TCP Control Socket

Version 6, Anthony Gilley, 2014-09-19

Revision history

Revision/Date	Author	Changes
1 / 2012-10-31	Anthony Gilley	First version
2/ 2014-01-30	Anthony Gilley	Updated for firmware 2.0.1
3/ 2014-02-25	Anthony Gilley	Added code example
4/ 2014-03-11	Anthony Gilley	Updated for firmware 2.1.1 with channel control
5/ 2014-05-20	Anthony Gilley	Added list of useful commands in chapter 6.
6/2014-09-19	Anthony Gilley	Added getEstPowerAndCurrent command and applicability column in chapter 6.

1. Objective

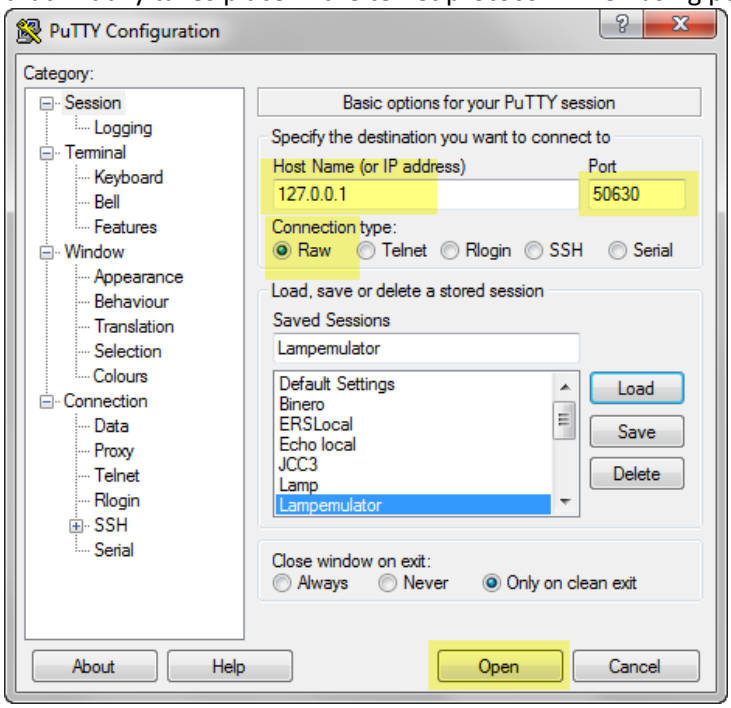
To provide a reference for controlling Heliospectra lamps via the TCP control socket.

2. Basics

The L4/LX60/RX30 lamps can be controlled either via the pages served by the embedded web server or by the TCP control socket. The communication via the control socket is a clear text, request / response and telnet like protocol.

In this document all examples are shown with putty. Putty is a free telnet/ssh client that makes it easy to test different. Putty can be downloaded here
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

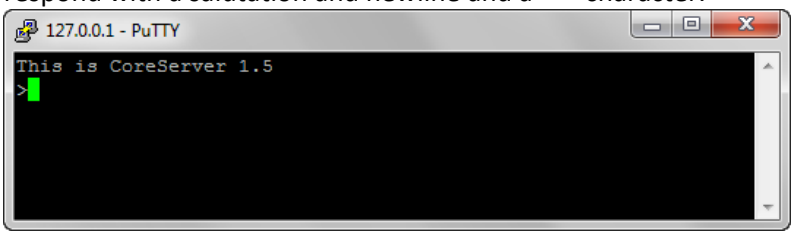
NOTE
Although the control socket uses a telnet like protocol it does not support the handshaking that initially takes place in the telnet protocol. When using putty connect in raw mode.



The host name is the IP address of the lamp you are connecting to. In this document an emulator is used which is why the loopback address 127.0.0.1 is used.

2.1. Connecting and disconnecting

Connect to the socket by opening a TCP socket connecting to port 50630 on the lamp. The lamp will respond with a salutation and newline and a “>” character.



