
TiSM
Sequencing Machine

Manual

Samuel DUFOUR-KOWALKI ¡dufourko@free.fr¿

October 5, 2004

Website: http://tism.sourceforge.net

1 General

1.1 Presentation

Computer assisted music cannot replace human interpretation during a performance. At the time the project started, there was no music software adapted to human performance. We needed a tool which allows to interact at a higher level than the sound directly (like sound effect) and which help to ”control the time” and allow to record, modify, repeat, stretch (...) musician actions. Our solution was a real-time controlable and flexible Midi sequencer.

TiSM is an interactive & programmable software which allows to transform, record and sequence generic Midi messages in **Real - time**. The main idea is to provide a system that behave like a hard sequencing machine more than a traditionnal computer sequencer software. TiSM allows to program the system behaviour by associating simple Tcl scripts with received messages (Midi keyboard, controller, computer keyboard) or sequenced messages, so the tool can be used for a lot of different real-time applications.

The tool is fully customisable in order to answer to the different user needs. TISM is based on a TCL interpreter and each system functionnality is accesible with a special TCL function. The users can write TCL *scripts which are executed at run-time*. This allows to control the sequencer in response to external events (MIDI, keyboard...) or scheduled events (from the sequencer itself).

TiSM is not a GUI based software.
A simple GTK interface can help users to access more directly to the main functionalities, but TiSM can be fully exploited only with Tcl scripts.

Main features:

- Sequence management (play, record, time properties, time streching, quantization, synchronisation...)
- Real-Time Message transformation.
- Real-Time Control
- Fully customisable and programmable.
- Generic I/O (Alsa & OSS MIDI Input/Output , Keyboard Input,...).

1.2 Starting the application

TiSM must be started in a console window, by typing the command :

```
>tism
```

You can start TiSM with diffrent options:

- —help : display options list
- —init *filename* : start with the specified tcl script (default is ~/.tismrc).
- —nodisplay : disable interface
- —sched : set real-time priority (recommanded). Need root privileges.
- —memlock : disable virtual memory. Need root privileges.
- —version : display version number and quit.

When TiSM is started, there is no default configuration. You must configure the input/output and how to control the system.
For that, you must either write a Tcl script and load it in the system (with the command line option or in the GUI), or use directly the GUI functionnalities.

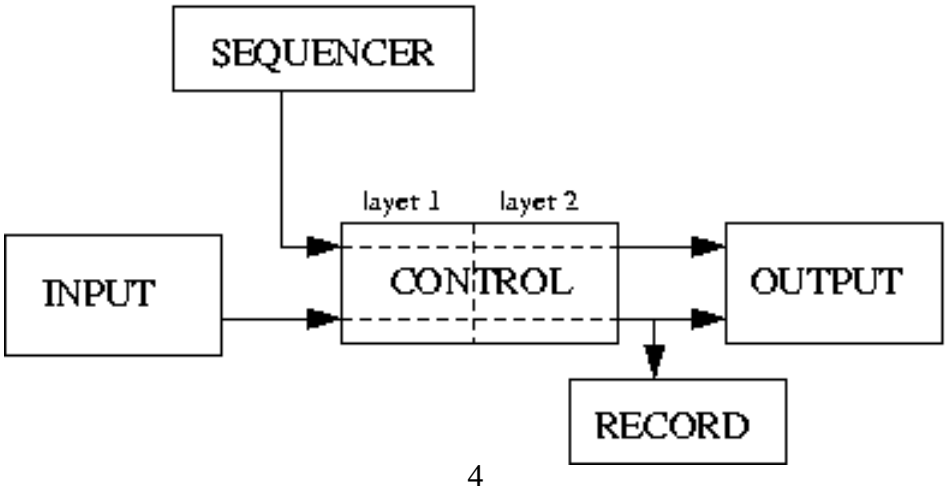
1.3 System structure

To use TiSM correctly, it is better to understand how the system work.

TiSM is a general message processing system. It is not limited to MIDI messages processing, but can handle any kind of list of 8-bits values. MIDI messages are a sub-set of this kind of messages.

TiSM is composed by different subsystems:

- Input/Output subsystem : communication with the outside world. It can support different message protocol as MIDI. See the IO section 2.
- Control subsystem : The tcl interpreter. It executes the scripts associated with received messages. There are two control system layer. See the Control section 4.
- Sequencer subsystem : it manages messages sequences playing and recording. See the Sequencer section 3.



1.4 Messages Format

TiSM manipulates generic messages.
A message is a list of value, each value is 1 byte length (a value is between 0 and 255). The message length (the number of values in the lsit) is unlimited.

The first message value corresponds to the I/O port. It is used by TiSM to route message to the correct output and is assigned depending the input the message has been received (see Input/Output subsystem). The other values haven't special meaning for tism.

MIDI message are represented in thsi format. For instance a MIDI Note ON message on channel 0 can be : 0 144 0 34 123
The used format is :

- Value 0 : TiSM port
- Value 1 : Midi status byte
- Value 2 : Midi channal byte
- Value 3 : Midi Data 1
- Value 4 : Midi Data 2

This structure can change depending of the status byte. In order to use MIDI in TISM, you need to know the MIDI protocol.

