About this Manual

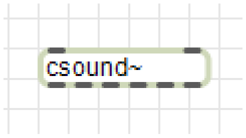
This is the manual for **csound~ v1.1.0**. Some v1.0.x features have been deprecated and, although they still work, they will not be discussed in this manual. Also, this manual is not a good introduction to either Max or Csound programming.

A note about terminology: the term **csound~** refers to the Max external while **Csound** (with a capitol C and no tilde) refers to the Csound audio programming language.

An excellent tutorial for csound~ can be found at:

http://en.flossmanuals.net/bin/view/Csound/CsoundInMax

What is csound~?

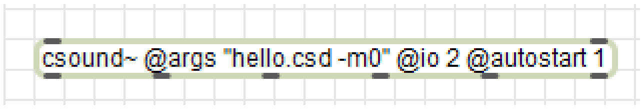


csound~ is a Max external that allows you to run Csound inside Max. Think of csound~ as a plugin and Csound as the synthesis engine. csound~ inlets accept audio, MIDI, control, and event data. After doing some processing with Csound, similar data appears at the outlets.

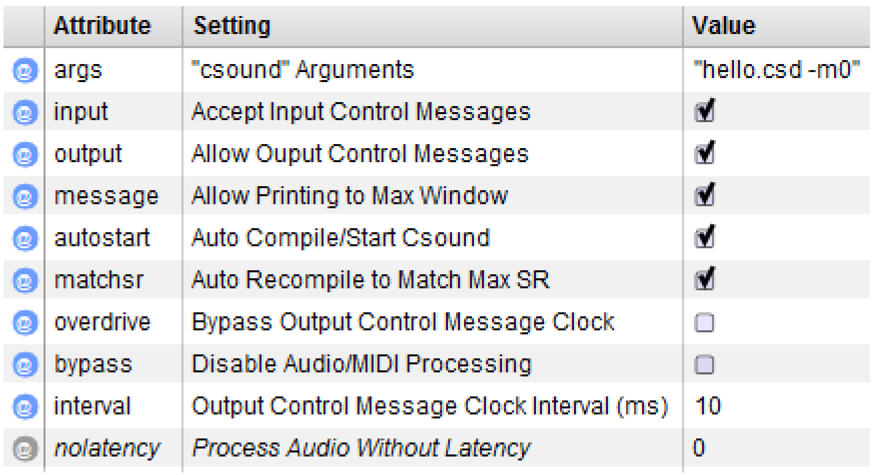
Csound is an audio programming language that allows you to synthesize and process audio. It has many similarities to Max, but is programmed using text files rather than visual patches. Although programming in Max has many benefits, sometimes visual programming can be cumbersome, especially for large projects. csound~ was created, in part, to combine the ease of visual programming with the benefits of textual programming. There are many audio externals in Max that have similar counterparts in Csound, but they often differ in sound character (and sometimes quality). With csound~, you can use the best of both worlds.

csound~ Attributes

The attributes can be specified within the csound~ box or with the inspector window:



```
csound~ @args "hello.csd -m0" @io 2 @autostart 1
```



Attribute	Setting	Value
args	"csound" Arguments	"hello.csd -m0"
input	Accept Input Control Messages	✓
output	Allow Ouput Control Messages	✓
message	Allow Printing to Max Window	✓
autostart	Auto Compile/Start Csound	✓
matchsr	Auto Recompile to Match Max SR	✓
overdrive	Bypass Output Control Message Clock	☐
bypass	Disable Audio/MIDI Processing	☐
interval	Output Control Message Clock Interval (ms)	10
nolatency	*Process Audio Without Latency*	0

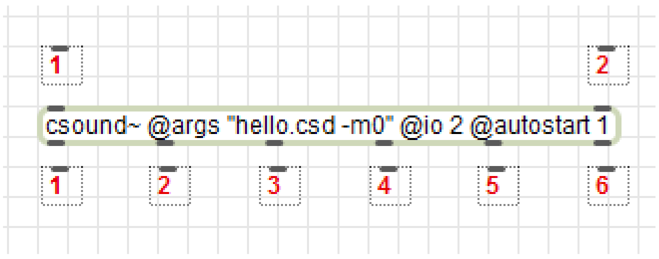
Attributes can also be set by sending messages of the form “attribute value” (no @ symbol).

