Iseg CC2x power supply integration into Entangle

Dear entangle users,

In the last months I continued work on the integration of the Iseg CC2x power supply into Entangle .

The main goal was to have a reliable and stable integration without limiting the flexibility of the power supply voltage ramping capabilities.

1. Installation steps (for linux x86_64):

   A. Install Entangle on the target machine.

   B. Download entangle-install-isegCC2x . This is a Linux binary compatible with all major Linuxes , tested among others on Centos7 and Ubuntu20 .

   C. Change permission to executable with [chmod +x entangle-install-isegCC2x] and run [./entangle-install-isegCC2x] . This will install the entangle iseg support and all needed libraries .

   D. Assuming defaults you will find examples for the .res configuration files in /etc/entangle/iseg/CC2xlib/example . For example Erwin-2seg.res or Erwin-small.res . Please open this file, edit the parameters as needed and copy the file to /etc/entangle.

2. Configuration of the .res file, basics.

   For flexible multichannel voltage ramping we use

```
type = "iseg.CC2x.IntelligentPowerSupply"
```

   Channels can be grouped together in Groups in the form

```
groups="""
{
 "GROUP": [
   {"Anodes": { "CHANNEL": ["0_1_0","0_1_1","0_1_2"]  ,"OPERATINGSTYLE": "normal" }}
]
}
"""
```

   This groups together channels which are defined by line_address_channel (to my understanding a better naming would be crate_slot_channel). In this example channels 0, 1, and 2 from crate 0 and slot 1 are grouped together into a GROUP called “Anodes”.

   Each group can have an optional OPERATINGSTYLE . This is just a set of commands for the Iseg CC2x grouped together.

   Commands for the Iseg modules are device specific and are specified in the url = 'http://' + address + '/api/getItemsInfo' where address is the ip address of the Iseg module.

Dipl. Phys. Andreas Langhoff , 089-289-12132 andreas.langhoff@frm2.tum.de

The entangle-isegCC2x interface will download this file upon first connection.

Typical commands are:

```
operatingstyle="""
{
    "Control.clearAll" : 1,
       "Control.currentSet" : 1.5,
    "Setup.delayedTripTime" : 1500,
       "Setup.delayedTripAction" : 2
}
"""
```

“Control.clearAll” resets all errors.
“Control.currentSet” sets the maximum current , above and above “SetupdelayedTripTime” the “Setup.delayedTripAction” will be set. More details to all these commands are found in the getItemsInfo.xml mentioned before.

3. Configuring transitions:

Transitions are changes of states of groups of channels. An example would be the request to ramp up the “Window” GROUP voltage, wait until ramped up, then ramp up the “Anodes” GROUP voltage and “CatodeStripes” group voltage in parallel and wait until finished.

Below the definition of the transition The transition for above action would be defined such

```
transitions="""
{
"TRANSITION" :[
{"goOn":  [
              {"GROUP":["Window"],"Control.clearAll": [1]},
                        {"GROUP":["Anodes"],"Control.clearAll": [1,1,1]},
                        {"GROUP":["CathodeStripes"],"Control.clearAll": [1,1]},
              {"GROUP":["Window"],"Control.voltageSet": [-1000]},
              {"GROUP":["Window"],"Control.on": [1] },
              {"GROUP":["Window"],"Status.ramping": [0] },
              {"GROUP":["Anodes"],"Control.voltageSet": [2075,2100,2085]},
              {"GROUP":["Anodes"],"Control.on": [1,1,1] },
              {"GROUP":["CathodeStripes"],"Control.voltageSet": [75,80]},
              {"GROUP":["CathodeStripes"],"Control.on": [1,1]},
              {"GROUP":["CathodeStripes"],"Status.ramping": [0,0]},
              {"GROUP":["Anodes"],"Status.ramping": [0,0,0] }
            ]
},
]
}
"""
```

The command “Status.ramping” returns either 1 if a channel in a GROUP is still ramping or it returns 0 once ramping has finished. Waiting for ramping finished can hence be monitored .

4. Hardlimits.py:

A user request was a safety feature so that it is not possible to exceed certain

voltage or current limits by error.

```
class HardLimits:
    unitTime = 'ms'
    unitCurrent = 'uA'
    maxTripCurrent = 10
    maxVoltage = 90
    tripEventAllModulesOff = True
```

The file hardlimits.py (installed in the iseg/cc2xlib contains maximum settings for a few parameters. Any value above will not be transmitted but truncated to the maximum allowed value. (Note: maxVoltage is set to 90 V only because I do not have permanent access to the device and 90 V dc is safe for humans in the unlikely case someone would touch the voltage output.)

5. Trip current behavior: CC2x events of type:
"Error.currentLimitExceeded", "Event.currentTrip", "Event.arc" and "Error.arc"

   will set the device into ALARM state. The current module will be switched off and if Hardlimits.tripEventsAllModulesOff == True then all modules will be switched off.

   A device reset will clear the ALARM state.

6. Nicos Interface: Work is still ongoing .
   Currently the best solution is to use the QuangoPlus . Windows users can download QuangoPlus_for_Windows
   With QuangoPlus you can control transitions and monitor all voltages.

7. Source code can be viewed on https://github.com/zweistein22/iseg

8. Iseg firmware problems and bug fixes:

   During long term tests 2 problems in the firmware were found , but it was possible by software to eliminate the effects by applying the following rules:

   A. All commands have to be send with the respective Unit (Units are defined in Hardlimits.py.

   B. After initial device connection the software will wait > 15 seconds until initial error messages (of unclear origin) were sent by the device. Only then the parameters for all channels including trip current settings can be send successfully to the device.