By this way, TISM can support standard MIDI messages, but also system exclusive messages, and all others kind of messages which fit with the TISM message format.

2 Input/Output

2.1 Protocols

TISM is able to have diffrent kind of inputs and outputs (ex Midi, keyboard, network...).
A I/O protocol defines how and where the messages are read and written, from and to the outside world.
Each protocol needs specials parameters (ex: device file).

TISM supports different protocols :

- **Input protocols**
  - MIDI standard (OSS) : protocol = midioss, param = midi device file (ex : /dev/midi)
  - MIDI standard (ALSA) : protocol = alsamidiraw, param = alsa midi device file (ex : hw:0,0)
  - KEY (for the keyboard) : protocol = key, param = keyboard device file (ex : /dev/tty)
- **Output protocols**
  - MIDI (OSS) : protocol = midioss, param = midi device file (ex : /dev/midi)
  - MIDI (ALSA) : protocol = alsamidiraw, param = alsa midi device file (ex : hw:0,0)

2.2 I/O Port

TISM can manage 16 inputs and 16 outputs. Each of them is defined by a port number (0-15).

A protocol and its parameters are associated to each port.

When a message is receive on a port, the first value is set to the input port number.
In the same way, the first value of an outgoing message defines the port on which the message is going to be emitted. If a message format doesn't fit with the protocol, it's discarded.

2.3 IO Tcl functions

The I/O Tcl functions are listed in the Tcl Reference (see Section6 for more information)

3 Sequencer

3.1 Sequence Properties

The system is able to manage sequences. A sequence can be considered like a messages list with time information.

In a sequence, the base time is expressed in **SECOND**. This time is converted later with play parameters. When editing a sequence, a message date can be noted *second:u_second* or with a fraction *second/precision*. In the later, *second* is divided by *precision*

The sequencer has a global tempo expressed in BPM. Furthermore a sequence has additional properties:

- **ID**
- **Begin Marker**
- **End Marker**
- **Play time length**
- **Quantization**
- **Quantization Mode**
- **Default synchronisation**
- **Ouput port**
- **Mute state**
- **Loop mode**

The unique **ID** is used to identified a sequence in a controller script.

The **Begin Marker** and the **End Marker** are used to define a selection in a sequence.
When playing the sequence, only the selection will be read.
The markers can be considered like special messages in the sequence. They have a time position like other message.

The **Play time length** is play parameter. It defines the effective playing time.
It is expressed by a fraction *len/precision* which corresponds to musical measure.
For instance, 4/4 corresponds to 4 beats.
This parameter is relative to the global tempo and can be use to do *time streching*.
If len is null, the sequence is played without stretching.

The **Quantization** parameter is expressed in the same way (len/precision). It allows time rounding at the precision specified.

The **Quantization Mode** defines how the messages are quantized.

The **Default synchronisation** parameter is expressed also in the same way (len/precision). It specifiies the default playback synchronisation.

The **Ouput port** defines the first value for all messages in the sequence. Output port is not assigned if its value is -1.

The **Mute state** defines if the sequence is muted or not.

The **Loop mode** defines if the playback is done in loop or not.

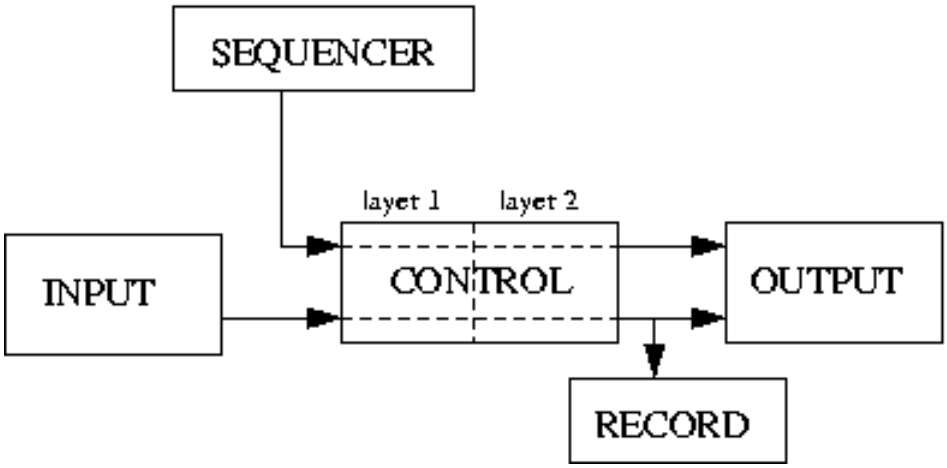
3.2 Play/Stop

A sequence is monophonic : a sequence cannot be played twice at the same time.

The playback start time can be synchronised.
This synchronisation is determined by a fraction *len/precision* which defines the number of beat to wait before the playback starts.
For instance, if the sync is 4/4, the playback will start in the next 4 beats.

3.3 Record

TISM can record every received messages.
The messages are recorded AFTER the control step.



The record process is done in 2 steps.

