Map Buss/Def/Func Feature guide

first, lets look at the basic unit. Lets exampine the basic features of MapFunc

MapFunc.new(name, input, output, recipe)
name - a symbol
input - a symbol
output - a symbol
recipe - see MapFunc documentation

make and instance and get a mapped value

```
a = MapFunc.new([[0,0.1,0.25,1],[0.4,0.3,0.3]]);

b = a.map(0.3); //.map returns a the results of a mapped input
```

MapFunc has a local environment used to keep track of everything.

```
a.environment //you can see that the last mapping was noted in the environment
```

you can add data to the environment

```
a.put(\size, 4);

a.environment
```

lets look at MapFunc with a function as the recipe. The input value will be is the first argument in the function.

```
c = MapFunc.new({ |vv| vv * 2 + 0.1.rand2 });
```

MapFunc will generate when given new input or on a clock

given input:

```
c.map(0.5);

c.environment //observe the input and output values
```

on a clock:

```
c.start
c.at(\output) //execute this line repeatedly to see that the? value is being continuuously updated
c.stop
```

By default, the wait time between steps is 0.1 seconds. wait is specified in seconds. Can be an integer, a float or a function that results in either.

c.setWait(0.25); //change the amount of time between updates to a quarter second

c.setWait({rrand(0.25, 1.0)}); //a function is re-evaluated at every step

Using Mapbuss

m = MapBuss.new; // a new instance

Build a MapFunc into the Mapbuss

m.addDef(\test, \asdf, \outout, [[0,0.1,0.25,1],[0.4,0.3,0.3]]);

m.set(\asdf, 0.3); //when a value is set on the Mapbuss, all MapFunc with that value as input are updated

MapFuncs built through the MapBuss are accessible through the MapDef interface.
The outout is written to the MapFunc environment
MapDef(\test).environment

and can be accessed using .at(key)
MapDef(\test).at(\outout);

//another example using an Env. Input should range from 0-1
m.addDef(\testenv, \testenv_in, \testenv_out, Env.triangle(1,1));
m.set(\testenv_in, 1);

//see the output in the MapFunc environment
MapDef(\testenv).environment

//get the output
MapDef(\testenv).at(\testenv_out)

//many to one
//
//only the first input is passed as an argument to the function,
//but all of the environment is available in the function.
//by setting the input to an array, an update to any of the values in the array will cause the MapFunc to update

```
m.addDef(\testFunc, [\one, \asdf], \outsplat, { |vv| vv * ~asdf + 0.1.rand2 });
m.set(\asdf, 0.1); //values for all input are not available so no output is generated
m.set(\one, 10);

MapDef(\testFunc).at(\outsplat);

//another style
m.addDef(\testFunc2, [\one, \asdf], \outsplat, { ~one * ~asdf + 0.1.rand2 });
m.set(\asdf, 0.1);
m.set(\one, 3.5);

MapDef(\testFunc2).at(\outsplat);


//one to many
m.addDef(\testOneMany, \one, [\outone, \outtwo], { |vv| ~outtwo = vv + 0.1.rand2; vv * 2 });
m.set(\one, 3.5);
MapDef.all
MapDef(\testOneMany).environment

MapDef(\testOneMany).at(\outone);
MapDef(\testOneMany).at(\outtwo);

//many to many
remember that the last statement in the function will always set THE FIRST output.

m.addDef(\testManyMany, [\one, \two], [\outred, \outblue], { |vv| vv.postln; ~outblue = ~one *
~two; vv * ~two + 0.1.rand2 });
m.set(\one, 3.6); //why is an error being thrown?  it should just be a warning!  warning happens,
then error?
m.set(\two, 1.25);
MapDef(\testManyMany).environment




//many to many in a ctrlbuss - this should abppear in one of the ctrlbuss guides
//many to many mapping to show that output responders are built for every output in every
mapping.
//in this example, if \in_two wasn't included in the outpu array, the EventDef would  not receive
the data
c = CtrlBuss.new;

c.addDef(\test, [\test], { |self|
        [\start, self.name, ~asdf, ~zxcv, ~in_one, ~in_two].postln
}, { |self|
        [\stop, self.name, ~asdf, ~zxcv, ~in_one, ~in_two].postln
}, [
        [\test_m_to_m, [\asdf, \zxcv], [\in_one, \in_two], {  ~in_two = ~asdf * ~zxcv; ~in_one =
~asdf + ~zxcv; }]
]);
```

```
c.set(\asdf, 0.1);
c.set(\zxcv, 2.0);

c.add(\test);
c.remove(\test);

EventDef(\test).environment
MapDef(\test_m_to_m).environment
MapDef.all
```