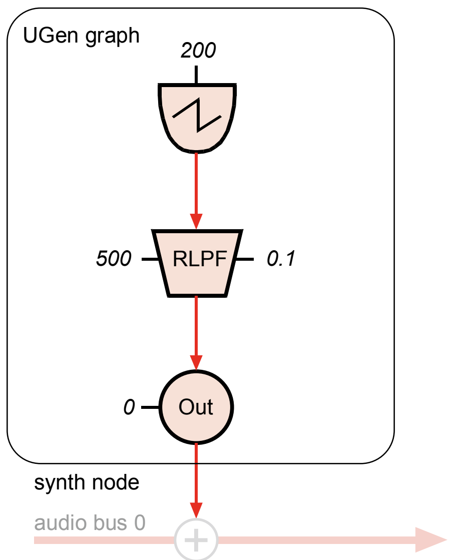
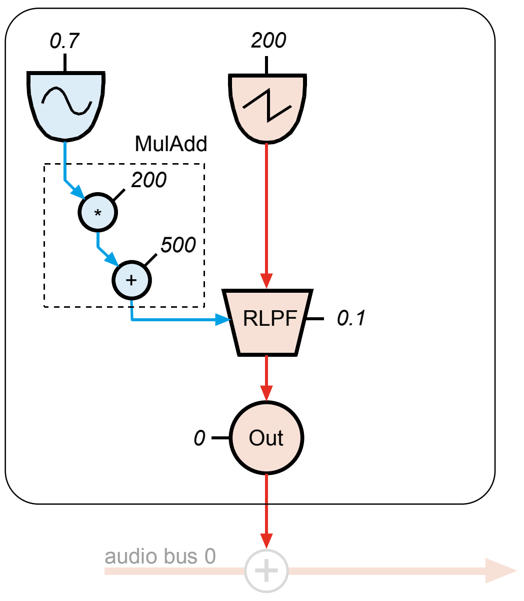


Synth Nodes and UGen graphs

```
(
SynthDef("simple", {
var sig;
sig = Saw.ar(200);
sig = RLPF.ar(sig, 500, 0.1);
Out.ar(0, sig);
}).play;
)
```

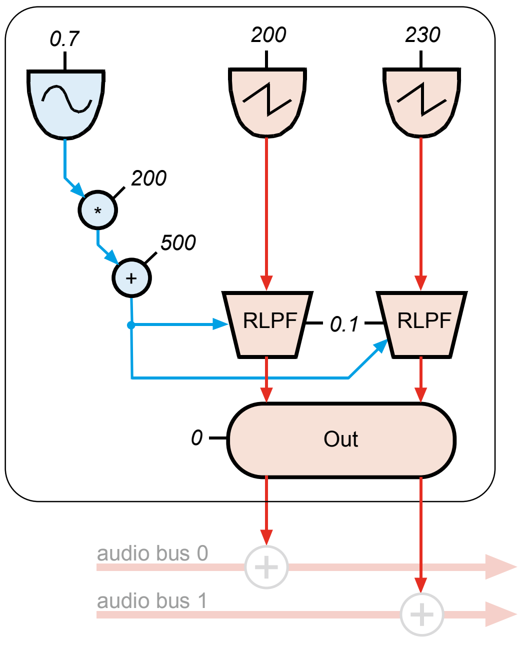


math operators become UGens

```
(
SynthDef("mod", {
var sig, resfreq;
sig = Saw.ar(170 + 30);
resfreq = SinOsc.kr(0.7) * 200 ;
sig = RLPF.ar(sig, 500 + resfreq, 0.1);
Out.ar(0, sig);
}).play;
)
```

these binary ops build BinaryOpUGens

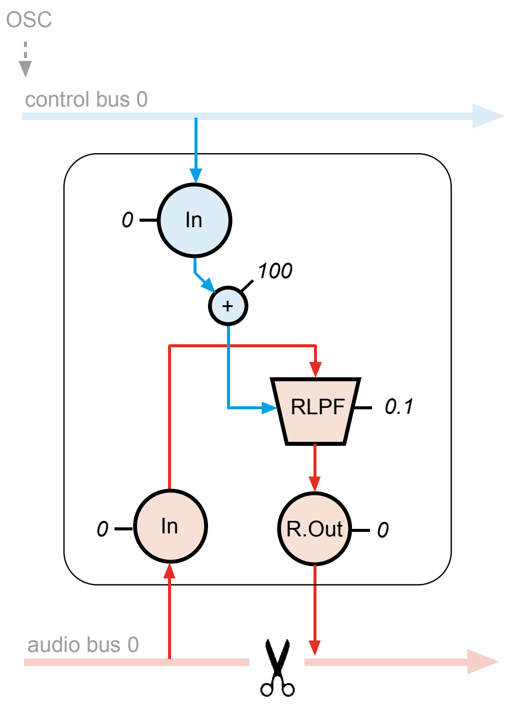
this is just another constant number



multi-channel expansion

```
(
SynthDef("modstereo", {
var sig, resfreq;
sig = Saw.ar([200, 230]);
resfreq = SinOsc.kr(0.7, 0, 200);
sig = RLPF.ar(sig, 500 + resfreq, 0.1);
Out.ar(0, sig);
}).play;
)
```

an Array as UGen argument causes as many UGens as elements are in the Array - thus an Array of UGens