To keep things simple only one connection is allowed at a time to the lamp. If more clients attempt

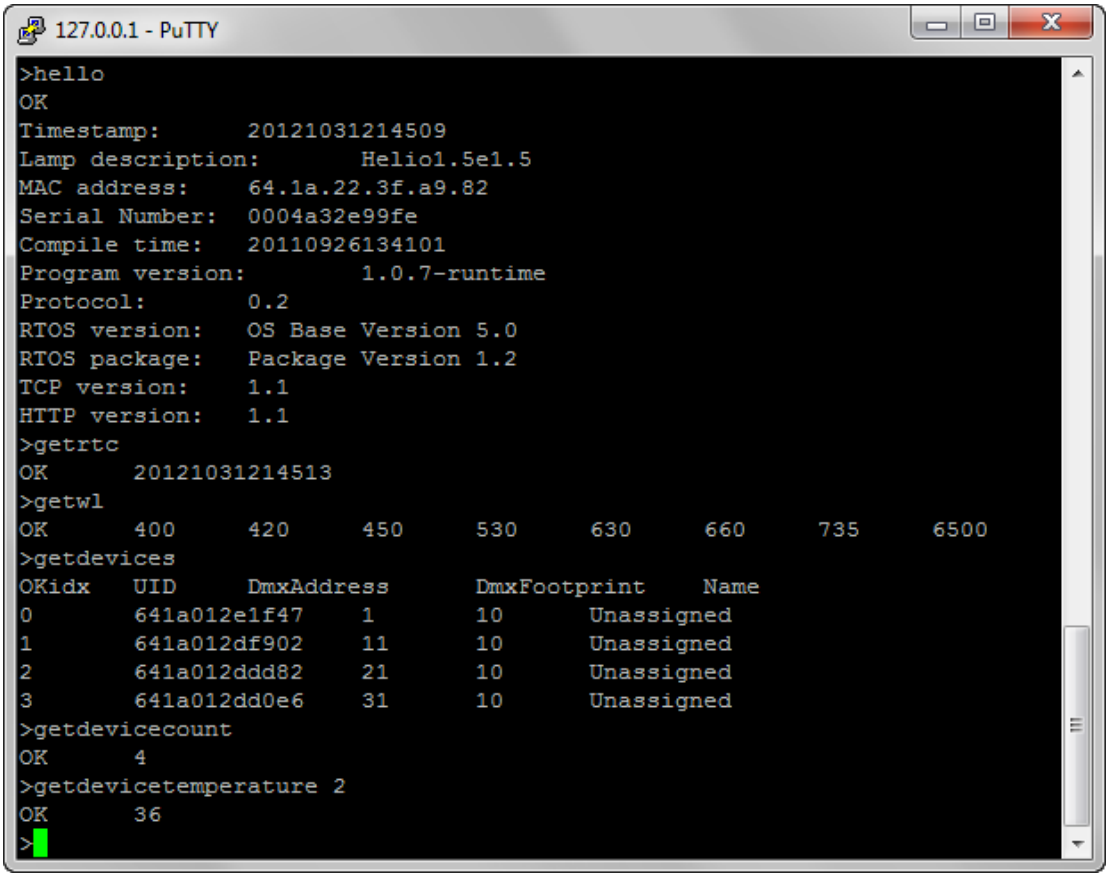
to connect the subsequent clients will receive an error message and the socket will close. The first client will remain connected.

Inactive connections will timeout after 30 seconds by default. Disconnecting from the socket is done by sending the `bye` command to the lamp. The lamp does not send anything in response and just closes the socket.

2.2. Information

The lamp supports the following commands to get information about the configuration and status of the lamp.

hello	Get basic lamp information including firmware version, serial nr, etc.
getrtc	Get the current time in the lamp.
help	List all the commands supported by the lamp with a brief description.
getwl	Get the list of wavelengths the lamp is equipped with. Values above 1000 indicate the color temperature of white LEDs.
getdevices	List low level data regarding the LED Driver cards.
getdevicecount	List the number of drivers in the lamp.
getdevicetemperature *n*	Return the temperature of the associated LED plate.

