Macintosh example
for switching on and off
the KMTronic USB One Relay

*(writen by **madu**)*

I recently bought a KMtronic USB 8 Channel Relay Board for a private project.
The last few days (and nights...) I did lots of reseaching to find nice and working ways to control the device with a Macintosh, especially for comfortable solutions for automated processes. Since there isn't much information about this topic here in this forum yet, I thought I'll share my experiences here. Someone might look for these things and find it useful.

Q: What will I find here?
A: Infos about the USB 8 Relay board used with a Macintosh

Some basics: Drivers, device serial number, commands, first test
Tools: Some Mac tools I found that can communicate with the board
Automation: Controlling the relays through AppleScript, make a GUI

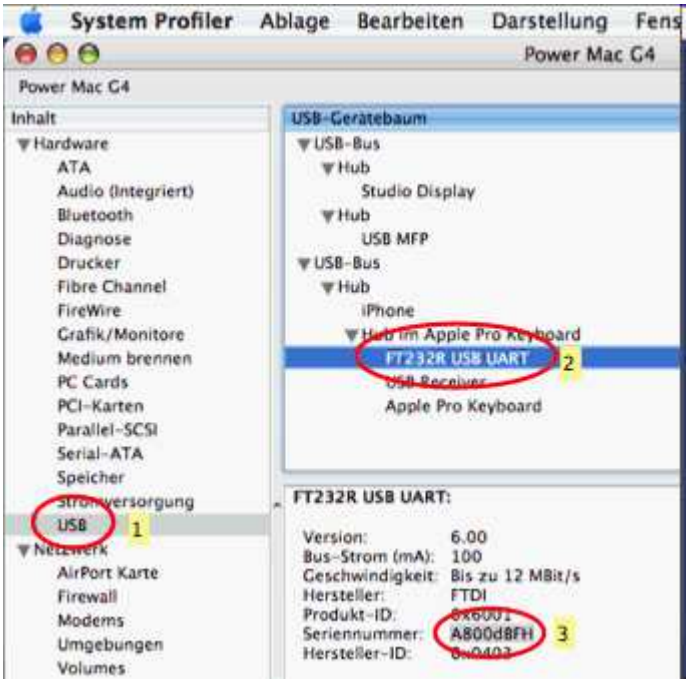
Be aware:
I am not a professional programmer. I have a good basic knowledge in computing and networking in general, specially on the Mac, but I am pretty bad with shell-scripts stuff. Terminal is not my favorite program. I just almost never really need it...
I have some basic knowledge in AppleScript and good skills in perl, but I am far apart of being a pro.
That all said, there is a good chance that many of the hints and tricks I tell here can be done much easier. There might probably be better, nicer, more stable or more powerful solutions. Also, my code may be buggy, it may let your TV explode and your car to refuse running. In other words, if my code pushes your life in a crises, it's not my fault. Use it at your own risk. ;)

Basic Stuff:

Instal drivers
First, you need to instal the Mac drivers.
On KMtronics download page take the first link to "FTDI drivers" and find the Mac OSX driver for your system.

Find your new device
When drivers are installed and the usb-board is plugged to your Mac, you can find your new device in the System Profiler.
(to open System Profiler click your Apple-Menu and select "about this Mac". In the window that opens, klick on "more information")

Select USB from the hardware list (1), and you should see your new device in one of your USB-Bus (2).
>> Write down (or copy&paste to somewhere) the 8 digit device serial number (3), you will need it later.



Of course, you can find your device through the terminal as well.
Open Terminal.app and type:

Code:
```
cd /dev/
ls cu.*
```
You get your serial devices listed.
One of them should be named "cu.usbserial-xxxxxxxx" where xxxxxxxx is the device serial number.

Commands
The commands can be found here in the forum and in the boards manual.
Commands are mostly noted in HEX format and look like this:
FF 01 00
where **01** indicates the relay (01 - 08) and **00** is the status (00 = off / 01 = on)
The same notation in decimal format would be: **255 1 0**

First test
Connect the relay-board and the usb-control-board with the flat cable.
Connect a 12V DC source to your relay-board. Without external power it will do nothing!
Connect the USB-Cable to your Mac.