Attribute List

Name	Default	Range	Description
@args	-	-	The arguments to the Csound command used to compile an orchestra. Usually contains a csd file.
@autostart	0	[0-1]	Controls automatic compilation of a Csound orchestra upon object creation. For this to work, a csd/orc/sco file(s) must be specified in the csound~ box with the **@args** attribute. If the **@args** attribute is missing, nothing will happen. If the specified orchestra requires certain control values to be initialized before the performance has started, don't use **@autostart**.
@bypass	0	[0-1]	When enabled, audio and MIDI processing is disabled. This also disables the automatic running of one k-cycle of the Csound performance that is normally done when Max DSP is off and an orchestra has compiled successfully.
@i	2	[1-128]	The number of signal inlets.
@input	1	[0-1]	Controls input control message processing. When enabled, **chnget** and **invalue** opcodes can receive values from Max messages sent to any csound~ inlet.
@interval	10	[1-inf]	The millisecond interval that determines the rate at which output control messages are sent from the Messages outlet. **@overdrive** = 1 overrides this setting.
@io	2	[1-128]	The number of signal inlets and outlets.
@matchsr	1	[0-1]	When enabled, csound~ will detect when the Max sample-rate is not equal to the Csound sample-rate. If a mismatch is detected, the Csound orchestra is recompiled with sample-rate and k-rate overrides (ksmps is preserved).
@message	1	[0-1]	Allows messages generated by csound~ or Csound to be printed to the Max window.
@nolatency	0	[0-1]	This attribute tells you when csound~ is processing audio from Max without latency. Currently, this attribute is read-only. To achieve zero-latency, set ksmps so that Max signal vector size modulo ksmps is zero (i.e. the remainder of vector size / ksmps = 0).
@o	2	[1-128]	The number of signal outlets.
@output	1	[0-1]	Enables output control message processing. When enabled, values generated by chnset and outvalue opcodes will be sent from the Messages outlet.
@overdrive	0	[0-1]	When enabled, the **@interval** attribute is ignored, and values generated by chnset and outvalue opcodes will be sent as soon as possible.

Inlets & Outlets

All inlets are audio inlets. All inlets accept messages. All inlets accept integers (interpreted as raw MIDI bytes). The four outlets to the right are non-audio outlets. The remaining oulets on the left are audio outlets. If there are two audio inlets and two audio oulets, then the inlet/outlet numbering is:

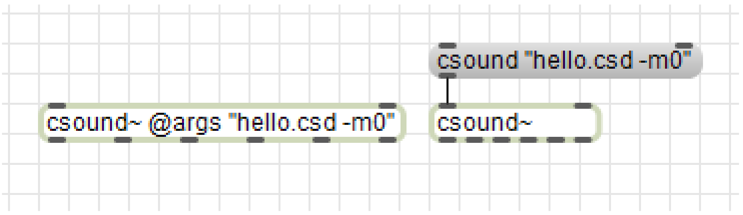


Outlet	Description
1	Audio output channel 1.
2	Audio output channel 2.
3	Sends output control messages generated by Csound opcodes like **outvalue** and **chnset**. Its also sends other types of messages (for example, the **rsidx** message).
4	Sends raw MIDI bytes generated by Csound.
5	Sends a bang when the Csound orchestra compiles successfully. This bang is often used to tell the patch that it should send initializing MIDI Control Change values to csound~.
6	Sends a bang when the Csound performance has finished or is prematurely stopped by the user.

To access audio inside your Csound orchestra, use the **inch** opcode. To send audio back to Max, use the **outch** opcode. Csound channel numbers start with one, which corresponds to the left-most inlet/outlet. Audio data is automatically scaled when Csound's 0dBFS level (typically 32768) does not match Max's 0dBFS (always 1).

The Csound Command

There are two ways to specify the csound command: the **@args** attribute and the **csound** message.



In most situations, you won't need to add any flags when specifying the command. csound~ coordinates communication between Max and Csound, so flags such as -i, -o, -M, -L, -B, -b are not needed. In the example above, the "-m0" flag prevents Csound from printing event messages (though compilation messages will still be printed). Remember to add double-quotes if the command has spaces.

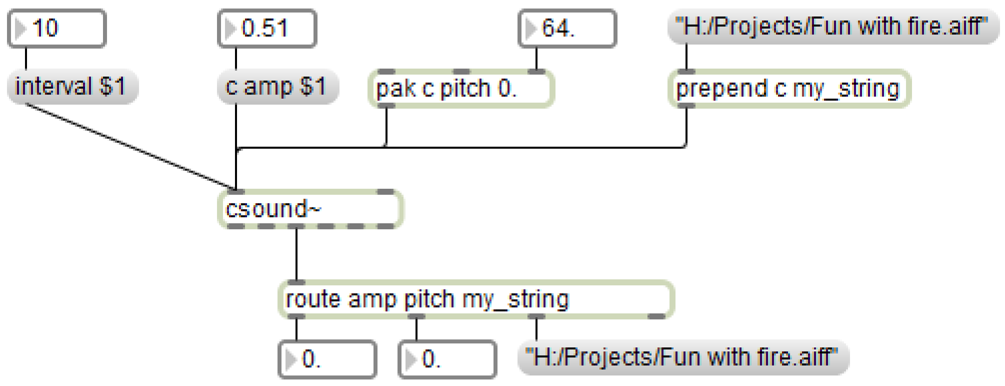
The csd file can be specified as a relative path (as in the example above). But keep in mind that relative paths will only work if your patch exists outside the Max search path. If you want to put a csound~ patch inside the Max search path, use absolute pathnames.