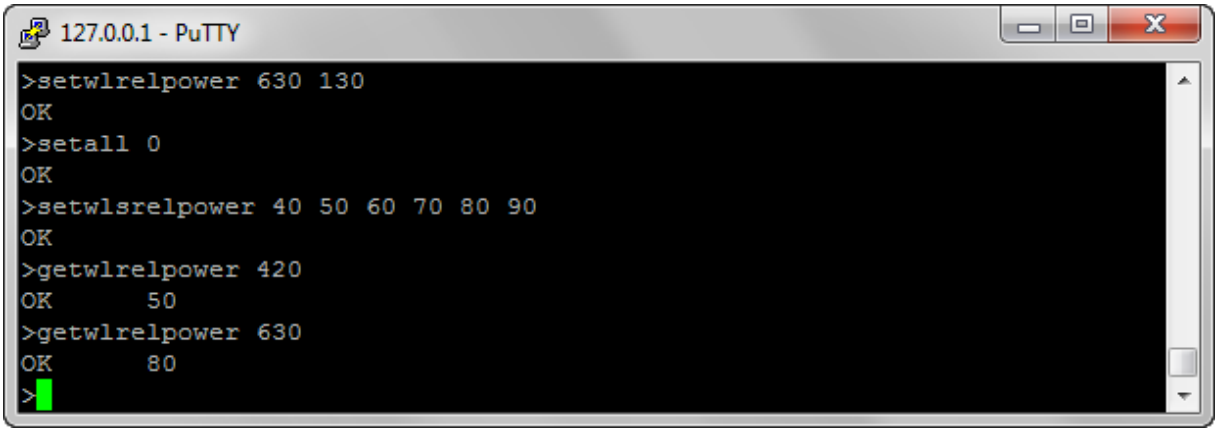


3. Setting and getting lamp intensity per wavelength

Lamp intensity is normally set per wavelength. This can be done either one wavelength at a time, all wavelengths set to the same intensity value or all wavelengths are set to different values.

NOTE In firmware before version 2.0.1 the intensity range in the following commands is 0-255. Starting with version 2.0.1 the range is 0-1000 as described below. The firmware version is given in the response for the hello command on the line starting with "Program version:"

setwlrelpower *wl [0-1000]*	Set the intensity of a single wavelength. The wavelengths must be among the ones returned by the getwl command. The intensity is a value from 0 to 1000. *setwlrelpower 400 143* will set the 400nm LEDs to intensity level 143.
setall *[0-1000]*	Set the intensity of all wavelengths. The intensity is a value from 0 to 1000. *setall 0* sets all wavelengths to 0 intensity which is equivalent to turning the wavelengths off.
setwlsrelpower *a b c d …*	Sets multiple wavelength intensity levels at a time. The intensities are assigned to the wavelengths in the order they are returned by the getwl command. If getwl returns *400 420 450 …* then setwlsrelpower 40 50 60 … will assign intensities 40, 50 and 60 to wavelengths 400, 420 and 450 respectively.
getwlrelpower *wl*	Read the current intensity setting of a specific wavelength. *getwlrelpower 420* would return 50 after the previous command.
getAllRelPower	Returns a list of the intensity settings of all wavelengths as returned by getWl.

