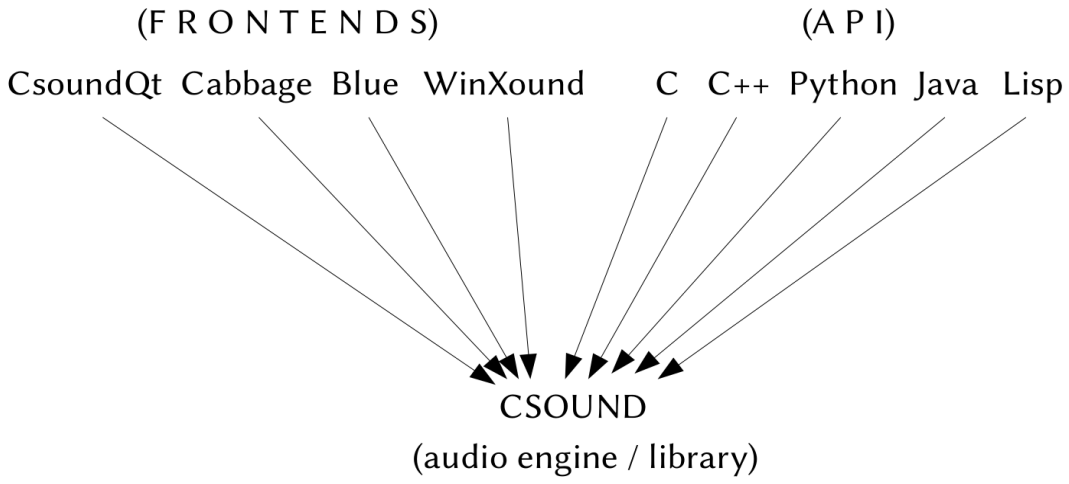
# Joachim Heintz
# Live Elektronics with modern Csound (and CsoundQt)

**IEM Graz**
**4 December 2018**

## 01 Download and Help

- Download **Csound** (**CsoundQt** is **included** in the installers for Mac and Windows)
  https://csound.com/download.html
- **Latest CsoundQt** (if necessary)
  https://github.com/CsoundQt/CsoundQt/releases
- **Csound6~** Object for **PD**
  https://github.com/csound/csound_pd/releases
- Free Introduction online (Csound **FLOSS Manual**)
  http://write.flossmanuals.net/csound
- CsoundQt Website and Documentation
  http://csoundqt.github.io
  http://csoundqt.github.io/pages/documentation.html
- *Getting Started* is integrated in CsoundQt
  Examples > Getting Started
- New Csound Book (2016)
  Lazzarini et.al. — Csound, A Sound and Music Computing Language
  Springer Site Table of Contents

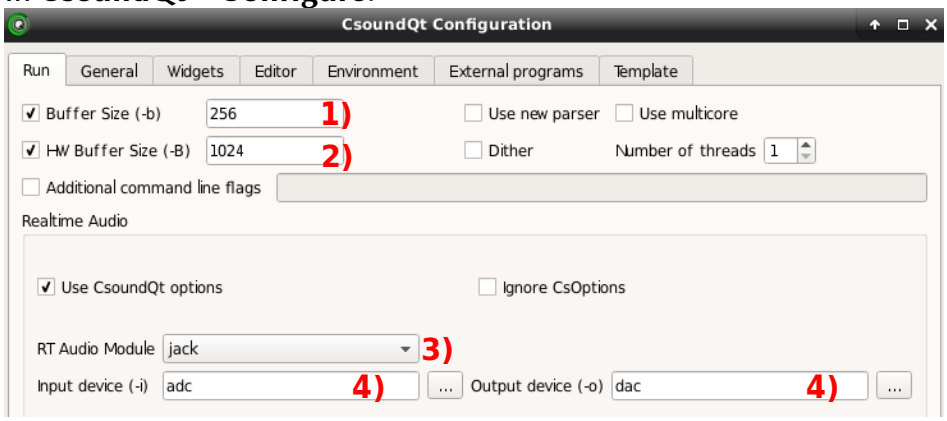
## 02 Csound and Frontends




In the case of CsoundQt
**CsoundQt ->** Editor, Widgets, Environment
**Csound ->** Syntax, Reference Manual, Audio Engine