Open Terminal.app and type: (replace xxx... with your device serial number)

Code:
```
echo -e '\xFF\x01\x01' > /dev/cu.usbserial-xxxxxxxx
```

Hit enter and Relay 1 should switch on.

Now type:
Code:
```
echo -e '\xFF\x01\x00' > /dev/cu.usbserial-xxxxxxxx
```

Hit enter and Relay 1 should switch off.

The \x notation is needed to tell the shell that this is hex-code.
Now you can play with the commands from above.

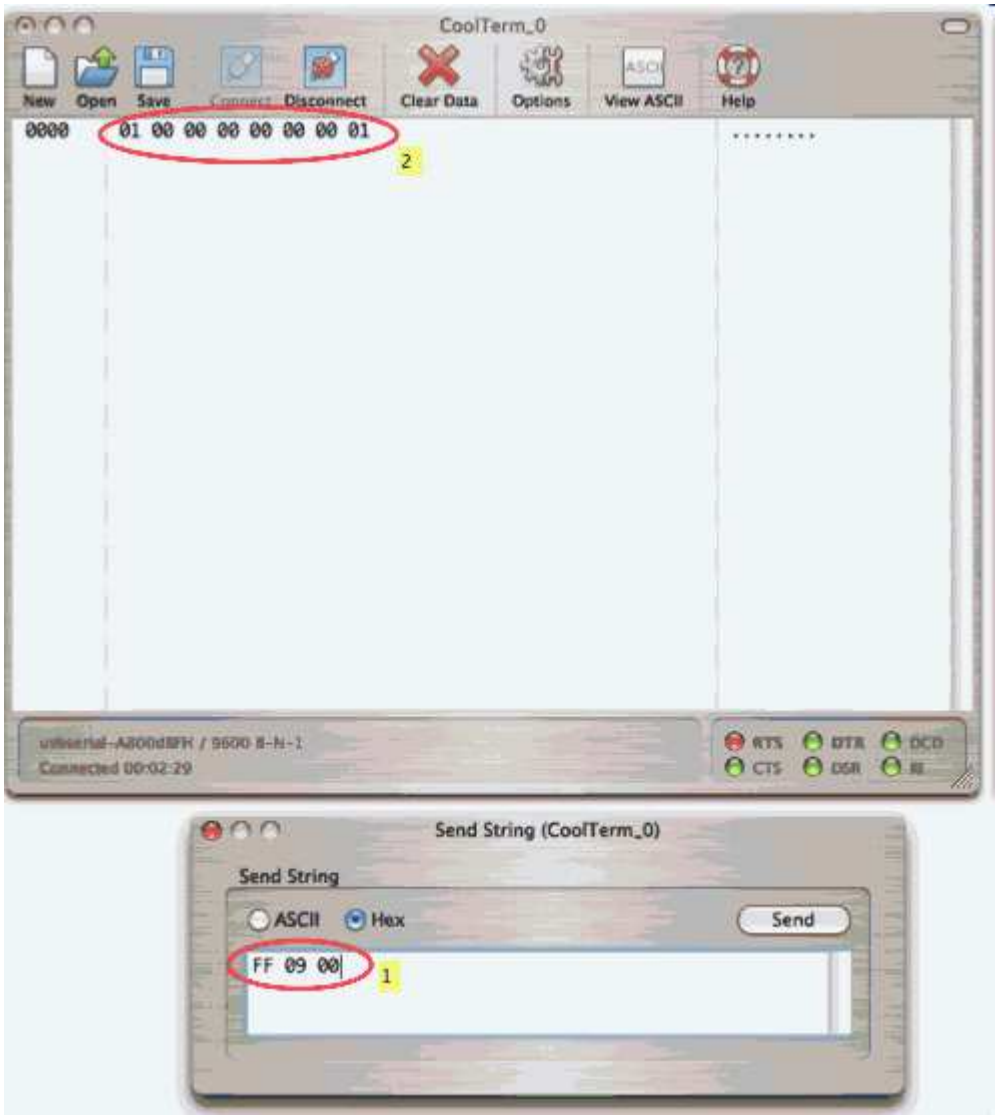
Unfortunately, I could not find out how to get status respond from the board through the terminal.
The command would be FF 09 00 (HEX) or 255 9 0 (DEC)
It is probably possible some way, I just don't know how...
I found references that it may be possible with "screen", which is part of Mac OSX, but I didn't find out how to work with it. Type "man screen" in your terminal to get the manual.

