# Software Workflow

## Introduction

This chapter describes the flow from the user-written configuration file until the execution of commands by the controller.

## Configuration File

The user writes a configuration file. As an example the next file is used. It will transpose all MIDI notes on MIDI channel 1 from notes C4 and higher, 5 semitones up.

Trigger TransposeC4Plus5 MC 1 C4~ TransposePlus5

Program TransposePlus5

Var Note = NoteNr

Add Note 5

Property NoteNr Note

Send

## Translation to Mestra File

To translate the configuration file to a Mestra file, the configuration file is parsed.

### Trigger translation

Each trigger in the configuration file can result in many triggers in the Mestra file. In principle every note has its own rule, so the rule would be copied for every note in the trigger (C4 and up). To reduce this amount, each group of 8 notes has its own list. Note C4 has note number 60. This means it will be added to the following note groups (mentioned ‘yes’)

All notes notes 0..127 No

Note Group 1: notes 0 to 7 No

Note Group 2..7

Note Group 8 Notes 56 to 63 No

Note Group 9 Notes 64 to 71 YES

Note Group 10..15

Note Group 16 Notes 120..127 YES

Note 17 Note 17 No

Note 59 Note 59 No

Note 60, 61, 62, 63 Note 60..63 Yes

Note 64, … Note 64..127 No (handled by note groups 9..16).

Also, notes 60, 61, 62 and 63 are missing, so the trigger also needs to be added in these notes.

### Hash Keys

There can be many trigger for each note and note groups (and per MIDI channel). Therefore, hash keys will be used to minimize the number of tables needed for each MIDI note/group/MC. Since hash keys do not have to be unique for different trigger sources (there can be multiple different MIDI notes within one hash key group), the trigger properties need to be stored and rechecked during execution.

The size of the hash table (number of hash keys) is typically 256.

The formula to translate a trigger to a hash key is.

Hash\_key = based on depending on MC, Note (group) or other MIDI message type

Note (group): The lowest notes values are used for the groups (0 means all), 1-16 for note groups 1-16, notes 17.. for notes.

Other MIDI message type: e.g. pitch bend, after touch do not have a note number and will be treated differently for calculating the hash key. See table below.

|  |  |  |
| --- | --- | --- |
| **MIDI Type (4 bits)** | **Byte 1 for hash key** | **Byte 2 for hash key** |
| Note On/Off | MIDI channel | Note number\* |
| Polyphonic Key Pressure (aftertouch) | MIDI channel | Note number\* |
| Control Change | MIDI channel | Control number |
| Program Change | MIDI channel | Patch number |
| Channel Pressure / aftertouch | MIDI channel |  |
| Pitch Bend Change | MIDI channel |  |
| Channel Mode Messages | Control numbers (120-127) |  |
| System Common/RT Messages/Sysex | LSB 4 bits |  |

So 4 bits (MIDI Type) + Byte 1 and optionally byte 2 are used to generate the hash key using a simple XOR.

Each (MIDI) trigger has the following format in the Mestra file:

* Byte 0:
  + MIDI Type (3 bits)
  + 1 bit: Enabled/disabled
  + 4 bits reserved
* Byte 1
  + 1 bit for all/note groups
  + 7 bits 1 for hash key (above)
* Byte 2
  + 1 bit reserved
  + 7 bits 2 for hash key (above)
* Byte 3
  + MSB Program ID (together with 5 bits above): 14 bits -> 16,384 programs