1. When record is started, TISM put somewhere all received messages.
2. The recorded messages will be put in the choosen sequence when the record process is stopped.
   The record messages can be merged in the sequence or the sequence can be overwritten with the new messages.
   The record process can be canceled without loosing data.

3.4 Sequencer Tcl functions

The sequencer Tcl functions are listed in the Tcl Reference (see Section6 for more information)

4 Control

4.1 Controller

The system beahviour is determined by controllers. they are used to:

- Transform messages in real-time
- Control the system.

Controller are applied in response of particular input messages.
In other words, it is possible to do special computations when the user activate an external device (ex: a computer keyboard, a MIDI keyboard...).

A controller is composed by:

- **an ID** : This unique number is used to identified the controller in script.
- **a MASK** : it is the **condition** required to apply the controller
- **2 TCL Scripts**: they are both executed when the condition is satisfied. The boths script have different **execution modes**, on-the fly or in parallel.

The script must be written in TCL.
All funtionalities of TCL 8.3 can be used.
Furthermore, special functions have been added to the language, in order to access the TISM functionalities like sequence managing, control etc.
These special TCL funtions are listed in the Tcl Reference (see Section6).

4.2 Controller Mask

The controller condition corresponds to a standard message called the **mask**.
When a message is received, it is compared to the set of declared masks.
The system retains the controller which has the closest mask to the receive message.

For instance, the following controller have been declared :

- Mask 1 : 0
- Mask 2 : 0 100
- Mask 3 : 0 100 10

The controller selection is done like this :

- 0 100 -¿ Mask 2
- 0 100 10 -¿ Mask 3
- 0 100 11 -¿ Mask 2
- 0 102 -¿ Mask 1
- 0 102 10 -¿ Mask 1

4.3 Groups

In order to have different system behaviours easily, a controller belongs to a group.

Two controllers can have the same Mask only if they are not in the same group.

Controller groups are used to define a subset of controllers to use. When the system is running, there are only TWO active groups. The others are desactivated. This means that there are only the scripts which belongs to controllers in the active groups which are applied.

When a message is processed by the control subsystem, it will first be processed by the first controller group and then by the second group.

4.4 Execution Modes

For each mask in a group, TISM supports 2 TCL scripts.

Script 1 - On the fly execution
This script is executed on the fly, before the message is sent.
It can be used to transform message.
It is highly recommended to avoid to use this kind of script for heavy computation.

Script 2 - Parallel execution:
This script is executed in a separate thread.
The time execution is not known, but it doesn't block the system. Heavy computation must be done in this script.

4.5 Use of Tcl Scripts

The most of TISM functionalities can be accessed with specials Tcl functions.
The controller scripts are executed at Run-time and with very *precise timing*. They are called in response to external events or sequencer events.
It is possible to use internal messages just for control purpose. Indeed, if the first message value is greater than 16, it will never be send to the outside world.
This kind of messages can be use to control the sequencer or a recording process : a particular sequence can send internal messages at precise time in order to execute control functions.

Note : Using Tcl variable in a "script 2 - parallel execution"
Because the script 2 of a controller is executed in a separate thread, standard tcl variable cannot be used to save data between different script executions.
Instead, TISM has its own variable manager which must be used in this case.

Theses function are listed in the Tcl Reference (see Section6 for more information)

4.6 Control Tcl functions

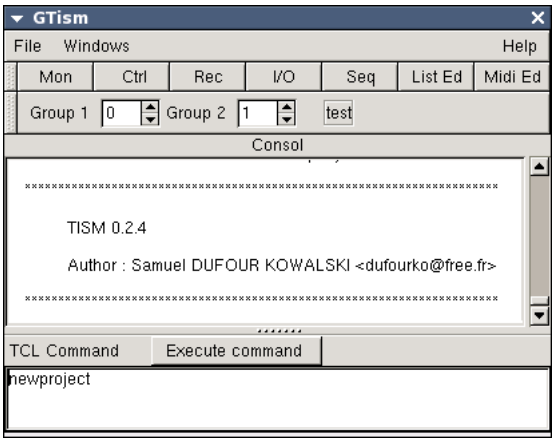
The control Tcl functions are listed in the Tcl Reference (see Section6 for more information)

4.7 Others Tcl functions

Some specials Tcl functions are necessary to access and transform messages. Theses functions are listed in the Tcl Reference (see Section6 for more information)

