CtrlBuss

provides an interface to create and coordinate connections between MatchDef and MapDef.

.addServer(svr)
svr - a server

.addDef(name, pattern, start, stop, map, excl, unless)
name - a symbol, a unique name in the scope of CtrlDef
pattern - an array of symbols
start - a function
stop - a function
map - an Array of Arrays. Arrays are [name, input, output, recipe]
excl - an array of symbols
unless - an array of symbols

.addToggleDef(name, pattern, start, stop, map, excl, unless)
name - a symbol
pattern - an Aarray of Symbols
start - a Function
stop - a Function
map - an Array of Arrays
excl - an array of symbols
unless - an array of symbols

.set(key, value, chan)
key - a Symbol, the name of a MapBuss channel to set
value - an object, typically a number to set on the MapBuss at key
chan - a symbol, optional. set the channel to communicate this data on the network
set a value on the MapBuss

.get(key)
key - a symbol
get a value from the Mapbuss

.signal(...args)
args - varialble
pass a signal to the MatchBuss

.add(key, chan)
key - a symbol

chan - a symbol

.remove(key, chan)
key - a symbol
chan - a symbol

.update(action, item)
action - a symbol (\add, \remove)
item -
pass an item to KeyWin if it exists.

.addMixer(numInput, inputSize, numGroup, groupSize, mainSize, outBus)
numInput - integer, specifiy the number of preamp/microphone input in use
inputSize - integer, the number of audio channels allocated to each input.
numGroup - integer, the number of audio channels for each subgroup
mainSize - integer, the number of audio channels for the main output
outBus -integer, the index of the first output channel on the server.
    commonly 0 when connecting audio interface output 1 as the first channel

.addKeyWin(...args)
args are key value pairs
\func, {}
\chan, symbol
\type, symbol [\press, \toggle]

.addNetwork(...args)
args are key value pairs
\maxLogins, integer
\protocol, symbol [\udp, \tcp]

.onBoot(func)
func - a function
set a function to be executed after scsynth starts. does not start the server.

.waitFoBoot(func)
func - a Function
set a function to execute when the server starts and start the server.

.boot
start the scsynth server

.waitForNetworkBoot(func)
func - a function
boot all the servers on the network and execute this function when all are booted

.addOSCPatt(path, chan)
path - an OSC path, like '\something'
chan - a symbol, the channel incoming data will be assigned to.
an OSC input for control data. messages are interpreted and routed here.
incoming messages can be [\CC, \NOTE_ON, \NOTE_OFF, \STATUS]
cc, note are OSC messages fomatted like -- oscPath, midi channel, note number, value
status can be anything, the incomming message is relayed to the post window.
other messages are formatted like [oscpath, key, value] and are interpreted like (key, value)

.addOSC(chan)
chan - a symbol, the channel incoming data will be assigned to.

incoming osc data that starts with "/OSC" as a root is interpreted and stored on the MapBuss.
data containing values 1.0 or 0.0 are .add or .remove to/from the MatchBuss.

.addMIDI(chan)
.addMIDIIn(chan)
processNoteOn(vv, nn, ch, src, chan)
processNoteOff(vv, nn, ch, src, chan)
processCC(vv, nn, ch, src, chan)
processMIDIArgs(chan)

chan - a symbol, the channel incoming data will be assigned to.

construct MIDIdefs for cc, note and process the incoming data for CtrlBuss.
processing of midi messages updates several data elements on the MapBuss and places keys in the MatchBuss

--noteOn results in
```
  map.set(\midiNoteOn, ("midiNote_c" ++ ch ++ "n" ++ nn).asSymbol, chan)
  map.set(("midiNote_c" ++ ch ++ "n" ++ nn).asSymbol, vv, chan);
  map.set(\midiNoteOnArgs, [\value, vv, \note, nn, \chan, ch, \src, src]);
  match.add(("midiNote_c" ++ ~chan ++ "n" ++ ~note).asSymbol);
```

if noteOn has a value of 0 we treat it like a noteOff
```
  map.set(\midiNoteOff, ("midiNote_c" ++ ch ++ "n" ++ nn).asSymbol, chan);
  map.set(\midiNoteOffArgs, [\value, vv, \note, nn, \chan, ch, \src, src]);
  match.remove(("midiNote_c" ++ ~chan ++ "n" ++ ~note).asSymbol);
```

--noteOff
```
  map.set(\midiNoteOff, ("midiNote_c" ++ ch ++ "n" ++ nn).asSymbol, chan);
```

```
  map.set(\midiNoteOffArgs, [\value, vv, \note, nn, \chan, ch, \src, src]);

cc
  map.set(("midiCC_c" ++ ch ++ "n" ++ nn ).asSymbol, vv, chan);
  map.set(\midiCCArgs, [\value, vv, \num, nn, \chan, ch, \src, src]);
  match.add( ("midiCC_c" ++ ~chan ++ "n" ++ ~num).asSymbol, chan ); //if value == 127
  match.remove( ("midiCC_c" ++ ~chan ++ "n" ++ ~num).asSymbol, chan ); //if value == 0
```

processOther(name, value)
name - a symbol
value - an number
map.set(name, value)
if 0, match.add(key)

.addHID(chan)
chan - a symbol, the channel incoming data will be assigned to.

addHID(chan)
chan - a symbol, the channel incoming data will be assigned to.
open every HID device except apple internals
every incoming channel is wtitten to the MapBuss and MatchBuss (0 is remove, everything else is add)

.addHIDFilter(...args)
args - a symbol or comma separated symbols
ignore HID devices whose name evaluates to any of these symbols
name is constructed like
(\HID ++ "_" ++ dev.info.productName.replace(" ","") ++ "_" ++
args[5].usageName.asString.replace("_","")).asSymbol

.removeHIDFilter(...args)
args - a symbol or comma separated symbols

.at(key)
retreive a value from the local environment

.put(key, value)
put a value into the local environment