To enable non-realtime rendering, add **-o SomeFileName.aif** to the command.

You should never use these flags: **-+rtmidi, -M, -L, -B, -b**

If you're not rendering in non-realtime, don't use these flags: **-o**, **-i**

Control Messages



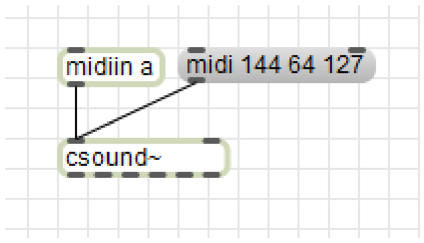
10
0.51
64.
"H:/Projects/Fun with fire.aiff"
interval $1
c amp $1
pak c pitch 0.
prepend c my_string
csound~
route amp pitch my_string
0.
0.
"H:/Projects/Fun with fire.aiff"

In the figure above, there are three input control messages being sent to csound~: amp, pitch, and my_string. The value of a control message can be a number or a string. Notice that control messages start with a **c** (or **control**).

Output control messages are sent from the Messages outlet (fourth from right). The **route** external is recommended as a way to distinguish between messages.

Notice the interval message being sent to csound~. This controls the rate at which output control messages are sent. In the figure above, the interval is set to 10 milliseconds. To increase the rate, lower the interval. If you want output control messages to be sent as soon as possible, use the message overdrive 1. Keep in mind that small intervals consume more cpu resources.

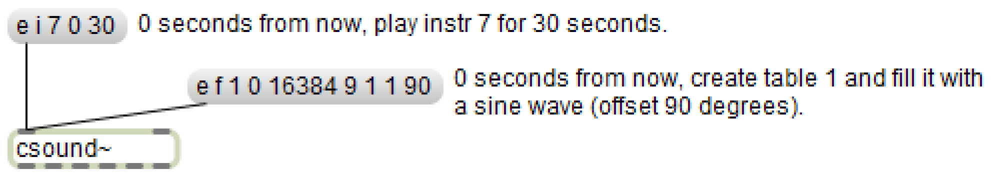
MIDI



Integers received in any inlet are interpreted as raw MIDI bytes. You can also use **midi** messages. Make sure each **midi** message contains a complete MIDI event. csound~ is able to process NoteOn, NoteOff, PitchBend, Aftertouch, ControlChange, ProgramChange, and ChannelAftertouch (sysex is ignored).

Remember to send initializing MIDI control change values after the csound orchestra has been compiled. Use the compiled bang outlet (2nd from right) to trigger the sending of MIDI initialization messages to csound~. If the MIDI opcodes in your orchestra are not properly initialized, you may hear nothing, something weird, or something very loud!

Events


e i 7 0 30
0 seconds from now, play instr 7 for 30 seconds.
e f 1 0 16384 9 1 1 90
0 seconds from now, create table 1 and fill it with a sine wave (offset 90 degrees).
csound~

csound~ accepts Csound score events using the **event** (or **e**) message. If you want the event to start immediately, use a start time of zero. If any element in the event has spaces (e.g. a pathname), wrap it with double-quotes.

Pathnames

In Windows, csound~ works with Max style paths, which is very similar to Windows style paths. The only difference is the slash type. Max uses forward slashes (/) and Windows uses back slashes (\). Since both Csound5 and csound~ work with forward slashes, just use forward slashes. Here's an example: **D:/samples/**

In MacOSX, csound~ works with either Max or POSIX style paths (as of v1.0.6). A typical Max style path:

MyFireWireHD:/samples/voices

and its POSIX equivalent:

/Volumes/MyFireWireHD/samples/voices

When you give csound~ a Max style path, it's converted to POSIX format internally.

The Sequencer

You can record and playback MIDI, **control,** and **event** messages. The data can be saved to json, xml, or binary files.

Use **recordstart** and **recordstop** to control recording. Use **playstart** and **playstop** to control playback. To save or load a sequence, use the **read** and **write** messages. The file extension you use determines the file format. If the extension is neither .json or .xml, the file is saved in binary format.

