CamillaDSP / Squeezelite /pCP DSP_Engine

Text in blue = Script's and outputs, strings etc… on pCP (RaspberryPi)
Text in red = Commands giving in terminal / cmd. Line. [Putty or Linux terminal]

Extend filesystem as instructed for piCorePlayer. [Main page, Resize FS]
SSH into the pCP/RPI ssh tc@192.168.1.95 (with the right ip number ofcause) or with Putty
Default password is: piCore

Install needed editor:
tce-load -w -i nano (RPI downloads and installs nano now)
Nano is an easy texteditor.
Quick use : [ctrl] + o = save
[ctrl] + x = exit

Edit a file:
nano /opt/bootlocal.sh
Insert the 3 lines as seen In bold below
Then save [ctrl] + o and exit nano [ctrl] + x

```
#!/bin/sh
# put other system startup commands here

GREEN="$(echo -e '\033[1;32m')"

echo
echo "${GREEN}Running bootlocal.sh..."

modprobe snd_aloop
#/home/tc/CamillaDSP.sh start
sleep 1

#pCPstart------
/usr/local/etc/init.d/pcp_startup.sh 2>&1 | tee -a /var/log/pcp_boot.log
#pCPstop------
```

Disable onboard analog audio on the Pi:
mount /mnt/mmcblk0p1
cd /mnt/mmcblk0p1
nano config.txt
Near the end of this file, comment-out[#] the 2 lines shown in bold below
onboard audio overlay
#dtparam=audio=on
#audio_pwm_mode=2
Again save and exit nano

Save files on machine and reboot:
sudo filetool.sh -b
sudo reboot (Wait some and ssh into the machine again)

Install python 3.6:
tce-load -w -i python3.6 (this loads and install's python3.6 which we shall use later)

Check loopback and like is working (snd_aloop kernel module):
aplay -l List's the playback interfaces (the dac & loopback's). Mine look's like this:

```
**** List of PLAYBACK Hardware Devices ****
card 0: Amanero [Combo384 Amanero], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: Loopback [Loopback], device 0: Loopback PCM [Loopback PCM]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 1: Loopback [Loopback], device 1: Loopback PCM [Loopback PCM]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
```

Now this is the tricky part, the /etc/asound.conf need to be corrected to fit the actual hardware and the loopback interfaces. (Further instructions in the Diyaudio camilladsp thread)

NOTE: This part with loopback interfaces and the next part with the asound.conf is the tricky and difficult parts for sure.

Adjust asound.conf to our hardware:
Before we do anymore, lets backup the /etc/asound.conf
sudo cp /etc/asound.conf /etc/BACKUP.asound.conf
sudo nano /etc/asound.conf (When done, save and exit nano)
Mine looks like this:

```
# Loopback device 0
pcm.squeeze {
type hw
card "Loopback"
device 0
format S32_LE
channels 2
}

# Loopback device 1
pcm.camilla_in {
type hw
card "Loopback"
device 1
format S32_LE
channels 2
}

# "sound_out" is the "real" hardware card (usbdac, soundcard etc...)
pcm.sound_out {
type hw
card 0
device 0
}

ctl.sound_out {
type hw
card 0
}
```

Make some dir's and install Henriks camilladsp dsp-engine:
mkdir -p /home/tc/DSP_Engine/filters/44100
mkdir /home/tc/DSP_Engine/filters/48000
mkdir /home/tc/DSP_Engine/filters/88200
mkdir /home/tc/DSP_Engine/filters/96000
Later if you need higher samplerates, create the dir's for them too / later :-)
The actual FIR / BiQuad's etc… filters have to be placed inside those dir's

Let's download the CamillaDSP Engine directly to the pCP/RPI:
cd /home/tc/DSP_Engine
wget https://github.com/HEnquist/camilladsp/releases/download/v0.3.1-alpha/camilladsp-linux-armv7.tar.gz

Unpack engine and set the executeable bit:
tar -xf camilladsp-linux-armv7.tar.gz
chmod +x camilladsp (maybe not needed, but dosen't hurt)
Let's save the things now, just in case :) sudo filetool.sh -b

Get files needed to make your'e Pi an SuperPlayer:
Now on you're laptop or whatever ;-), get my files from GitHub like this:
git clone https://github.com/Lykkedk/SuperPlayer.git

And transfer them with midnight-commander, scp or like to the pCP dir /home/tc:
cd /home/tc
ls -l
/home/tc should have these files now:
CamillaDSP.sh
exec_44100.py
exec_48000.py
exec_88200.py
exec_96000.py
filter.sh
null_44100.yml
null_48000.yml
null_88200.yml
null_96000.yml
py_six.tcz
py_websocket.tcz
README.md
squeezelite-custom

Set the executeable bits and copy files into they're right locations:
chmod +x CamillaDSP.sh
chmod +x filter.sh
chmod +x squeezelite-custom (This is the hacked version which detects and switch samplerate's)
cp py_six.tcz /mnt/mmcblk0p2/tce/optional
cp py_websocket.tcz /mnt/mmcblk0p2/tce/optional