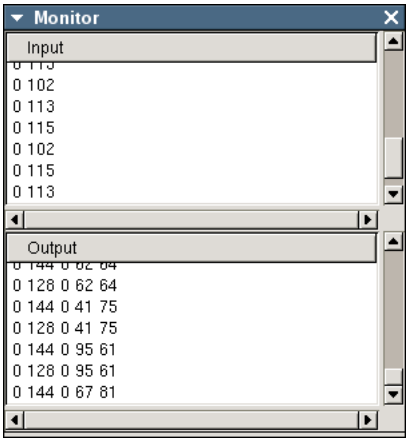
5 GTiSM Graphical interface

5.1 Main Window



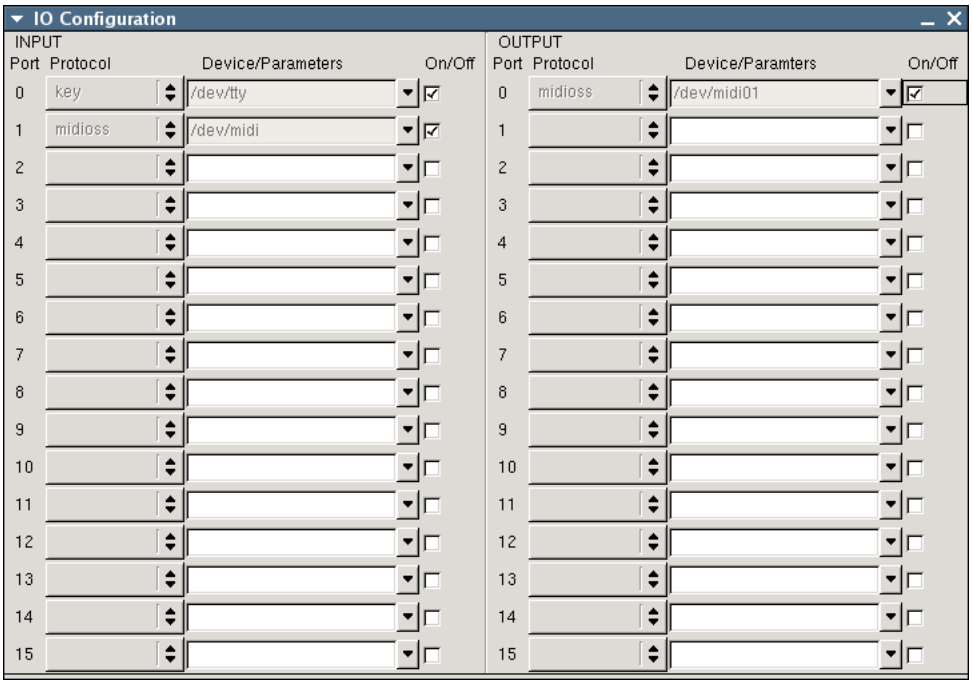
- The 'Group 1' spin button can be used to change the first layer active controller group.
- The 'Group 2' spin button can be used to change the second layer active controller group.
- On the console are displayed status messages and errors.
- Tcl Command block allows to execute directly Tcl commands. The text doesn't disappear after clicking 'execute' when the tcl code is erroneous.

5.2 Monitor window



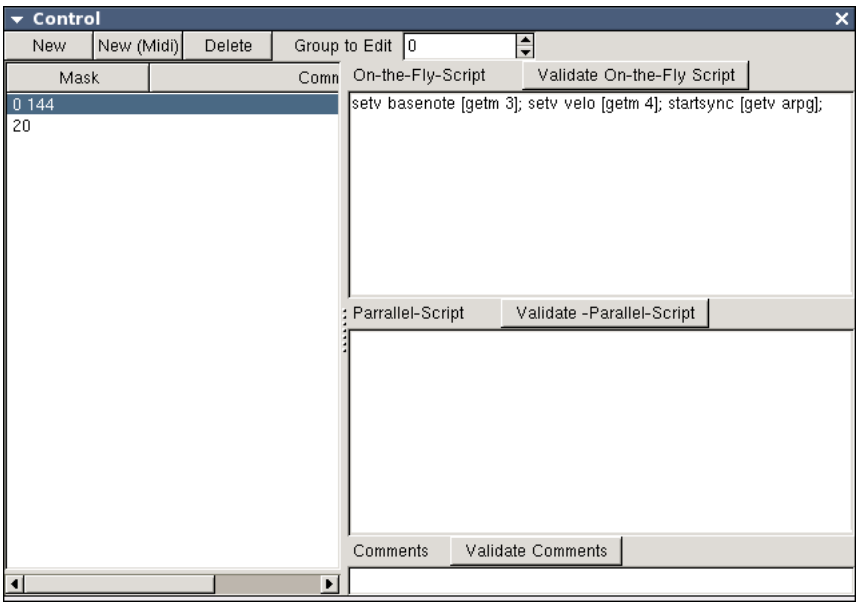
The monitor displays input and output messages.

5.3 IO configuration window



The I/O configuration can be used to set the protocol and its parameters for each port.

