Controlling the *mTDelHarmo16~* [1] object

Alain Bonardi

General description of the mTDelHarmo16~ object

mTDelHarmo16~ object is a set of 16 processes, each of them including a delay line and an harmonizer by doppler effect (temporal approach thanks to variable delay) with possible reinjection to any line. It has been developped using Faust language (see *mTDelHarmo16.dsp* code).

The general scheme of processing is the following (figure 1):

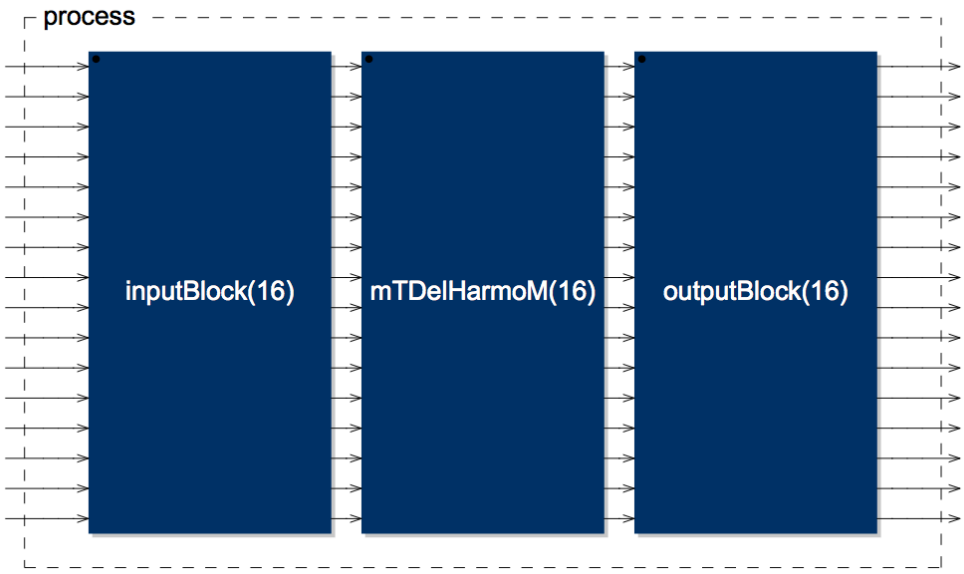

Figure 1. General scheme of processing of the *mTDelHarmo16~* object.

The *inputBlock* and the *outputBlock* are sets of 16 linear gains enabling to vary the levels of inputs and outputs. We will focus on the description of the *mTDelHarmoM* block, whose structure appears on the following block-diagram (figure 2). It only includes two blocks: *fdToMatrixBlock* and *DelHarmoBlock*. The first one (*fdToMatrixBlock*) deals with reinjection, enabling to send the output of any line to the inputs of any combination of lines thanks to an internal matrix. The second one (*DelHarmoBlock*) processes delaying and harmonizing that are organized in two separate blocks[2] (see figure 3).

[1] From version 1.2 on, this module is proposed with 8, 12 or 16 processing lines. The principles of the treatment are exactly the same, the only difference is the number of lines. Faust codes are in mTDelHarmo8.dsp, mTDelHarmo12.dsp, mTDelHarmo16.dsp files. Generated objects for Max and PureData are mTDelHarmo8~, mTDelHarmo12~, mTDelHarmo16~. Moreover all delay lines are now limited to a maximum delay that is 21 seconds.

[2] The *DelBlock* includes 16 delay lines but has 32 inputs to enable reinjection (each input is the sum of an incoming signal and of the reinjection).