Load and install the websocket I have packet in an pCP/Tinycore Linux format (.tcz):
Now we load the .tcz extensions on the pCP.
tce-load -i py_websocket.tcz
tce-load -i py_six.tcz
nano /mnt/mmcblk0p2/tce/onboot.lst
Paste the two lines in bold below into that file: (To load them automatically at boot)
pcp.tcz
pcp-6.1.0-www.tcz
nano.tcz
python3.6.tcz
py_websocket.tcz
py_six.tcz

Save and exit nano [ctrl] + o, exit [ctrl} + x

Copy some files to the right locations:
Hacked squeezelite-custom:
cp squeezelite-custom /mnt/mmcblk0p2/tce/squeezelite-custom
And the files needed two change samplerates on the fly:
cp *.py /home/tc/DSP_Engine/filters
cp *.yml /home/tc/DSP_Engine/filters

Save backup of machine:
Just in case!
sudo filetool.sh -b

Test CamillaDSP and squeezelite:
Let's see if camilladsp are ready:
sudo /home/tc/DSP_Engine/camilladsp -V
Output from terminal: CamillaDSP 0.3.1 (Looking good!)
Then execute to see if camilladsp is working:
sudo /home/tc/DSP_Engine/camilladsp -v /home/tc/DSP_Engine/filters/null_44100.yml
Looking good ??? : … Should look like this when okay:
[2020-07-15T09:14:14Z DEBUG camilladsp] Read config file
Some("/home/tc/DSP_Engine/filters/null_44100.yml")
[2020-07-15T09:14:14Z DEBUG camilladsp] Config is valid
[2020-07-15T09:14:14Z DEBUG camilladsp] Wait for config
[2020-07-15T09:14:14Z DEBUG camilladsp] Config ready
[2020-07-15T09:14:14Z DEBUG camillalib::filters] Build new pipeline
Buffer frames 8192
[2020-07-15T09:14:14Z DEBUG camillalib::filters] Build from config
[2020-07-15T09:14:14Z DEBUG camillalib::filters] Build from config
[2020-07-15T09:14:14Z DEBUG camillalib::processing] build filters, waiting to start processing loop
[2020-07-15T09:14:14Z DEBUG camillalib::alsadevice] Opened audio device "camilla_in" with parameters: HwParams { channels: Ok(2), rate: "Ok(44100) Hz", format: Ok(S32LE), access: Ok(RWInterleaved), period_size: "Ok(2048) frames", buffer_size: "Ok(16384) frames" }, SwParams(avail_min: Ok(2048) frames, start_threshold: Ok(0) frames, stop_threshold: Ok(16384) frames)

[2020-07-15T09:14:14Z DEBUG camilladsp] Capture thread ready to start
[2020-07-15T09:14:15Z DEBUG camillalib::alsadevice] Opened audio device "sound_out" with parameters: HwParams { channels: Ok(2), rate: "Ok(44100) Hz", format: Ok(S32LE), access: Ok(RWInterleaved), period_size: "Ok(1024) frames", buffer_size: "Ok(8192) frames" }, SwParams(avail_min: Ok(1024) frames, start_threshold: Ok(3072) frames, stop_threshold: Ok(8192) frames)
[2020-07-15T09:14:15Z DEBUG camilladsp] Playback thread ready to start
[2020-07-15T09:14:15Z DEBUG camillalib::alsadevice] Starting captureloop
[2020-07-15T09:14:15Z DEBUG camillalib::alsadevice] Starting playback loop
[2020-07-15T09:14:15Z INFO camillalib::alsadevice] Capture device supports rate adjust

[ctrl] + c to kill it.

Execute this to kill default squeezelite and start our custom-hacked squeezelite:
sudo killall squeezelite (it's proberly not running)
Start the CamillaDSP.sh start script to see if things are working before we go further.
cd /home/tc
sudo ./CamillaDSP.sh start
ps aux | grep camilladsp – Will show something like this if it's running:
root   0:04 /home/tc/DSP_Engine/camilladsp -p3011 /home/tc/DSP_Engine/filters/null_44100.yml

Now if we start our custom-squeezelite like this:
sudo /mnt/mmcblk0p2/tce/squeezelite-custom -n DSP_DAC -o squeeze -a 160:4::1 -b 10000:20000 -r 44100-192000:2500 -U -U -z

Please "X" your fingers now ;-) … It should work now, and changing samplerates on the fly is done automatically. -Try to play some music with samplerates between 44100 and 96000 Hz and look for samplerate changes on your'e dac (if dac is showing that is)

Let me explain the "custom" squeezelite parameters here quickly:
-n DSP_DAC   = name of LMS player
-o squeeze   = The loopback which catches the LMS stream (See /etc/asound.conf)
160:4::1   = Settings which works good for my dac (see pCP/squeezelite howto for more)
-b 10000:20000 = Good when used as streamer from Tidal/Qobuz etc.. (pCP/squeezelite howto)
-r 44100-192000:2500 = This one tells squeezelite that samplerates between 44100 and 192000 should make a 2,5sec. break when samplerate is switching (the :2500 needs some tweaking to got right, to make switching without noise, clicks and pops)