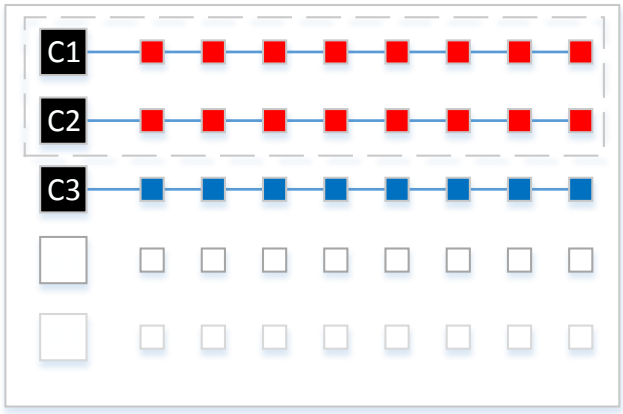


4. Setting and getting lamp intensity per channel (fw 2.1.1 and higher)

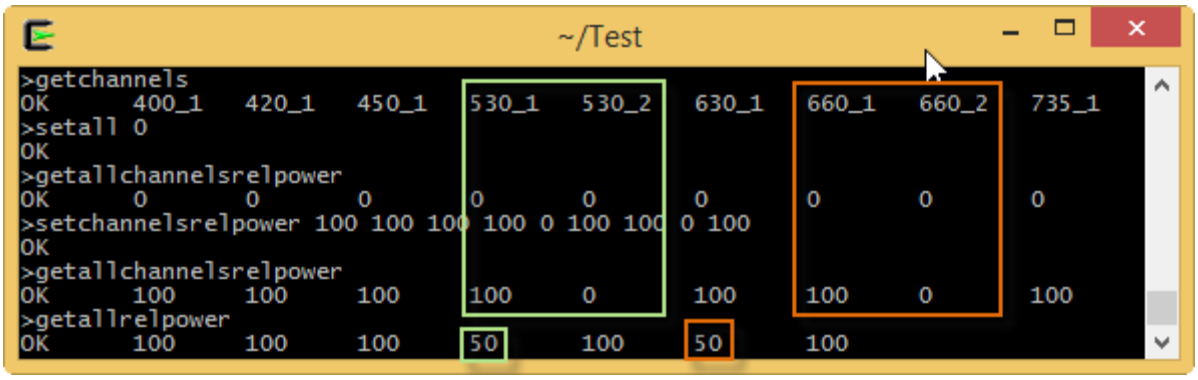
Starting with fw version 2.1.1 each LED channel can be controlled individually. As described in 3 intensities are normally assigned to all LEDs of the same wavelength. However, to increase the granularity of control one can also control an individual channel.

One "wavelength" is split into more than one channel if the number of LEDs used is greater than what a single driver can power (see illustration below).

Drivers C1 and C2 both drive red LEDs (eg 660nm). Both C1 and C2 can be controlled in unison with the *setwlrelpower* command.
Or they can be controlled separately with the *setChannelsRelPower* command.