The sequencer makes creating high-quality non-realtime renderings of realtime performances easy. Just capture a performance using low-quality instruments that don't tax the cpu, then render it in non-realtime using high-quality/cpu-intensive instruments. The maximum recording time is 60 minutes and the timing resolution is approximately one millisecond.

If you start a non-realtime rendering (e.g. by adding "-o SomeFileName.aif" to your **csound** message) and a sequence is loaded, it will play back along with the rendering. Non-realtime rendering takes place in a separate thread.

Audio Latency

If the ksmps value set in your orchestra evenly divides the Max signal vector size, then csound~ will run in zero-latency mode. In other words, it will not delay the audio being processed. If ksmps does not evenly divide the Max signal vector size, ksmps samples of delay will be added to the processed audio.

If you're unsure whether csound~ is operating in zero-latency mode, open the inspector window and check the **@nolatency** attribute. If it equals 1, then you are indeed processing audio without latency.

Tips

Immediately after creating a new patch, there's no way to determine which folder contains that patch. After you create a csound~ instance, save and re-open the patch. That way, csound~ will know which folder contains the patch. This is important if you're using relative paths to specify your csd/orc/sco files.

If you're concerned about performance (perhaps in a live situation), then disable message output by passing **message 0** to csound~, or by adding the flag **-m0** to your **csound** message. This will prevent Csound from printing to the Max window.

outvalue and **chnset** processing can be disabled with "output 0". This is recommended if you're not using output control messages. **invalue** and **chnget** processing can be disabled with "input 0".

If you make changes to a csd/orc/sco file(s), but the names and locations haven't changed, simply send **start**, **bang**, or **reset** to recompile and re-start the performance. You don't have to re-send the **csound** message.

For greater timing accuracy of MIDI, **event**, and **control** messages to/from csound~:

Lower your orchestra ksmps (i.e. increase kr).

Decrease the MSP signal vector size(Options -> DSP Status...).

Enable "Max Scheduler In Overdrive" in the Options menu.

Enable "Scheduler In Audio Interrupt" in the DSP Status window (Options -> DSP Status...).

Send the message "overdrive 1" to csound~.

To minimize timing latency, decrease the I/O vector size in MaxMSP (Options -> DSP Status...). To find the best settings for a particular setup, adjust for maximum timing accuracy and minimum latency, then scale back if the audio is breaking up.

The **event** message allows you to load/replace Csound f-tables in realtime. However, Csound may **crash** when you are replacing a table that is being accessed by an instrument.

One crash-free solution is to use the **loadsamp** message with a positive table argument.

Another solution is to make sure there aren't any active instruments using that table. If you have an instrument that's activated by MIDI, then flush all MIDI notes and wait until all active instances stop. If your instrument is score activated, then use the turnoff or turnoff2 opcodes to deactivate active instances (or just wait until their durations expire).

Scripting the Patcher

Patcher scripting statements can now be added to csd files. They must be enclosed within <~> and </~> delimiters. If you're not familiar with patcher scripting, please consult the Max documentation. In a nutshell, statements such as:

<~> @maxclass slider @varname kittens @floatoutput 1 @min -1. @size 3. </~>

can be place anywhere inside your csd file. After loading a csd (with either **@args** or the **csound** message), you can send the **parse** message to generate Max objects specified in your csd. Optionally, you can send **parse tosub** if you want to create new objects inside a subpatch (to keep you main patch clean).

By default, companion objects are created for named (e.g. **@varname kittens**) scripted objects so that the values output from objects are sent to the object called "csound~". By naming the csound~ object "csound~", all values will be sent to your csound~ object.

The scripting feature is most likely to benefit those who are experienced with patcher scripting. It allows one to accelerate the initial phase of patch development. However, as of v1.1.0, the scripting feature is one-way; changes made to the patch are not reflected in the csd file.

csound~ Messages

Alternative message identifiers are enclosed in parentheses.