5.4 Control configuration window



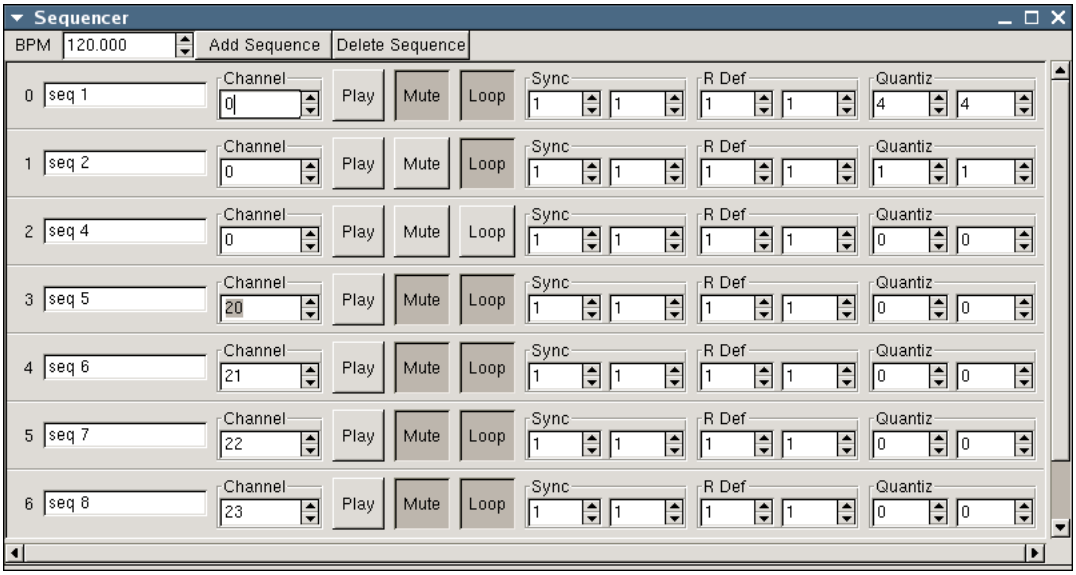
The controller configuration window can be used to set the controller for each group.
Only controllers which belong to ’GROUP’ are listed on the left screen part.

To edit a controller, it must be selected in the list.

After having changed a field (script1, script2 or comment), it must be validated by clicking on the appropriate button.

The ’add’ button can be used to create a new controller. The controller mask must be provided at this time, and will not be editable anymore. In order to change a controller mask, the controller must be deleted and re-created with the correct mask.

5.5 Sequencer window

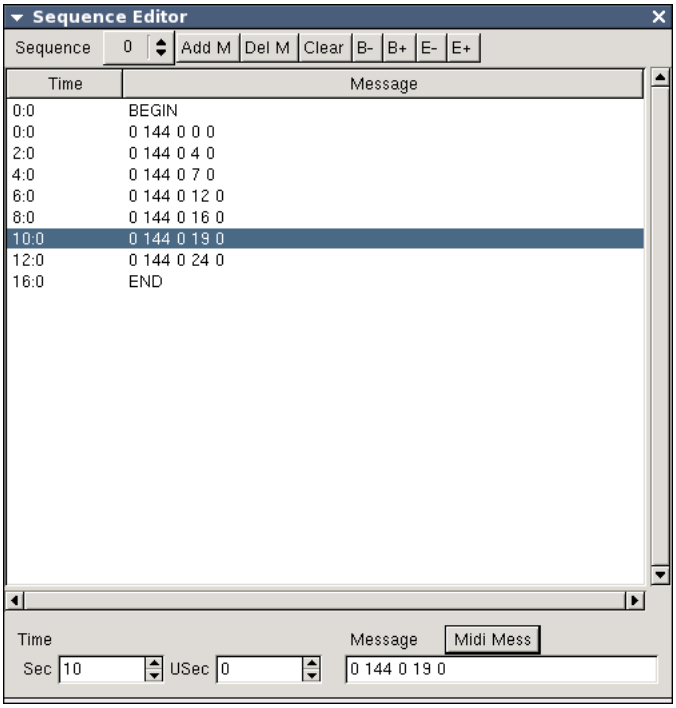


The sequencer window can be used to manage sequences.
The ’add button’ allows to create a new sequence.
The ’delete button’ allows to delete the *last created* sequence.
The main sequence properties are editable :

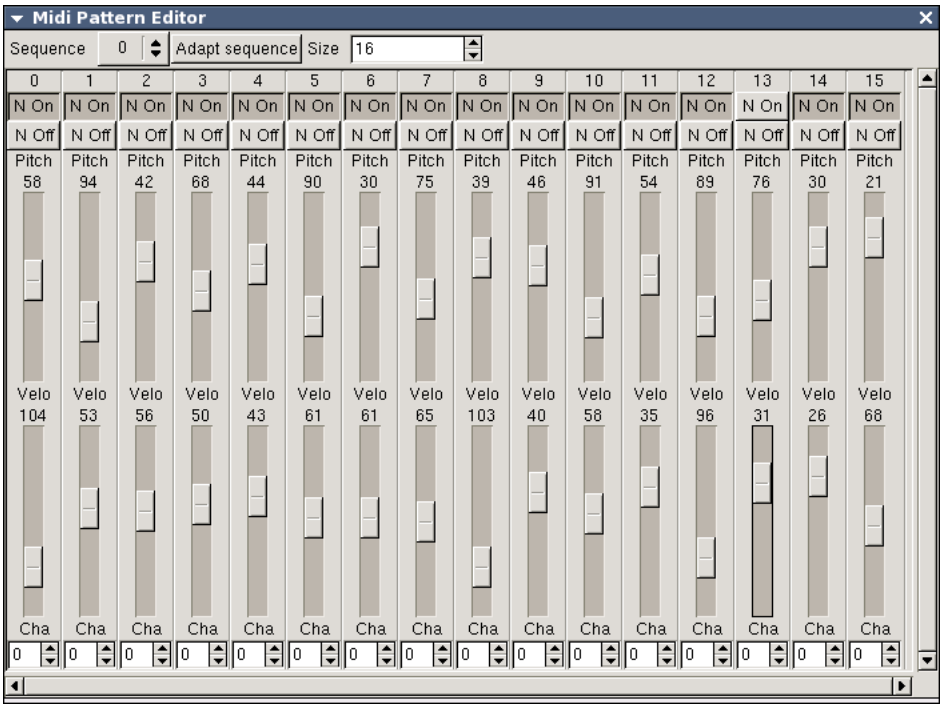
- Output channel,
- Mute
- Loop mode
- Time stretching (R Def)
- Quantization
- Default Synchronisation