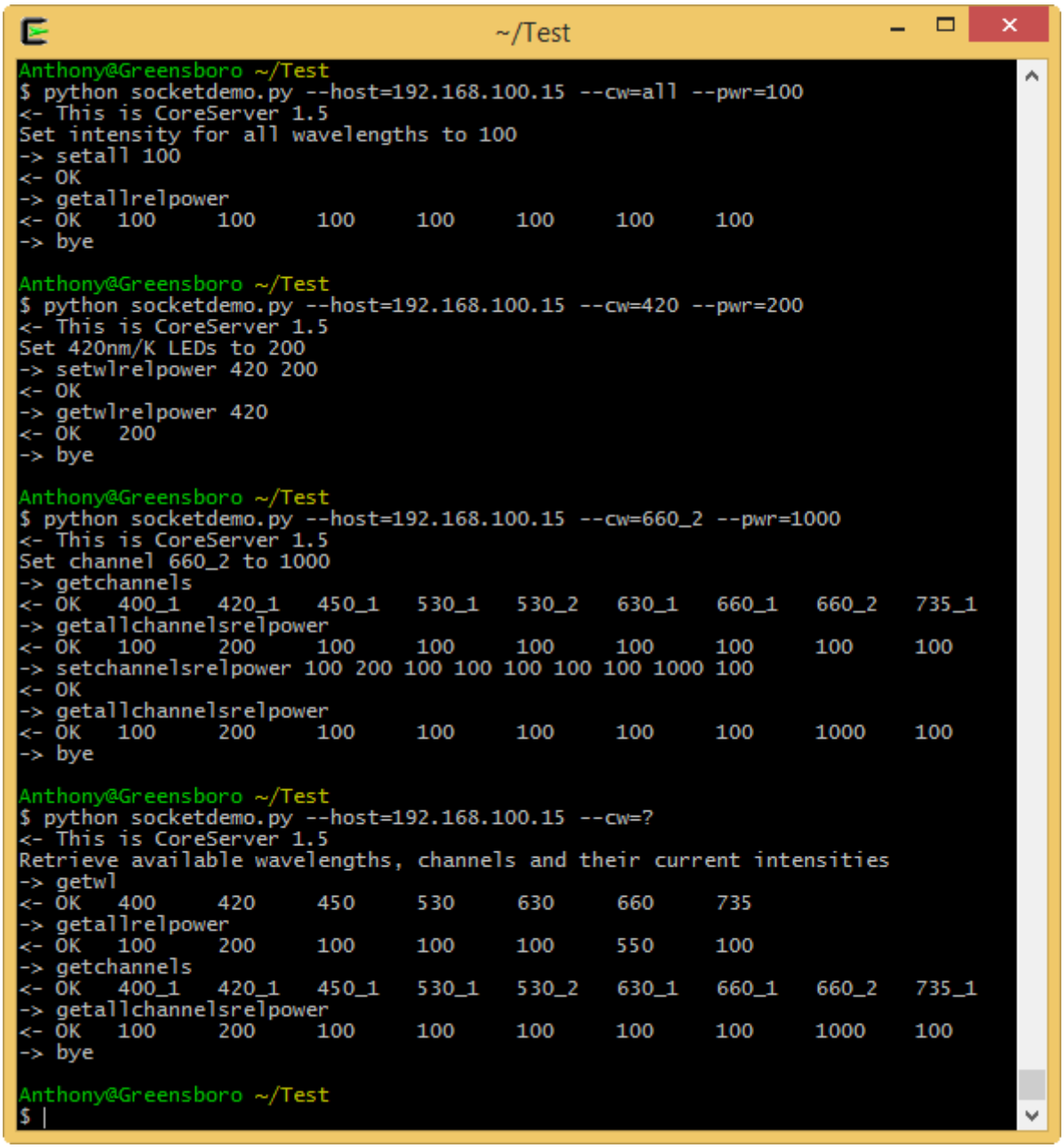
getChannels	Get the list of channels the lamp is equipped with. The response is a tab separated list of channels. Each channel is labelled with the wavelength followed by an index value. If several channels have the same LED type (e.g. 450nm) the channels will be labelled 450_1, 450_2, …
getAllChannelsRelPower	Returns the list of intensity values for all channels [0-1000]. The intensities are listed in the same order as the response to getChannls.
setChannelsRelPower *a b c …*	Sets multiple channel intensity levels at once. The intensities are assigned to the wavelengths in the order they are returned by the getChannels command. If getChannels returns *400_1 420_1 450_1 …* then setChannelsRelPower 40 50 60 … will assign intensities 40, 50 and 60 to channels 400_1, 420_1 and 450_1 respectively.
getwlrelpower *wl*	Read the current intensity setting of a specific wavelength. *getwlrelpower 420* would return 50 after the previous command.



Note that by setting channels 530_1 and 530_2 to 100 and 0 the overall intensity for the 530 wavelength is 50 (green highlight). The equivalent is done for the two 660 channels (orange highlight).

5. Code example

The following is a simple script written in Python 2.7 showing how to interact with a lamp over the TCP socket. The script allows the following type of interaction