Tools:

Since I did not manage to get response from the relay-board with Terminal and this is essential for automated processes, I was looking for other ways.
These are all tools for Mac that can communicate to serial ports. I found them with Google when searching for Macintosh serial communication tools.
Although, most of them didn't work (for me!).
This does not mean that the really don't work. Most likely I am just too stupid to use them in the right way! You might have more luck, so check them out yourself.

ZTerm, Serial Tools, goSerial, QuickTerm
These are all tools to communicate with serial ports.
While they all can connect to the Relay-Board, I couldn't do anything useful with them, but you might want to try yourself.
ZTerm | Serial Tools | goSerial | QuickTerm

CoolTerm
Cool Term for Mac (also available for WIN) was the one and only of all the tools I found, which I could send commands AND receive status response with!
It has a possibility to send HEX-code.
Actually, I can control everything of the Relay-Board with this tool.



In this example I first turned relay 1 + 8 on (commands FF 01 01 + FF 08 01).
Then I ask for the current state with command FF 09 00 (1) and get the response in format 01 00 00 00 00 00 00 01 (2).

Well, for doing automated things, I need to be able to work with the output somehow.
None of these tools helped me with this.
This brought me back to AppleScript.

Relay-Board & AppleScript:

Of course it's easy to just wrap the shell-commands from above in an AppleScript.
If you just need to switch on and off your relays out of a script, this can be done with no big effort:

Code:
```
set testString to "echo -e '\\xFF\\x01\\x01' > /dev/cu.usbserial-xxxxxxxx"
do shell script testString
```

Note the double-backslash needed to mask (\\x) !
But, still, you can not ask for the current status of the relays this way.

SerialPort X
SerialPort X is a ScriptingAddition to access standard serial ports from AppleScript.
Its syntax include:

```
serialport list
serialport open
serialport write
serialport read
serialport close
```

I found and tried SerialPort X before, but I always stucked with sending the commands in the right format.
After lots of try&error I finally found the solution.
We need to use the decimal notation and convert it to string of characters:

Code:
```
-- open the serial port and set a reference:
set portRef to serialport open "/dev/cu.usbserial-xxxxxxxx"
delay .1
-- make a sendable string out of the decimal command, in this example turn relay 1 on:
set myCmd to MakeString({255, 1, 1})
-- send this to the board:
serialport write myCmd to portRef
-- close the port:
serialport close portRef

-- This function takes a list of byte values and
-- converts them into a string of characters
on MakeString(theBytes)
  set thestr to ""
  repeat with i from 1 to length of theBytes
    set thestr to thestr & (ASCII character (item i of theBytes))
  end repeat
  return thestr
end MakeString
```

Bingo! This works perfect.
But what about status respond?
No problem. The same way we convert the commands, we have to convert the "answer" of the board back to something we can read and work with.