control (c)	Sends an input control message to Csound. The control message can be accessed inside your orchestra using the **chnget** or **invalue** opcodes. **@input** must be non-zero in order to process input control messages.
csound	Sets the command that will be passed to Csound when the **start** or **bang** message is received. The command must contain a csd/orc/sco file(s). If the command has spaces, wrap it in double-quotes.
event (e)	Sends a score statement to Csound during a realtime performance.
loadsamp	Loads an audio file into a Csound table. **arg 1**: Table argument. If it references a non-existent table, it will be created. If the positive and the table exists, it won't be replaced. This is useful when you want to replace table data on the fly without causing a crash. If the table is too small for the requested file, frames will be read until table is full. If the table argument is negative and the table exists but is too small for the file you want to load, then it will be replaced. Replacing tables during a performance is **NOT SAFE!** **arg 2**: Channel argument. It determines which channel of audio data is read. For stereo files, 1 is left, 2 is right. To read in all channels as interleaved data, use 0. **arg 3**: Filename. On Windows, use forward (/) slashes. On MacOSX, use either Max style paths or POSIX style paths. Relative paths are relative ot the patch containing csound~. **arg 4** (optional, default = 0): Read offset time (seconds). If negative, time is in sample frames. **arg 5** (optional, default = 0): Amount of time to read (seconds). If zero, read in all data or as much data as possible. If negative, time is in sample frames. Example: loadsamp 1 2 "/Users/Georg/samples/fun.aif" If table # 1 exists, then read in channel 2 of fun.aif until the table is full

	(the table is not replaced). If table # 1 does not exist, then create it and read in all channel 2 data. When using loadsamp, some of the table opcodes will not operate as expected. Some of the affected opcodes: **nsamp**,**ftchnls**,**ftlptim**,**ftsr** To minimize frustration, always load a similar type of sample to the one you're replacing. In other words, if a table contains a single channel of data at 44100 sr, then only load samples with similar characteristics in that table.
message (printout)	Controls printing to the Max window. A non-zero argument enables printing.
midi (m)	Sends a complete MIDI message to Csound. Example: **m 144 64 127**
open	Opens the current csd/orc/sco file(s) in the default editor. In order for this to work, you must first associate .csd/.orc/.sco files with your favorite Csound text editor.
path	Set the current directory for this instance of csound~. Argument must be an absolute pathname.
playstart	Starts playback of the loaded sequence.
playstop	Stops playback of the loaded sequence.
read	Loads a previously recorded sequence into the sequencer.
readbuf	Copies audio data from a [buffer~] to a Csound table. **arg 1**: Target Csound table number. A negative number replaces an existing table with one whose size equals the amount of data requested in arg 5. If positive, table is not replaced, but will created if it doesn't exist. **arg 2**: Channel to read. 0 means read all channels. If > 0, then read from that channel into the Csound table (treating the table as a single channel table). **arg 3**: The name of the [buffer~] to read from. **arg 4** (optional, default = 0): Read offset time (from beginning of buffer~). If positive, specifies time in seconds. If negative, specifies time in sample frames. **arg 5** (optional, default = 0): Amount of audio data to read. 0 means read in all data or as much data as possible. If positive, specifies time in seconds. If negative, specifies time in sample frames.
recordstart	Puts the sequencer into record mode so that events, MIDI, and control data is recorded.
recordstop	Stops the sequencer from recording.
reset	Stop the current performance and recompile the orchestra.
rewind	Set the score offset time to zero seconds.
rsidx	Read a Csound table value. **arg 1**: table number

	arg 2: table index In response, csound~ will send an **rsidx** message from the Message outlet with this format: **rsidx table_num table_index table_value**
run	Executes a command on your system. If the command you want to run is not in your PATH variable, you must specify the absolute path to the command. File arguments may be relative or absolute paths. Relative paths are relative to the folder that contains the patch that contains [csound~]. When using paths with spaces in them, enclose the path in double-quotes. In Mac OSX, many apps are actually folders. In order to use these apps with the run command, you must specify the path to the actual binary executable which is often in Contents/MacOS/. In Mac OSX, you can use either Max style or POSIX style paths. In Windows, use forward slashes (/) instead of back slashes (\).
sfdir	Set the Csound variables SFDIR, SSDIR, and SADIR. Accepts one argument: an absolute pathname to a directory. It has been provided for compatiblity with Ingalls' csound~ patches.
start (bang)	Compile the csd/orc file specified in the **csound** message (which should have been received before **start**). If compilation is successful, the performance will start once DSP processing is started. If DSP processing is already active, then the csound performance will start immediately. When rendering to a file, DSP processing need not be active. Upon successfull compilation, if DSP is not active, one k-cycle of the performance is automatically processed. This will force all initializations (including events with a start time of zero) to take place. To bypass this feature, set @bypass = 0.
stop	Stop the csound performance and send a bang out the rightmost outlet.
tempo	Set the relative playback speed of recorded events. Accepts one float argument. **tempo 1.0** means playback at original speed. **tempo** messages are not recorded.
write	This message will save a recorded sequence to a binary file. It has one argument: the pathname of a file to save to. Use a .json or .xml extension to save as a JSON or XML file. Otherwise, it's saved as a binary file.
wsidx	Write a Csound table value. **arg 1**: table number **arg 2**: table index **arg 3**: value