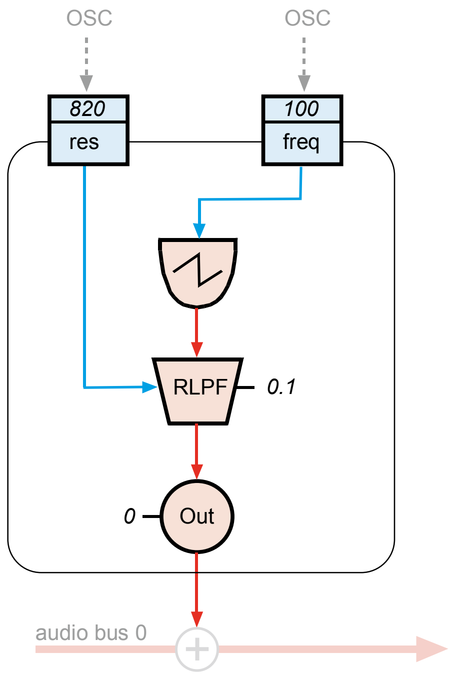
control and audio buses

```
(
SynthDef("bustest", {
var sig, res;
res = In.kr(0) + 100;
sig = In.ar(0);
sig = RLPF.ar(sig, res, 0.1);
ReplaceOut.ar(0, sig);
}).play;
)
```

reads from a control bus

reads from an audio bus

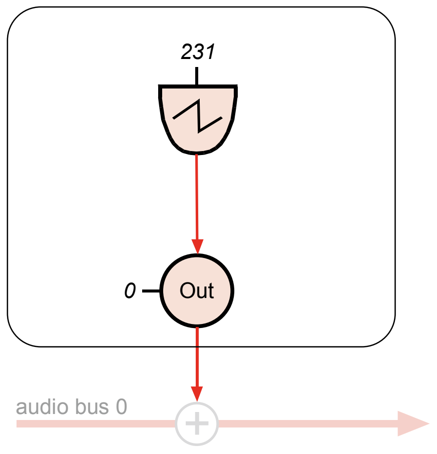
(over)writes to an audio bus



function arguments become Controls

```
(
SynthDef("argtest", {
arg freq=100, res=820;
var sig;
sig = Saw.ar(freq);
sig = RLPF.ar(sig, res, 0.1);
Out.ar(0, sig);
}).play;
)
```

this builds special control-rate UGens, called Controls

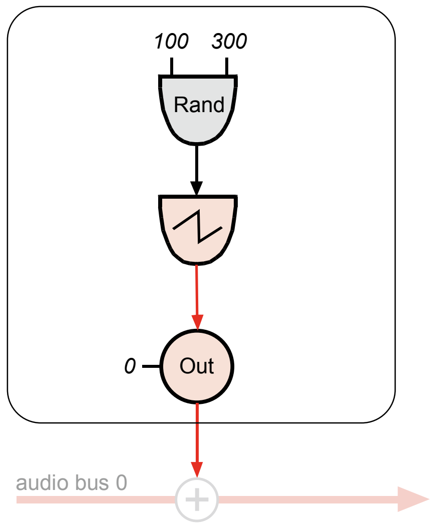


static random numbers

```
(
SynthDef("rrandtest", {
var sig;
sig = Saw.ar(rrand(100, 300));
Out.ar(0, sig);
}).store;
)

Synth("rrandtest");
Synth("rrandtest");
```

this calculates a random number only once in sclang - it appears in the synth only as a constant number!

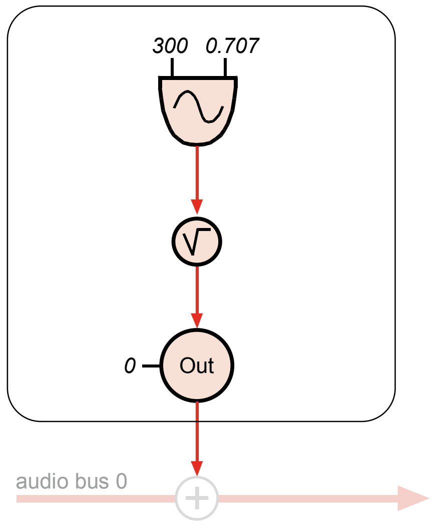


scalar or init rate UGens

```
(
SynthDef("randfreq", {
var sig;
sig = Saw.ar(Rand(100, 300));
Out.ar(0, sig);
}).store;
)

Synth("randfreq");
Synth("randfreq");
```

this is an init-rate UGen - it generates a random number only once when the synth starts



methods and UnaryOpUGens

```
(
SynthDef("sineroot", {
var sig;
sig = SinOsc.ar(300, 0, 0.5.sqrt).sqrt;
Out.ar(0, sig);
}).play;
)
```

this gets calculated only once in sclang - it appears in the synth only as a constant number!

this builds an audio-rate UnaryOpUGen