The ’play’ button allows to start the sequence with the default synchronisation parameters.

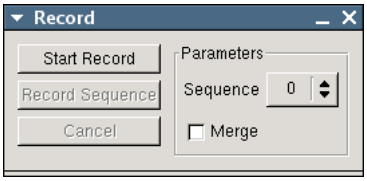
5.6 List editor



5.7 Pattern editor



5.8 Record window



6 TCL Functions reference

IO

```
TISM TCL REFERENCE IO
   * addinprotocol port protocol param : configure input port
   * addoutprotocol port protocol pram : configure output port
   * deleteinprotocol port : desactivate input port
   * deleteoutprotocol port : desactive output port
   * deleteallprotocol : desactivate all
```

CTRL

```
TISM TCL REFERENCE CTRL
   * createctrl group mask : create a new controller (return ctrlid) in a group
   * setscript1 group ctrlid script : edit On-th-Fly-Script of a controller in a group
   * setscript2 group ctrlid script : edit Parallel-Script  of a controller in a group
   * setcomment group ctrlid string : edit comments of a controller in a group
   * deletectrl group num : delete a controller in a group
   * deletegroupctrl group : delete all controller in group
   * getctrl group mask: get the controller id in a group associated with a mask
   * getcgroup : return the first current group number
   * setcgroup group : set the first current group
   * getcgroup2 : return the second current group number
   * setcgroup2 group : set the second current group
```

SEQUENCE

```
TISM TCL REFERENCE SEQUENCE
    * createseq name : create a new sequence (return seq id)
    * nameseq seqid name : set sequence name
    * seqchanout seqid output_channel : set sequence output channel
    * seqrdef seqid len precision : set sequence playing time
    * quantize seqid len precision : set sequence quantization value
    * quantizetype seqid typename : set sequence quantization type (default, midi)
    * seqdefaultsync seqid len precision : set sequence default sync value
    * seqmute seqid status : mute a sequence (status = 0)
    * seqtogglemute seqid : mute state inversion
    * seqloop seqid status : set sequence loopmode (status!=0 -> loop on)
    * seqtoggleloop seqid : loopmode inversion

    * beginmark seqid sec precision : set sequence begin mark
    * endmark seqid sec precision : set sequence end mark
    * beginmarksec seqid sec usec : set sequence begin mark
    * endmarksec seqid sec usec : set sequence end mark
    * beginmarknext seqid : Move begin mark to the next message
```

```
    * beginmarkprev seqid : Move begin mark to the previous message
    * endmarknext seqid : Move end mark to the next message
    * endmarkprev seqid : Move end mark to the previous message

    * addevt seqid sec precision message : add a message in a sequence
    * addevtsec seqid sec usec message : add a message in a sequence
    * deleteseq seqid : delete a sequence
    * clearseq seqid : clear all messages in a sequence
    * copyseq seqid1 seqid2 : copy seqid1 to seqid2

    * startsync seqid [numbeat] [precision] : start playback
    * stopsync seqid [numbeat] [precision]) : stop playback
    * toggleplaysync seqid [numbeat] [precision] : inverse play status

    * recbegin : start record
    * recseq seqid : put record buffer in a sequence
    * recseqmerge seqid : merge record buffer in a sequence
    * recstop : cancel record

    * syncemit len precision message : send a message
    * setbpm bpm : set the tempo
```

UTIL

```
TISM TCL REFERENCE UTIL
    * getm pos : get the pos'th message value
    * setm pos value : set the pos'th message value
    * resizem size : resize message
    * getv varname : get variable value
    * setv varname value : set variable value
    * quit : quit TISM
    * exec_file filename : execute a Tcl script file
    * newproject : new project
    * loadproject filename : load project
    * saveproject filename : save project
    * saveallseq filename : save all sequences
    * saveseq id filename : save sequence id
```

DISPLAY

```
TISM TCL REFERENCE DISPLAY
    * printtitle str : display screen in the title frame
    * print str : display screen in the console
```