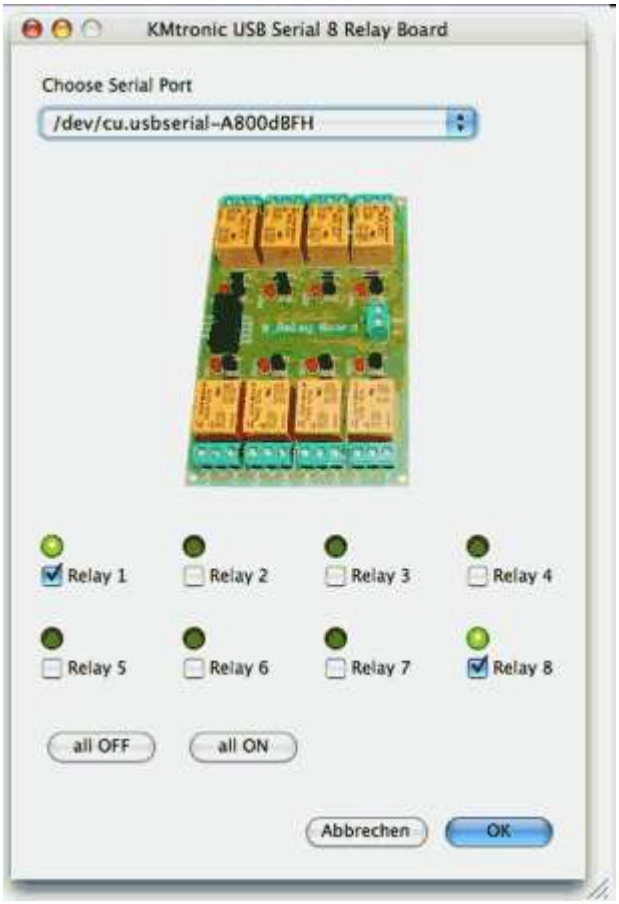
Code:

```
-- as before, switch relay 1 on:
set portRef to serialport open "/dev/cu.usbserial-xxxxxxxx"
delay .1
set myCmd to MakeString({255, 1, 1})
serialport write myCmd to portRef

-- now ask for the status:
set myCmd to MakeString({255, 9, 0})
serialport write myCmd to portRef
delay .1
-- read the answer:
set reply to serialport read portRef
-- and translate the answer to something readable:
set output to TranslateString(reply)

serialport close portRef
return output

-- convert output
on TranslateString(theReply)
   set thestr to {}
   repeat with i from 1 to length of theReply
      set thestr to thestr & (ASCII number (item i of theReply))
   end repeat
   return thestr
end TranslateString

on MakeString(theBytes)
   set thestr to ""
   repeat with i from 1 to length of theBytes
      set thestr to thestr & (ASCII character (item i of theBytes))
   end repeat
   return thestr
end MakeString
```

This will turn relais 1 on, then ask for status, and give back a list that looks like this:
{1,0,0,0,0,0,0,0}
telling us that relay 1 is on and 2-8 are off.
We can now do anything we want with the list returned and build our automation script around the codes.
Hurray!

A GUI would be cool
Well, the world is devided in terminal geeks and GUI-guys.
While I have at least some basic skills in coding, many people do not at all or prefer a grafical interface over typing cryptical commands.
I also would love to have some graphical interface to control my relay-board.
This brought me to Pashua.
Pashua is a tool for creating native Aqua dialog windows from all kind of scripting languages.
With Pashua, it is very easy to create a GUI-like interface out of AppleScript (or any other language like Perl, PHP, Python, Rexx, Ruby, shell scripts and Tcl). All coding is done in AppleScript (or the language you prefer) while the Pashua application just lies in your application-folder and does its work (create the dialog window) in the background for you.
When you save your AppleScript as application, you will get a double-clickable item that looks and acts like an application - which is just what I and many others want.

Below you see what I've done just with AppleScript with SerialPort X and Pashua:



When I start this, it will ask the relay-board for the current state of the relays and show it in the dialog window with green LED-graphics and checked checkboxes. Now I can check any relay on and off using the appropriate checkbox, and hit OK when I'm done. The window will disappear, the script will switch the relays, then it will ask the status again and pop back on with the new status of relays displayd. 8-)

<<EOF>>

I hope that all this information here might be useful for some people.
Enjoy your Relay-Board with your Mac ;)