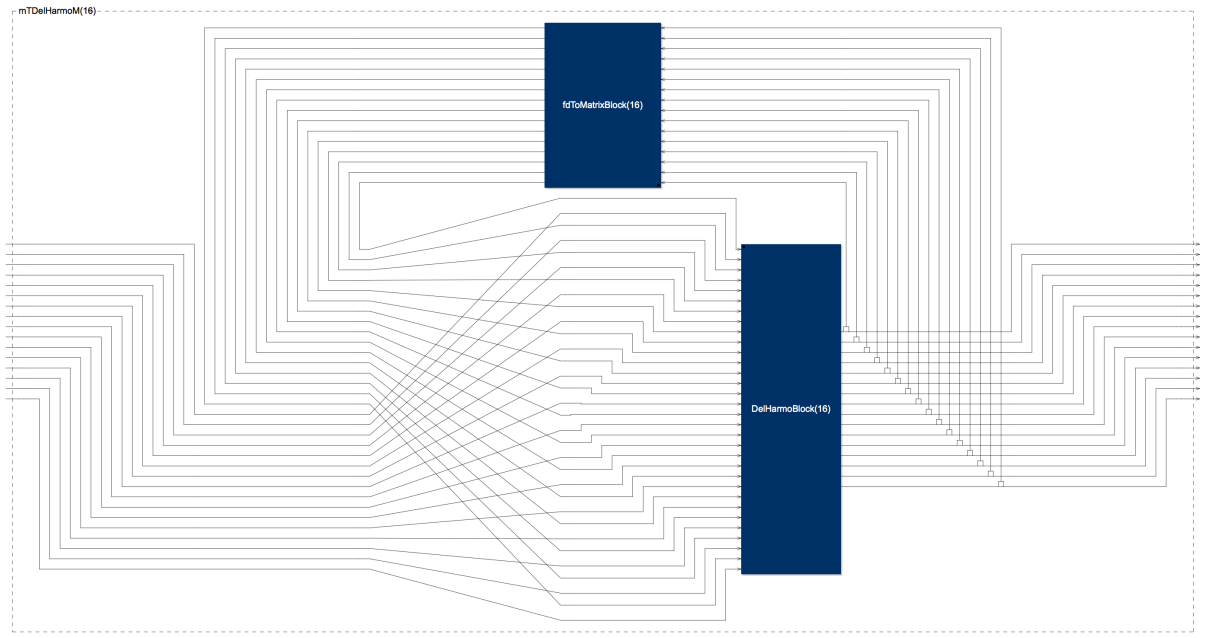

Figure 2. The *mTDelHarmoM* block-diagram.

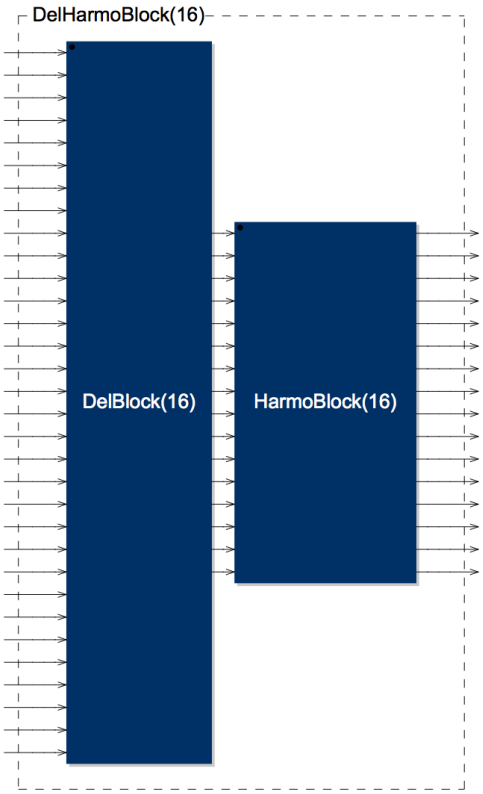

Figure 3. The two blocks composing the *DelHarmoBlock*: *DelBlock* and *HarmoBlock*.

Each delay line is a double overlapped delay enabling the change of duration without clicking (figure 4).

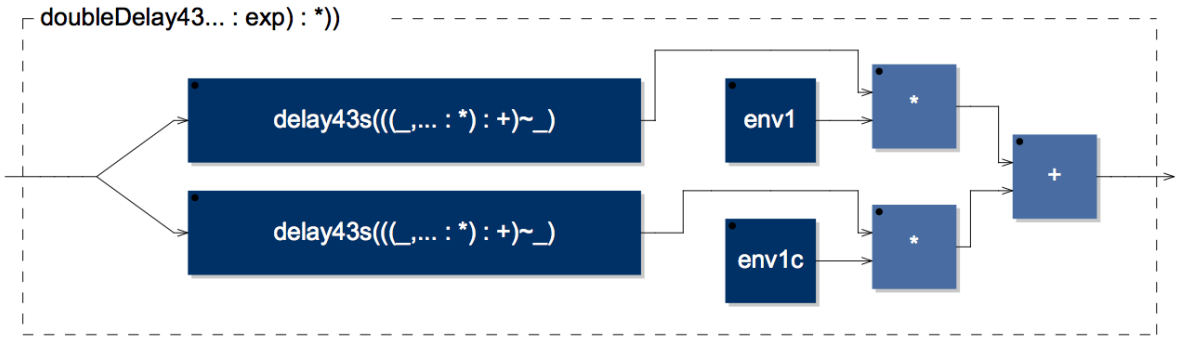

Figure 4. A delay line made of a double overlapped delay.

