

SOClib Manual

Installation	1
Using SOClib	1
Creating SoundObjects	2
Adding Processes	2
The Graphical Interface	2
Parameters	3
SOClib Reference	4

Installation

SOClib is written as a Quark for SuperCollider. As such, it requires that SuperCollider is installed first, from:

<https://supercollider.github.io/download>

Once SuperCollider is installed, the source for *SOClib* must be installed in the appropriate location for SuperCollider to locate it.

- 1) Launch SuperCollider
- 2) File > Open User Support Directory
- 3) If it does not exist, create a folder called *Extensions*
- 4) Copy the *SOClib* folder from:
 - Software/2-SOClibinto the Extensions folder
- 5) Close and restart SuperCollider

Using SOClib

SOClib is a language extension for SuperCollider, so some familiarity it is required. A demonstration file is included alongside this manual to provide example usage. This file can be opened within SuperCollider, and sections of the code (denoted by round-brackets) can be executed by placing the text-cursor within the section and typing `Ctrl+Enter` on Windows or Linux, or `Cmd+Enter` on Mac.

A brief video demonstrating its use is included alongside this manual.

Creating SoundObjects

To create a SoundObject, create an instance of `SOC_SoundObject`, passing the path to a sound file as an argument:

```
~myObj = SOC_SoundObject(Platform.resourceDir+"/"+sounds/allw1k01.wav");
```

Adding Processes

At this point, the `SOC_SoundObject` exists, but we must add at least one `SOC_Process` to hear anything.

The processes (as distributed) are listed in the *SOClib* reference at the end of this manual.

The following code demonstrates adding a process:

```
~myObj = SOC_SoundObject(Platform.resourceDir+"/"+sounds/allw1k01.wav");  
~myObj.add(SOC_Player());
```

Processes are applied in the order in which they are added.

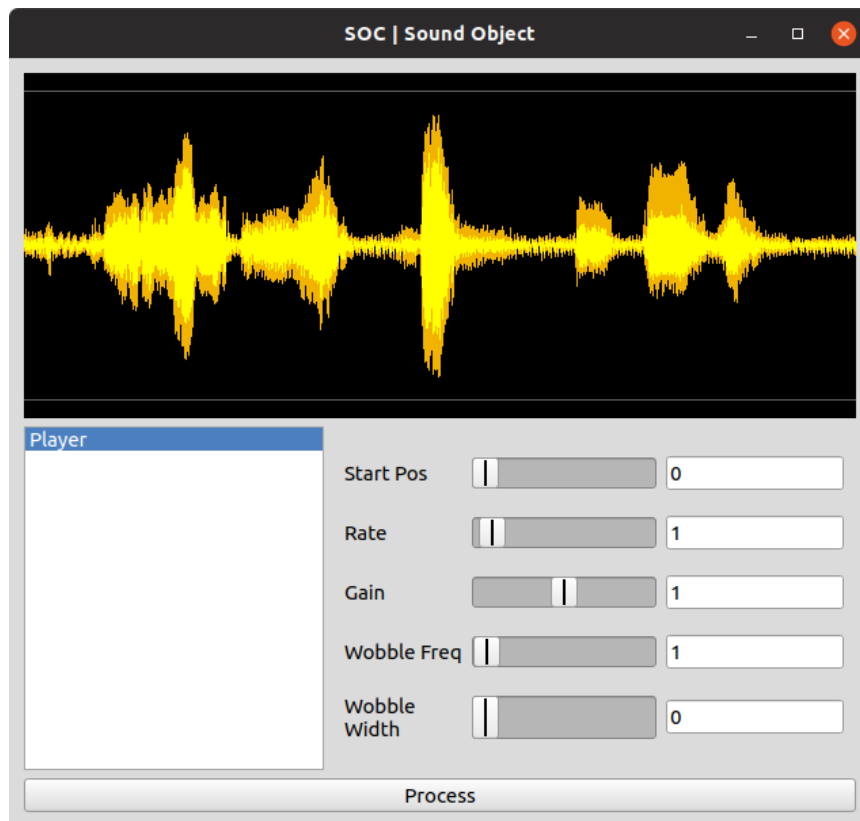
Once a process has been added, it can be accessed again from the SuperCollider language using array indexing, as shown below:

```
~myObj[0]; // returns the first process, SOC_Player
```

The Graphical Interface

If we now request the GUI, using the code below, a window similar to the following will be created:

```
~myObj.gui;
```



The waveform view allows a smaller section of the sound file to be used by clicking and dragging to make a selection. The waveform view can be zoomed in and out with Shift+Right-Mouse-Button (click and drag).

The processes are listed underneath the waveform view, and to their right are the parameter controls. These are, with the exception of silent parameters (see below), all controllable in real-time.

The SoundObject can be requested to process its audio at any time from the language with the `process` method:

```
~myObj.process;
```

Parameters

Each process has a number of parameters. There are two kinds, a standard `SOC_Param` and a `SOC_SilentParam`. These are both the same, except that `SOC_SilentParam` does not send real-time updates. This was required for parameters that cannot change in real-time, for example, the stack size in `SOC_Stack` must be calculated in advance of any processing. Parameters have a minimum, maximum, and default value.

Parameters can be accessed by name through the Process to which they belong. Each parameter has a value method that allows their exact value to be set:

```
~myObj[0].getParam("Rate").value = 2;
```

Parameters also have a variation mechanism, accessible from the SuperCollider language. The request to vary can be set through the process itself, passing the name of the parameter to vary:

```
~myObj[0].vary("Rate");
```

At which point, a new value for the parameter will be generated.

In addition to vary, the restore method offers a way to revert back to the value before variations were requested. This is demonstrated below:

```
~myObj[0].getParam("Rate") = 0.5;
~myObj[0].vary("Rate");
~myObj[0].vary("Rate");
~myObj[0].vary("Rate");
~myObj[0].restore("Rate"); // value = 0.5
```

SOCLib Reference

***indicates SOC_SilentParam**

Process	Parameter	Limits
SOC_Player	Start Pos, Rate, Gain, Wobble Freq, Wobble Width	0-1, 0.01-20, 0-2, 0-100, 0-2
SOC_RingMod	Freq, Width	0-1000, 0-1
SOC_Pan	Position, Random Speed, Random Width	-1-1, 0.01-20, 0-1
SOC_Balance	Position, Random Speed, Random Width	-1-1, 0.01-20, 0-1
SOC_Stack	Stack Size*, Freq Coef*, Freq Offset, Time Offset, Gain	1-800, 0.1-10, 0-100, 0-100, 0-10
SOC_Granular	Grain Rate, Grain Size, Grain Pos, Grain Pos Rand, Playback Rate, Playback Rate Rand	0.01-20, 0.001-2, 0-1, 0-1, 0.01-20, 0-1
SOC_BPF	Freq, rQ, Amp	10-15000, 0.001-2, 0-5
SOC_Comb	Note, Decay, Amp	0-127, 0-2, 0-1
SOC_Envelope	Attack Time, Sustain Time, Sustain Level, Release Time, Gate	0-20, 0-20, 0-1, 0-20, -1-1