7 TCL script exemples

7.1 A simple MIDI channel converter

```
############ TISM TUTORIAL 1 ##########
######  Simple MIDI Channel converter


###### Input / Output #################

# Alsa MIDI Input (device 0)
# On Tism port 0 (All messages received will begin by 0)

addinprotocol 0 alsamidiraw hw:0,0

# Computer Keyboard Input on Port 1 (messages begin with 1)

# Note : The Terminal window where Tism has been lauch
# must be FOCUSED to capture keyboard event

addinprotocol 1 key /dev/tty
printtitle test

# Alsa Midi Output (device 0)
# On Tism Port 0 (All messages which begin by 0 will sent)

addoutprotocol 0 alsamidiraw hw:0,0

###### Control ######################

# Create scripts to transform channel number of incoming Midi message
# We assign 1 script in 8 group corresponding to 8 midi channel
# note : we reserve the group 0 for later
# The script consist of changing the third value of message
# The script is applied to every message received on port 0 (mask : 0)

#GROUP 1 -> Midi Channel 0

set a [createctrl 1 0]
setscript1 1 $a "setm 2 0"

#GROUP 2 -> Midi Channel 1

set a [createctrl 2 0]
setscript1 2 $a "setm 2 1"
```

```
#GROUP 3 -> Midi Channel 2

set a [createctrl 3 0]
setscript1 3 $a "setm 2 2"

#GROUP 4 -> Midi Channel 3

set a [createctrl 4 0]
setscript1 4 $a "setm 2 3"

#GROUP 5 -> Midi Channel 4

set a [createctrl 5 0]
setscript1 5 $a "setm 2 4"

#GROUP 6 -> Midi Channel 5

set a [createctrl 6 0]
setscript1 6 $a "setm 2 5"

#GROUP 7 -> Midi Channel 6

set a [createctrl 7 0]
setscript1 7 $a "setm 2 6"

#GROUP 8 -> Midi Channel 7

set a [createctrl 8 0]
setscript1 8 $a "setm 2 7"


### Assign to keyboard key script to change 2nd active group
### Changing active group will allow to change midi channel
# numeric key 0 (mask: 1 48)

set a [createctrl 0 1 48]
setscript1 0 $a "setcgroup2 1"

# numeric key 1 (mask: 1 49)

set a [createctrl 0 1 49]
setscript1 0 $a "setcgroup2 2"

# numeric key 2 (mask: 1 50)
```

```
set a [createctrl 0 1 50]
setscript1 0 $a "setcgroup2 3"

# numeric key 3 (mask: 1 51)

set a [createctrl 0 1 51]
setscript1 0 $a "setcgroup2 4"

# numeric key 4 (mask: 1 52)

set a [createctrl 0 1 52]
setscript1 0 $a "setcgroup2 5"

# numeric key 5 (mask: 1 53)

set a [createctrl 0 1 53]
setscript1 0 $a "setcgroup2 6"

# numeric key 6 (mask: 1 54)

set a [createctrl 0 1 54]
setscript1 0 $a "setcgroup2 7"

# numeric key 7 (mask: 1 55)

set a [createctrl 0 1 55]
setscript1 0 $a "setcgroup2 8"

################ Default value ######
# first active group : group 0 (key)
setcgroup 0

# first active group : group 1 (Midi channel set to 0)
setcgroup2 1
```

7.2 A simple MIDI arpegiator

```
############ TISM TUTORIAL 2 ##########
######  Simple MIDI Arpegiator


###### Input / Output #################

# Alsa MIDI Input (device 0)
```

```
# On Tism port 0 (All messages received will begin by 0)

addinprotocol 0 alsamidiraw hw:0,0

# Computer Keyboard Input on Port 1 (messages begin with 1)

# Note : The Terminal window where Tism has been lauch
# must be FOCUSED to capture keyboard event

addinprotocol 1 key /dev/tty
printtitle test

# Alsa Midi Output (device 0)
# On Tism Port 0 (All messages which begin by 0 will sent)

addoutprotocol 0 alsamidiraw hw:0,0

###### Sequences ############################

setbpm 160

# The arpegiator pattern
set tmp [createseq "arpegiator1" ]

#pattern parameters

# output port 20
seqchanout $tmp 20
# not muted
seqmute $tmp 1
# loop on
seqloop $tmp 1
# quantization 1/16
quantize $tmp 1 16
# pattern length 1/1 (4 beats)
seqrdef $tmp 1 1
# no default sync
seqdefaultsync $tmp 0 0
# begin & end marks
beginmarksec $tmp 0 0
endmarksec $tmp 16 0

#pattern messages

addevtsec $tmp 0 0 0 144 0 0 0
```

```
addevtsec $tmp 2 0 0 144 0 4 0
addevtsec $tmp 4 0 0 144 0 7 0
addevtsec $tmp 6 0 0 144 0 12 0
addevtsec $tmp 8 0 0 144 0 16 0
addevtsec $tmp 10 0 0 144 0 19 0
addevtsec $tmp 12 0 0 144 0 24 0

#save seq id
setv arpg $tmp

################## Control ######################


# What to do when receive Note On message on port 0
# mask : 0 144
# We save the note number and velocity to transpose the pattern
# and we start the sequence

set a [createctrl 0 0 144]
setscript1 0 $a {setv basenote [getm 3]; setv velo [getm 4]; startsync [getv arpg];}

#define controller to transpose pattern (on port 20)
# redirect to port 0 (midi output), change pitch and velocity

set a [createctrl 0 20]
setscript1 0 $a { setm 0 0; setm 3 [expr [getm 3] + [getv basenote]]; setm 4 [getv velo]}


################ Default value ######
# first active group
setcgroup 0
# second active group
setcgroup2 1
```