Each harmonizer consists in four overlapped variable delays (doppler effect), see figure 5.

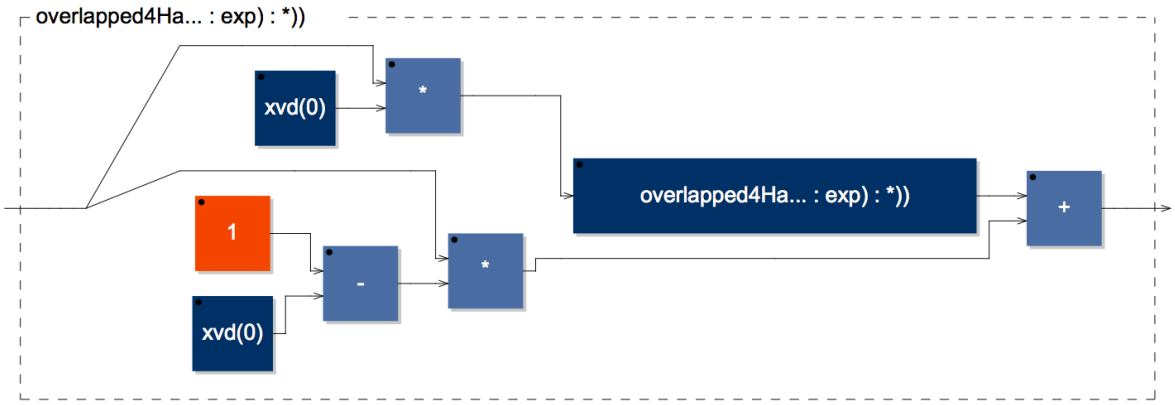

Figure 5. The structure of the harmonizers based on four overlapped variable delays.

For each process {delay line + harmonizer}, we have the following control parameters:

- the **duration** of the delay in milliseconds
- the **feedback** value, between 0 and 0.99
- the **xvd** value (effeX versus delay), that sets the distribution between the harmonizer (1) and the delay (0)
- the **transposition** in midicents (between -2400 and +2400 midicents)
- the linear **input gain** (between 0 and 1)
- the linear **output gain** (between 0 and 4).

Moreover there are global controls applied to the 16 lines:

- the **delay stretch**: this factor multiplies all the delay durations. If set to 1, no change. If set to 2, the delay durations are multiplied by 2, etc. Minimum is 0.01, maximum is 10 [3].
- the **transposition stretch**: this factor multiplies all the transpositions (expressed in midicents). If set to 1, no change. If set to 2, the transpositions are multiplied by 2, etc. If multiplied by -1, the transpositions are inverted. Minimum is -10 and maximum is 10 [4].

[3] Delay durations are however limited to 21 seconds.
[4] Transpositions remain between -2400 and 2400 midicents.

- the **smooth duration**: this is the global duration in milliseconds of the interpolation between current values and new values of control. This duration is set between 20 and 5000 milliseconds.
- the **harmonizer window width**: this global parameter contributes to transform the timbre of the transposed sound. The parameter is a decimal number between 0.0 and 127.0. The lower the original sound is, the higher the value should be, and reciprocally. But values should be tested and should enable different results between faithful transpositions and the ones that also alter the original timber.

Finally the reinjection is managed thanks to a matrix that enables to send any output to any combination of inputs. This internal matrix contains 16 x 16 = 256 toggles (each of them values either 1 or 0).

Direct control of *mTDelHarmo16~* object [Max and PureData]

The first and most simple way consists in controlling the *mTDelHarmo16~* object by sending messages to its first inlet. This is possible both in Max and PureData.

1) For Max software, the maxhelp patch of the object shows (in a large message on the right) how to update control values of the *mTDelHarmo16~* object (figure 6).

