip8/dp/9757970> |

## Resources

### USB – MIDI

Roland Serial to MIDI Driver

<http://www.usb.org/developers/devclass_docs/midi10.pdf>

<http://forum.arduino.cc/index.php?topic=22047.0>