7.3 A simple sequence recorder

```
############ TISM TUTORIAL 3 ##########
######  A midi recorder ##############

# This tutorial show how to control the sequencer
# we use 13 sequences, each sequence is assigned to a midi note (starting at the note 60)
# There are different mode : start mode, mute mode, loop mode, copy mode, Null mode
# START MODE : when the midi key is pressed, the assigned sequence is started or stopped
# MUTE MODE : when the midi key is pressed, the assigned sequence is muted or unmuted
```

```
# LOOP MODE : when the midi key is pressed, it change the assigned sequence loop status.
# COPY MODE : when the midi key is pressed, it replace the sequence by the previous recor
# NULL MODE : do nothing

# the computer keyboard is used to change the mode. on French keyboard :
# START MODE : w
# MUTE MODE : x
# LOOP MODE : c
# COPY MODE : v
# NULL MODE : <

# More over the space bar is used to record.

###### Input / Output #################

# Alsa MIDI Input (device 1)
# On Tism port 0 (All messages received will begin by 0)

addinprotocol 0 alsamidiraw hw:1,0

# Computer Keyboard Input on Port 1 (messages begin with 1)

# Note : The Terminal window where Tism has been lauch
# must be FOCUSED to capture keyboard event

addinprotocol 1 key /dev/tty
printtitle test

# Alsa Midi Output (device 1)
# On Tism Port 0 (All messages which begin by 0 will sent)

addoutprotocol 0 alsamidiraw hw:1,0

###### Sequences ###################

#temporary sequence to store record message
set tmp [createseq "recseq" ]

#pattern parameters
seqchanout $tmp -1
seqmute $tmp 1
seqloop $tmp 1
quantize $tmp 1 32
seqrdef $tmp 1 1
seqdefaultsync $tmp 1 1
```

```
beginmarksec $tmp 0 0
endmarksec $tmp 16 0

#create 13 sequences
#and its controllers
for {set i 0} {$i<13} {incr i} {

set tmp [createseq "seq" ];

#pattern parameters
seqchanout $tmp -1;
seqmute $tmp 1;
seqloop $tmp 1;
quantize $tmp 1 32;
seqrdef $tmp 1 1;
seqdefaultsync $tmp 1 1;
beginmarksec $tmp 0 0;
endmarksec $tmp 16 0;


#set base midi note
set BASENOTE 60;

# start command (group 1)
set a [createctrl 1 0 144 0 [ expr $BASENOTE + $i] ];
setscript1 1 $a "toggleplaysync $tmp";
# mute command (group 2)
set a [createctrl 2 0 144 0 [ expr $BASENOTE + $i] ];
setscript1 2 $a "seqtogglemute $tmp";
# loop command (group 3);
set a [createctrl 3 0 144 0 [ expr $BASENOTE + $i] ];
setscript1 3 $a " seqtoggleloop $tmp";
# copy command (group 4)
set a [createctrl 4 0 144 0 [ expr $BASENOTE + $i] ];
setscript1 4 $a " clearseq $tmp; copyseq 0 $tmp";
}

###### Control ######################

#Mode

#null mode (set second layer to group 10)
#key w
set a [createctrl 0 1 60]
setscript1 0 $a {setcgroup2 10; printtitle "null mode"}
```

```
#start mode (set second layer to group 1)
#key w
set a [createctrl 0 1 119]
setscript1 0 $a {setcgroup2 1; printtitle "start mode"}

#mute mode (set second layer to group 2)
#key x
set a [createctrl 0 1 120]
setscript1 0 $a {setcgroup2 2; printtitle "mute mode"}

#loop mode (set second layer to group 3)
#key c
set a [createctrl 0 1 99]
setscript1 0 $a {setcgroup2 3; printtitle "loop mode"}

#copy mode
#key v
set a [createctrl 0 1 118]
setscript1 0 $a {setcgroup2 4; printtitle "copy mode"}


#RECORD
#key space
#syncronized record with internal messages (record time = 4 beats)
set a [createctrl 0 1 32]
setscript1 0 $a {syncemit 1 1 80 0; syncemit 2 1 80 1 }

#internal messages 80 0 and 80 1
#start recording
set a [createctrl 0 80 0]
setscript1 0 $a {recbegin; print "recbegin"}
#stop recording
set a [createctrl 0 80 1]
setscript1 0 $a {recseq 0; print "recend"}

################ Default value ######
setbpm 120

# first active group : group 0 (key)
setcgroup 0

# first active group : group 1 (Midi channel set to 0)
setcgroup2 10
```

8 Usefull links

MIDI spcification :

- http://www.midi.org
- http://www.harmony-central.com/MIDI/Doc/

Tcl documentation :

- http://www.tcl.tk/doc/

Sound & MIDI Software For Linux :

- http://linux-sound.org/