In Max, the messages use the following addresses:

- durations are indicated by *d00*, *d01*, …, up to *d15*
- feedback values are indicated by *fd00*, *fd01*, … up to *fd15*
- xvd values are indicated by *xvd00*, *xvd01*, …, up to *xvd15*
- input gains are indicated by *inp00*, *inp01*, …, up to *inp15*
- output gains are indicated by *out00*, *out01*, …, up to *out15*
- the duration stretch is indicated by *dStretch*
- the harmonizer stretch is indicated by *hStretch*
- the harmonizer window width is indicated by *hWin*
- the smooth duration is indicated by *smoothDuration*

For the reinjection matrix, the convention is a bit different. The various cells are indicated by:

- on the first row: *r000* *r001* … *r015*
- on the second row: *r016* *r017* *r031*
- on the third row: *r032* *r033* *r047*
- …
- on the 16th row: *r240* *r241* *r255*

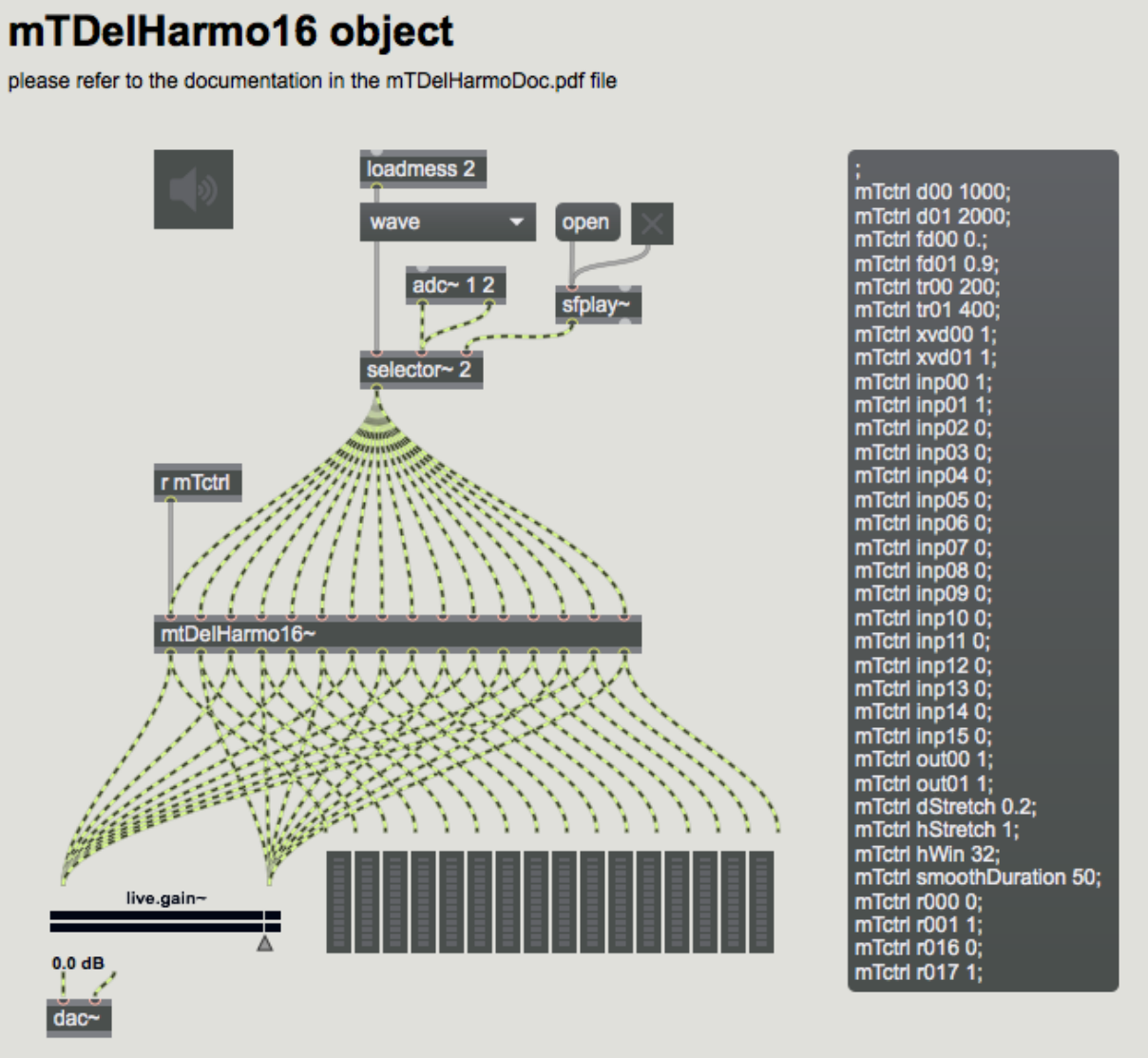

Figure 6. The maxhelp patch of the *mTDelHarmo16~* object.