The source for the script is listed on the next page.

```
# socketdemo.py - Heliospectra, Anthony Gilley, 20140225
from telnetlib import Telnet
from sys import *
from getopt import *
eol = "\r\n"

def connect(host):
    # Connect to the lamp on port 50630
    tn = Telnet(host, 50630)
    # Get and discard the first salutation
    readresponse(tn)
    return tn

def disconnect(tn):
    # Disconnect from the socket.
    command = "bye"
    sendcommand(tn, command, False)
    tn.close()

def setwlrelpwr(tn, wl, pwr):
    # Set the power for the given wavelength
    command = "setwlrelpower " + wl + " " + pwr
    sendcommand(tn, command)
    # Read back the power value
    command = "getwlrelpower " + wl
    sendcommand(tn, command)

def setchannelrelpwr(tn, ch, pwr):
    # Set the power for the given channel
    command = "getchannels"
    chls = sendcommand(tn, command).split()
    intensities = sendcommand(tn, "getallchannelsrelpower").split()
    if ch in chls:
        intensities[chls.index(ch)] = pwr
        command = "setchannelsrelpower " + ' '.join(intensities)
        sendcommand(tn, command)
        sendcommand(tn, "getallchannelsrelpower")
    else:
        print ch + " not found in lamp"

def setall(tn, pwr):
    # Set the power for the given wavelength
    command = "setall " + pwr
    sendcommand(tn, command)
    # Read back the power values
    command = "getallrelpower"
    sendcommand(tn, command)

def getwl(tn):
    # Get the wavelengths
    command = "getwl"
    sendcommand(tn, command)
    # Read back the power values
    command = "getallrelpower"
    sendcommand(tn, command)

def getchannels(tn):
    # Get the wavelengths
    command = "getchannels"
    sendcommand(tn, command)
    # Read back the power values
    command = "getallchannelsrelpower"
    sendcommand(tn, command)

def sendcommand(tn, command, read = True):
    r=''
    print "-> " + command
    tn.write(command + eol)
    if(read):
        r=readresponse(tn)
        if r.startswith('OK'):
            r = r[3:].strip()
    return r

def readresponse(tn):
    r = tn.read_until(">")[:-3]
    print "<- " + r
    return r

# Decode the command line arguments and issue the command
if len(argv)>0:
    opts, args = getopt(argv[1:], '', ['cw=','pwr=','host='])
    for opt, arg in opts:
        if opt=='--cw':
            cw = arg
        if opt=='--pwr':
            pwr = arg
        if opt=='--host':
            host = arg
tn = connect(host)
if(cw=='all'):
    print "Set intensity for all wavelengths to " + pwr
    setall(tn, pwr)
elif(cw=='?'):
    print "Retrieve wavelengths, channels and intensities"
    getwl(tn)
    getchannels(tn)
elif('_' in cw):
    print "Set channel " + cw + " to " + pwr
    setchannelrelpwr(tn, cw, pwr)
else:
    print "Set " + cw + "nm/K LEDs to " + pwr
    setwlrelpwr(tn, cw, pwr)
disconnect(tn)
```

6. List of useful commands

Command	Explanation	Applicability	Example
bye	Close the connection with the lamp.	L4, LX60, RX30	bye
getDeviceCount	The number of connected LED devices in the lamp.	L4, LX60, RX30	getDeviceCount
getDevices	The list of connected devices with status information.	L4	getDevices
getWl	The list of available wavelengths.	L4, LX60, RX30	getWl
getWlsMaxPower	Max electrical power value (deciwatts) for available wavelengths in getWl order.	L4, LX60, RX30	getWlsMaxPower
getDeviceTemperature	The temperature of a LED device.	L4, LX60, RX30	getDeviceTemperature 1
getDeviceInfo	Detailed information about a specific LED device.	L4	getDeviceInfo 1
getDeviceInputVoltage	Input voltage of a device.	L4	getDeviceInputVoltage 1
getEstPowerAndCurrent	Estimated power and current consumption	LX60, RX30	getEstPowerAndCurrent
getWlRelPower	Returns relative setting [0-1000] of a wavelength.	L4, LX60, RX30	getWlRelPower 660
getAllRelPower	Returns relative setting [0-1000] of all wavelengths.	L4, LX60, RX30	getAllRelPower
setWlsRelPower	Provide settings for multiple wavelengths [0-1000]*n. Provided in getWL order.	L4, LX60, RX30	setWlsRelPower 850 200
setWlRelPower	Set wavelength to relative value [0-1000]. none	L4, LX60, RX30	setWlRelPower 660 300
setAll	Set all wavelengths to the same relative value [0-1000]. none	L4, LX60, RX30	setAll 1000
setTimeout	Set the connection timeout value [10-600] seconds. The connection will close after an inactive timeout period.	L4, LX60, RX30	setTimeout 60
getTimeout	Returns the connection timeout value.	L4, LX60, RX30	getTimeout
help	The entire list of commands.	L4, LX60, RX30	help
hello	Lamp identification and firmware information.	L4, LX60, RX30	hello
getUptime	The uptime of the lamp in seconds.	L4, LX60, RX30	getUptime
getmac	Returns the MAC address of the lamp.	L4, LX60, RX30	getmac
setBeep	Turn the beeping on and off.	L4	setBeep 0
getBeep	Returns the status of the beep.	L4	getBeep
mute	Mutes audible warning signal until next reboot.	L4	mute
reboot	Restart the CPU in the lamp.	L4, LX60, RX30	reboot