**03 Some Basic Settings**

In **CsoundQt > Configure:**




1) Set software buffer size to 256 samples
2) Set hardware buffer size to 1024 samples
3) Choose Real Time Audio Module according to you operating system (usually portaudio)
4) Choose **adc** as input device and **dac** as output device. **Select your audio card in the operating system.**

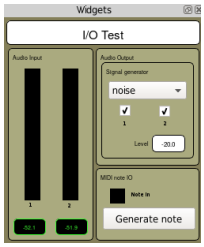
In your **Csound program (.csd file):**

```
<CsoundSynthesizer>
<CsOptions>
</CsOptions>
<CsInstruments>

sr = 44100 5)
ksmps = 32  6)
nchnls = 2  7)
0dbfs = 1  8)

instr 1
endin

</CsInstruments>
<CsScore>

</CsScore>
</CsoundSynthesizer>
```

5) Set sample rate (usually 44100 or 48000)
6) Set number of samples per block or control-cycle (usually 32)
7) Set number of channels (usually 2)
8) Set the number which represents 0 dB full scale (always 1)

Save this in Configure > Template. It will then be opened for each new file in CsoundQt.

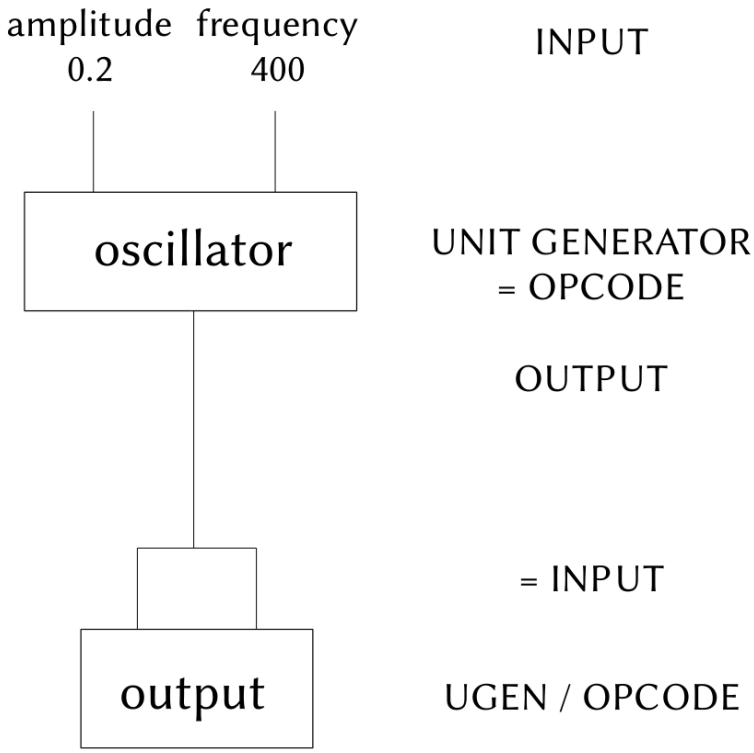
Now run **CsoundQt > Examples > Useful > IO Test**:
1. Klick on the “Run” button top left.
2. Klick on the “Widgets” button. You should see this panel:




3. Klick on the checkboxes to hear noise test signal.

**04 Coding**

A typical (very simple) flow chart




This is the way to translate such a flow chart to Csound code:

- Know the **Opcode Names** (here: oscillator = *poscil*, output = *out*)
- For each Input-Opcode-Output chain, write one line in the order
  OUTPUT OPCODE INPUT
  (If there are more than one inputs or output, seperate by commas.)
- The first character of a variable specifies the type:
  - **i**Bla -> stays constant during the call of this instrument (instance)
  - **a**Bla -> audio signal (every sample is rendered)
  - **k**Bla -> control signal (every block is rendered)

The actual code is encapsulated in an **Instrument** (starting with the keyword *instr* and ending with the keyword *endin*):

```
instr Sine_Test
 aSine poscil 0.2, 400
 out aSine, aSine
endin
```

There are different possibilities to **call** an instrument. (Or more precise: to create an instance.)
1. Write a line like this in the *CsScore* section of the .csd file:

```
   i "Sine_Test" 0 3
   (= call instrument “Sine_Test” with start=0 and duration=3)
```

2. Right-click in the widgets panel and select “Create Button”. The *Properties* dialog opens. Insert the same score line as “Event”:

```
   i "Sine_Test" 0 3
```

Now start Csound **with empty score section**. Klick on the Button then starts the instrument.

3. Using the *schedule* opcode, the instrument can be called inside the Csound Code (not in the score section):

```
   schedule "Sine_Test", 0, 3
```

### 05 Call yourself

By this way, an instrument can also call itself. This code plays instrument *Perpetuum_Sine* for two seconds and calls the next instance in three seconds (= one second after the tone):

```
instr Perpetuum_Sine
 schedule “Perpetuum_Sine”, 3, 2
 aSine poscil 0.2, 400
 out aSine, aSine
endin
```

In the examples this is extended in different steps resulting in this version:

```
instr Perpetuum_Sine_5

 iPreviousNote = p4
 iDeviation random -3, 3
 iThisNote = iPreviousNote + iDeviation

 iDb random -30, -6
 iAmp = ampdb(iDb)
 aEnv transeg iAmp, p3, -3, 0
 aSine poscil aEnv, mtof(iThisNote)
 out aSine, aSine

 iStart random 1/3, 2
 iDur random 1, 7
 schedule "Perpetuum_Sine_5", iStart, iDur, iThisNote

endin
schedule "Perpetuum_Sine_5", 0, 3, 60
```

Or if you prefer **functional syntax**:

```
instr Perpetuum_Sine_6

 iThisNote = p4 + random:i(-3,3)

 aEnv = transeg:a(ampdb(random:i(-30,-6)),p3,-3,0)
 aSine = poscil:a(aEnv,mtof(iThisNote))
 out(aSine,aSine)

 schedule("Perpetuum_Sine_6",random:i(1/3,2),random:i(1,7),iThisNote)

endin
schedule("Perpetuum_Sine_6",0,3,60)
```

### 06 Live Input

Get a microphone as audio signal via the *inch* opcode.
(Be aware of possible feedback with the following code which directly outputs the input.)

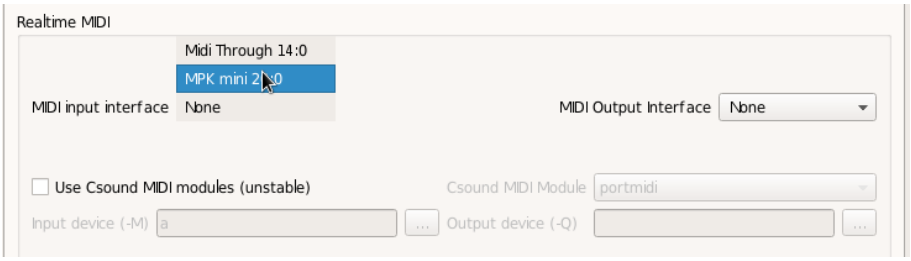
```
 aMic inch 1
 out aMic, aMic
```

As simple example, this is ring modulation of the live input with a modulator frequency of 400 Hz:

```
 aMic inch 1
 aRM poscil aMic, 400
 out aRM, aRM
```

## 07 MIDI Keyboard (note on/off messages)

First select the MIDI device in CsoundQt's Configure > Run:




Per default every key press creates an instrument instance:

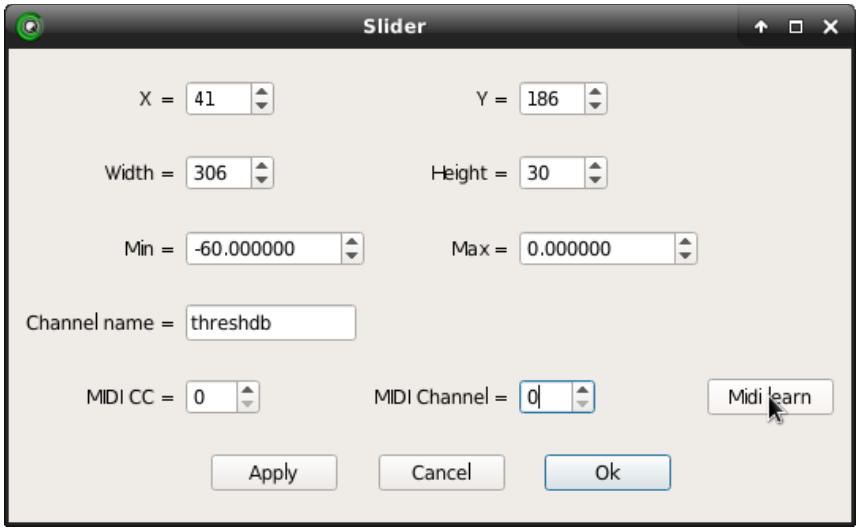
```
instr OneMIDINote
 aNote poscil .2, 400
 out aNote, aNote
endin
```

Reading the MIDI Key number (60 = C4) and the velocity (1-127) and applying a fade-out if a key is released creates a simple (and unlimited polyphone) synthesizer:

```
 iCps cpsmidi
 iVel veloc -30, 0
 aNote poscil ampdb(iVel), iCps
 aEnv linenr aNote, 1/20, 1/5, 1/100
 out aEnv, aEnv
```

## 08 MIDI Control Change

If you use CsoundQt, it is most simple and flexible to connect the MIDI controller with a widget (slider, knob, scroll box, spin box). Use the "Midi learn" button to assign the controller:



### 09 ASCII Key (Computer keyboard) as control

The opcode *sensekey* returns two values: *kDown* gets 1 if any key has been pressed (otherwise 0); *kKey* returns the ASCII key number. This code simply shows the number in a widget with channel "key":

```
kKey, kDown sensekey
if kDown == 1 then
 outvalue "key", kKey
endif
```

This code calls the instrument *Beep* (with a random duration) each time the space bar is pressed:

```
if kDown == 1 && kKey == 32 then
 event "i", "Beep", 0, random:k(0.2,1)
endif
```

See also in CsoundQt Examples > Useful > SF Snippets Player.

### 10 Receive OSC (Open Sound Control) from PD or any other application

Csound has to know this to receive OSC messages:

- the port which is used to send OSC by the sender
- the string with the OSC address
- the data type of the values in the message (i=integer, f=float, s=string etc.)

This code receives a (float) number from PD via the address "/test" on port 9001 and displays it:

```
giPort OSCinit 9001

instr Receive
 kVal init 0
 kPing OSClisten giPort, "/test", "f",  kVal
 outvalue "val", kVal
endin
schedule "Receive", 0, 999
```

### 11 Send OSC (to PD or any other application)

This you need to specify if you want to send an OSC message:

- the IP address of the receiver, or "localhost" (or "") for this computer
- the port which is used to send the OSC message
- the OSC address
- the data types which are being sent

This sends a random number between 1 and 10 once a second PD at the address "/test2" via port 9002:

```
instr Send
 kVal randomh 1, 10, 1
 OSCsend kVal, "", 9002, "/test2", "f",  kVal
endin
schedule "Send", 0, 999
```