2) For PureData software, the system of messages works the same but the addresses used are different. Figure 7 shows the help patch of the *mTDelHarmo16~* object.

In PureData, the messages use the following addresses:

- durations are indicated by *d-0*, *d-1*, *d-2*, …, *d-9*, *d10* up to *d15*
- feedback values are indicated by *fd-0*, *fd-1*, …, *fd-9*, *fd10* up to *fd15*
- xvd values are indicated by *xvd-0*, *xvd-1*, …, *xvd-9*, *xvd10* up to *xvd15*
- input gains are indicated by *inp-0*, *inp-1*, …, *inp-9*, *inp10* up to *inp15*
- output gains are indicated by *out-0*, *out-1*, …, *out-9*, *out10* up to *out15*
- the duration stretch is indicated by *dStretch*
- the harmonizer stretch is indicated by *hStretch*
- the harmonizer window width is indicated by *hWin*
- the smooth duration is indicated by *smoothDuration*

For the reinjection matrix, the various cells are indicated by:

- on the first row: *r-0* *r-1* … *r-15*

- on the second row: *r-16* *r-17* *r-31*
- on the third row: *r-32* *r-33* *r-47*
- …
- on the 16th row: *r240* *r241* *r255*

(for indices from 0 to 99: *r-0*, *r-1*, …, *r-99*, for indices from 100 to 255 : *r100*, *r101*, …, *r255*).

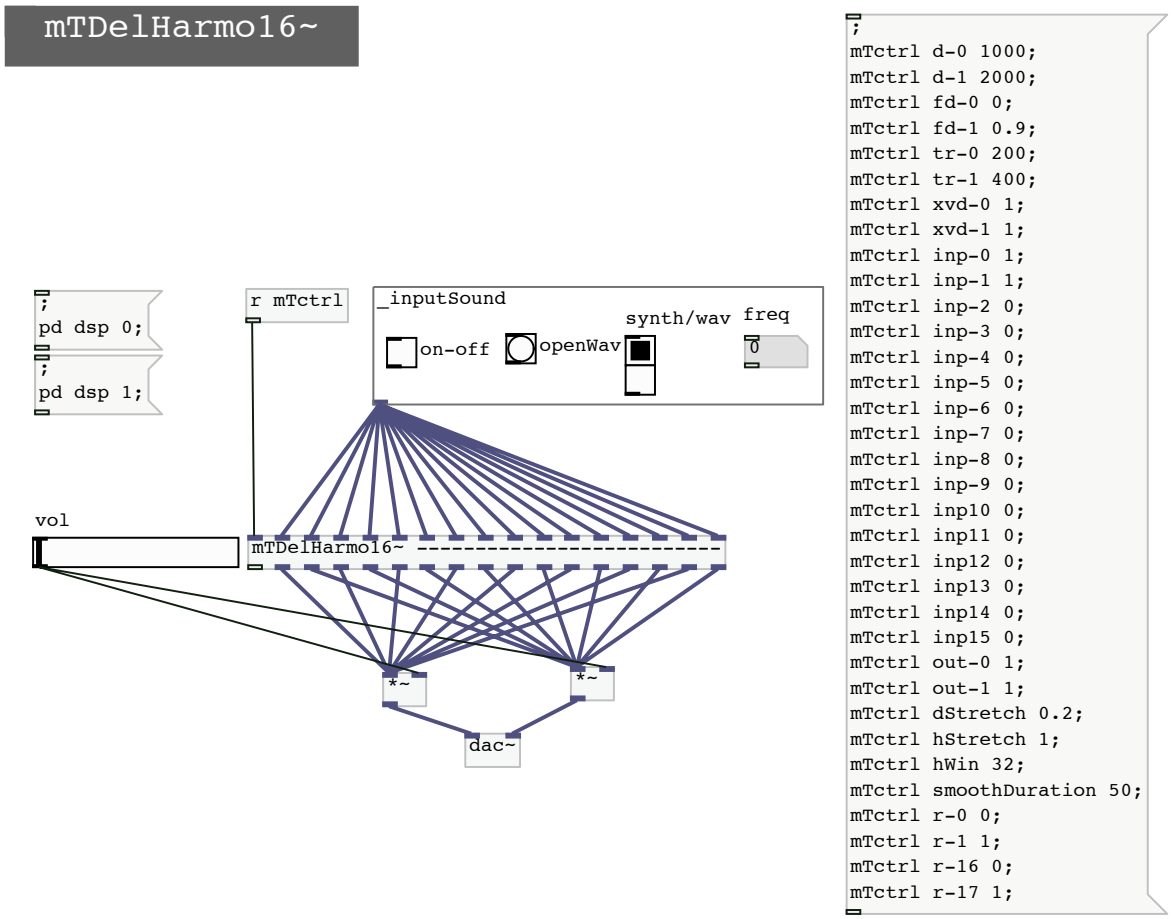

Figure 7. The PureData help patch of the *mTDelHarmo16~* object.

Control of mTDelHarmo16~ object through a javascript program [Max]

The second way to control the *mTDelHarmo16~* object in Max consists in using a Javascript program that handles high level messages that are easier and more concise to implement. This Javascript program implemented in Max is named *mTDelHarmo16MessageHandler.js*. It is encapsulated in an abstraction named *mTDelHarmo16Controller.maxpat* designed to be used in a bpatcher in Max (see figure 8).

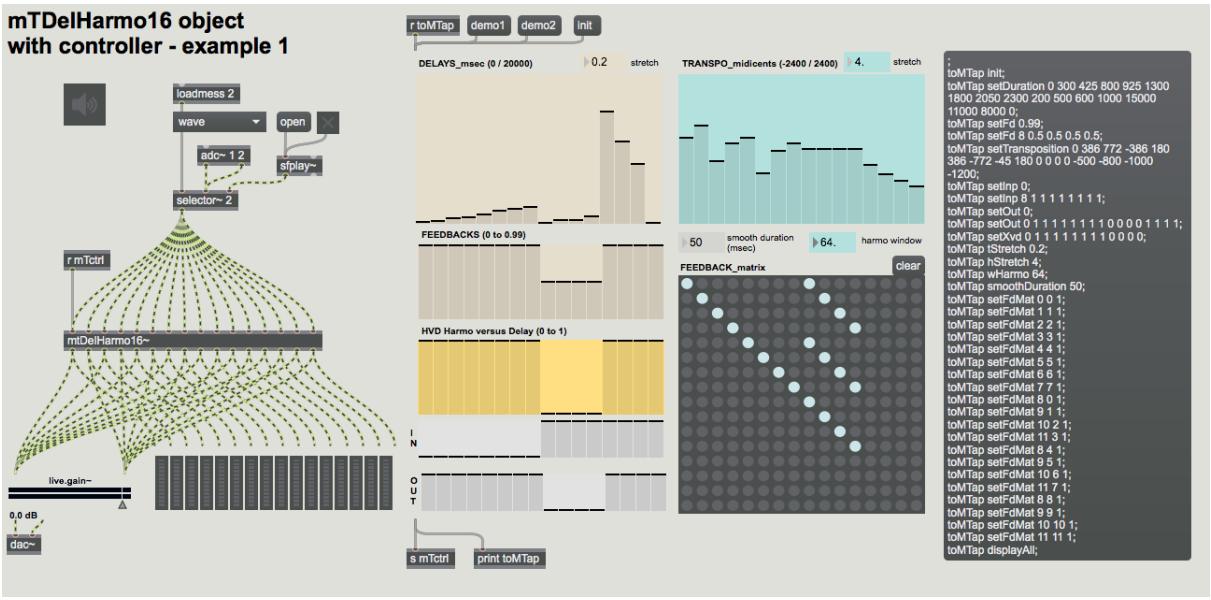

Figure 8. The abstraction *mTDelHamo16Controller* encapsulated in a bpatcher (center of the figure).

Delay durations

setDuration(list)

This message sets duration values and displays them on the interface

If only one argument, this is the common value of all durations
If there are several arguments: the 1st element of the list is a starting index (i)
Other elements of the list are durations : d0, d1, etc.
Therefore duration # i is set to d0, duration # (i+1) is set to d1, etc.

Examples

- with a single argument: setDuration(100) sets all durations to 100
- with one index and one value: setDuration(1, 200) sets dur[1] to 200
- with one index and several values: setDuration(3, 300, 100, 500) sets dur[3] to 300, dur[4] to 100 and dur[5] to 500.

outputAllDurations()

This message sends to outlet 1 all the durations prepended by a *mDelToDisplay* message

Feedback values

setFd(list)

This message sets feedback values and displays them on the interface

If only one argument, this is the common value of all feedbacks
If there are several arguments: the 1st element of the list is a starting index (i)
Other elements of the list are feedbacks: fd0, fd1, etc.
Therefore feedback # i is set to fd0, feedback # (i+1) is set to fd1, etc.

Examples

- with a single argument : setFd(0.6) sets all durations to 0.6
- with one index and one value : setFd(1, 0.2) sets fdbk[1] to 0.2
- with one index and several values : setFd(3, 0.3, 0.1, 0.5) sets fdbk [3] to 0.3, fdbk [4] to 0.1 and fdbk [5] to 0.5.

outputAllFds()
Sends to outlet 1 all the feedbacks prepended by a *mFdToDisplay* message

Transposition values

setTransposition(list)
This message sets feedback values and displays them on the interface

If only one argument, this is the common value of all transpositions
If there are several arguments: the 1st element of the list is a starting index (i)
Other elements of the list are transpositions: tr0, tr1, etc.
Therefore transposition # i is set to tr0, transposition # (i+1) is set to tr1, etc.

Examples

- with a single argument : setTransposition(250) sets all transpositions to 250
- with one index and one value : setTransposition(1, 200) sets tra[1] to 200
- with one index and several values : setTransposition(3, 300, 100, 500) sets tra[3] to 300, tra[4] to 100 and tra[5] to 500.

outputAllTranspositions()
This message sends to outlet 1 all the feedbacks prepended by a *mTraToDisplay* message

Xvd values (Xvd = effeX versus delay)

setXvd(list)
This message sets xvd values and displays them on the interface.

If only one argument, this is the common value of all xvd values
If there are several arguments: the 1st element of the list is a starting index (i)
Other elements of the list are xvd values : xvd0, xvd1, etc.
Therefore xvd # i is set to xvd0, xvd # (i+1) is set to xvd1, etc.

Examples

- with a single argument : *setXvd(0.5)* sets all xvd to 0.5
- with one index and one value : *setXvd(1, 0.2)* sets xvd[1] to 0.2
- with one index and several values : *setXvd(3, 0.3, 0.1, 0.5)* sets xvd[3] to 0.3, xvd[4] to 0.1 and xvd[5] to 0.5.

outputAllXvds()
This message sends to outlet 1 all the xvd prepended by a *mXvdToDisplay* message.

Input values

setInp(list)
This message sets input gain values and displays them on the interface

If only one argument, this is the common value of all input values
If there are several arguments: the 1st element of the list is a starting index (i)
Other elements of the list are input values : inp0, inp1, etc.
Therefore inp # i is set to inp0, inp # (i+1) is set to inp1, etc.

Examples

- with a single argument : setInp(0.5) sets all inputs to 0.5
- with one index and one value : setInp(1, 0.2) sets inp[1] to 0.2
- with one index and several values : setInp(3, 0.3, 0.1, 0.5) sets inp[3] to 0.3, inp[4] to 0.1 and inp[5] to 0.5.

outputAllInps()
This message sends to outlet 1 all the inputs prepended by a *mInpToDisplay* message.

Output values

setOut(list)
This message sets output gain values and displays them on the interface

If only one argument, this is the common value of all output values
If there are several arguments: the 1st element of the list is a starting index (i)
Other elements of the list are output values : out0, out1, etc.
Therefore out # i is set to out0, out # (i+1) is set to out1, etc.

Examples

- with a single argument : setOut(0.5) sets all outputs to 0.5
- with one index and one value : setOut(1, 0.2) sets out[1] to 0.2
- with one index and several values : setOut(3, 0.3, 0.1, 0.5) sets out[3] to 0.3, out[4] to 0.1 and out[5] to 0.5.

outputAllOuts()
This message sends to outlet 1 all the outputs prepended by a *mOutToDisplay* message.

Global parameters

tStretch(s)
Sets the temporal stretch to s.

hStretch(s)
Sets the harmonizer stretch to s.

wHarmo(x)
Sets the harmonizer window width to x.

smoothDuration(x)
Sets the smoothing duration to x.

Feedback routing

setFdMatCell(x, y, val)
Sets the value of the cell of coordinates (x, y) to val (but does not send the value to be displayed).

setFdMat(x, y, val)
Same as setFdMatCell but also displays the value val using a *matrixctrl* Max object.

setFdMatList(list)
Sets feedback routing values and displays them on the interface

If only one argument, this is the common value of all feedback routing values
If there are several arguments: the number of arguments should be a multiple of 3, since arguments are given by triplets with the coordinates (x, y) of the cell and its value val.

Examples

- with a single argument: setFdMatList(1) sets all cells of the matrix to 1.
- with triplets: setFdMatList(4, 5, 1, 3, 9, 1) sets both cells of coordinates (4, 5) and (3, 9) to 1.

Control of *mTDelHarmo16~* object using a PureData interface

The *mTDelHarmo16.pd* patch (figure 9) contains both the *mTDelHarmo16~* object and the interface to control the object with PureData user interfaces objects (sliders, checkboxes, number boxes).

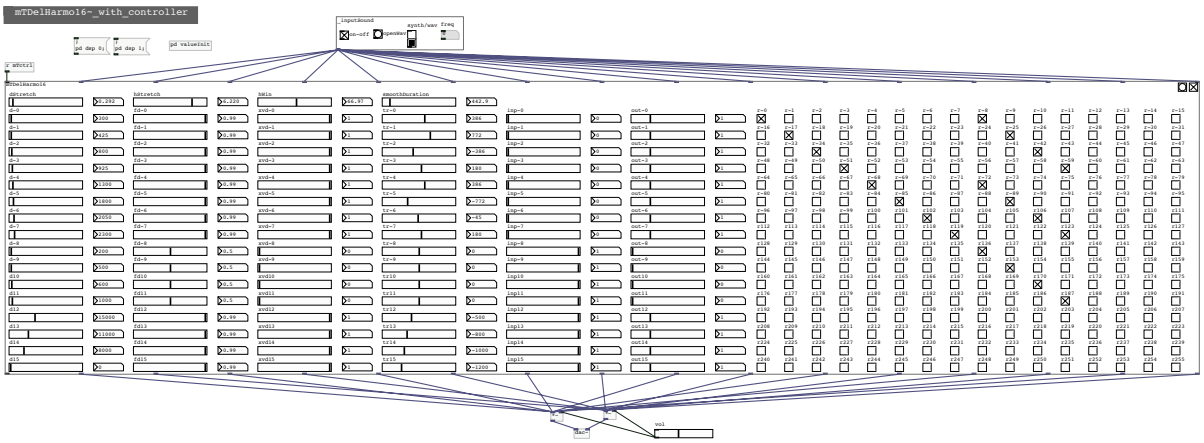

Figure 9. The *mTDelHarmo16.pd* patch.

The labels of the controllers are indicated just above them. They can be addressed by messages as shown on figure 10.

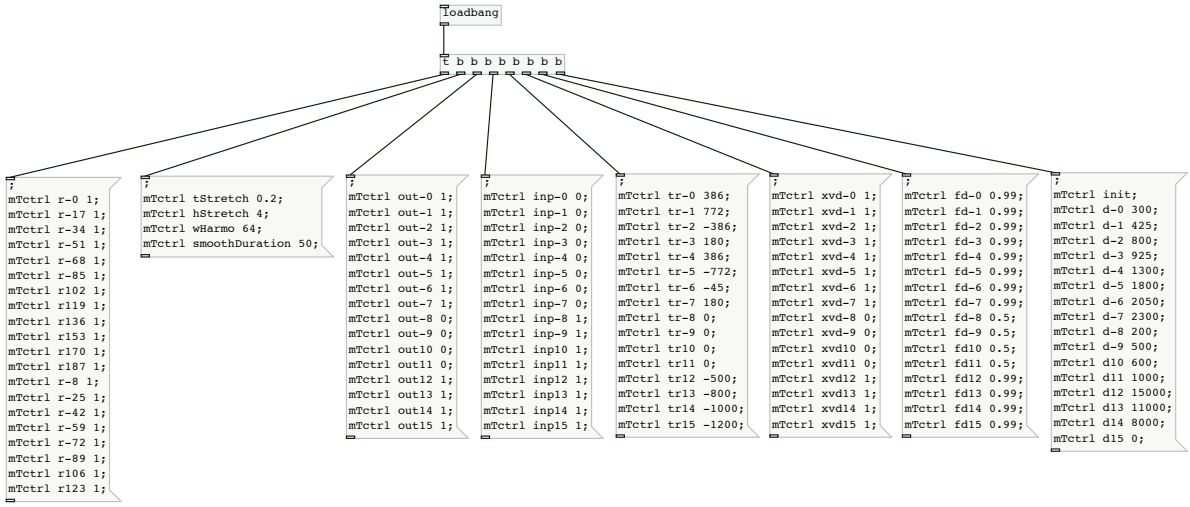

Figure 10. Examples of messages to control the *mTDelHarmo16~* object via the *mTDelHarmo16.pd* interface.