Edit some:
Now edit the /opt/bootlocal.sh file, and remove the # in front of the line which starts the CamillaDSP.sh script. (The line in bold below)
nano /opt/bootlocal.sh
```
#!/bin/sh
# put other system startup commands here

GREEN="$(echo -e '\033[1;32m')"

modprobe snd_aloop
/home/tc/CamillaDSP.sh start
sleep 1

echo
echo "${GREEN}Running bootlocal.sh..."
#pCPstart------
/usr/local/etc/init.d/pcp_startup.sh 2>&1 | tee -a /var/log/pcp_boot.log
#pCPstop------
```

Execute sudo filetool.sh -b and sudo reboot

Check if camilla is running after reboot:
When the RPI is done rebooting, login through SSH again and do
ps aux | grep camilladsp – Should show something like if it's running:
root 0:04 /home/tc/DSP_Engine/camilladsp -p3011 /home/tc/DSP_Engine/filters/null_44100.yml
This means that now camilladsp is started automatically – Great!

Now we can execute:
sudo /mnt/mmcblk0p2/tce/squeezelite-custom -n DSP_DAC -o squeeze -a 160:4::1 -b 10000:20000 -r 44100-192000:2500 -U -U -z
And try to play some music again to verify everything is good.

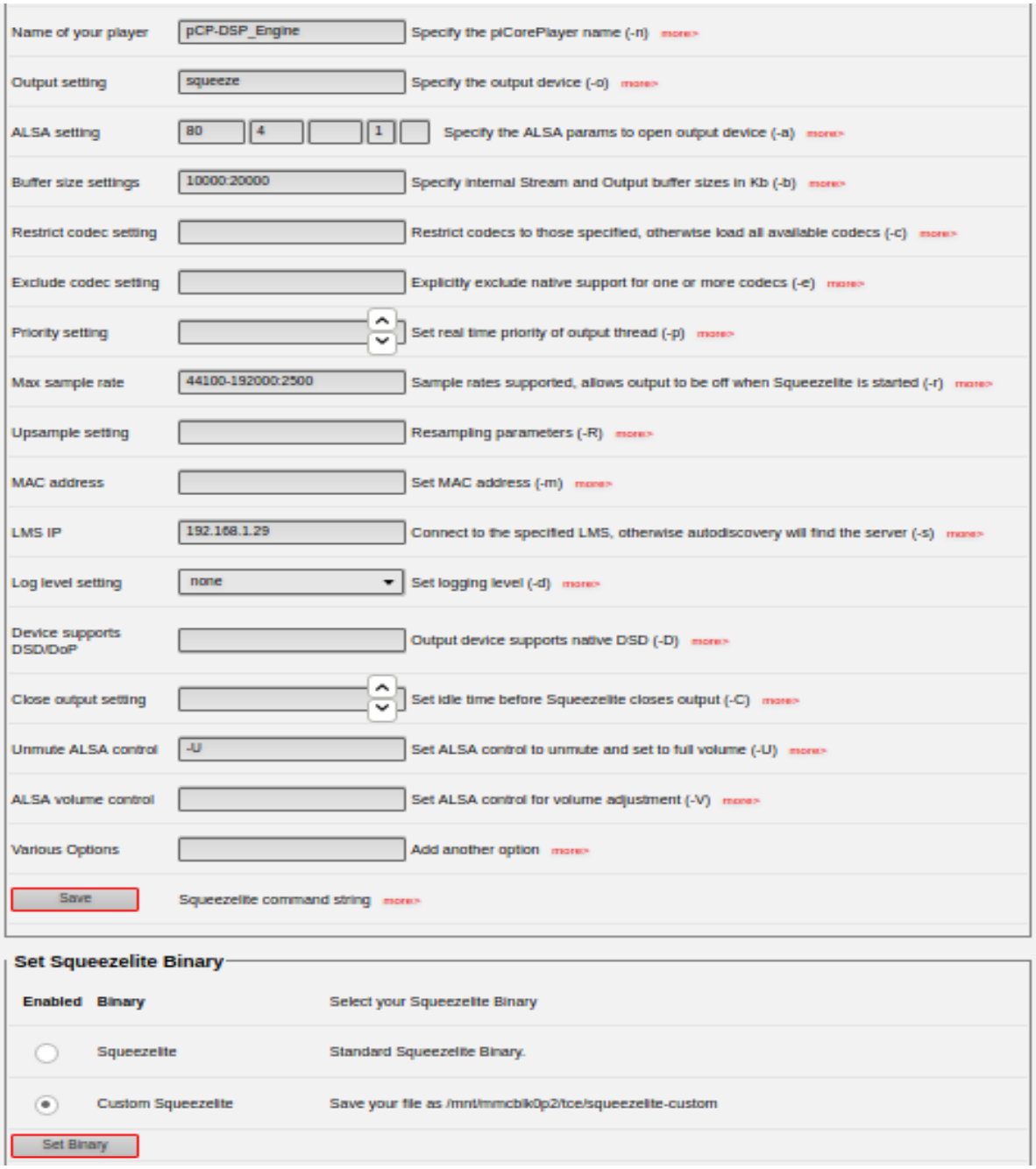
Explain some technical stuff :-) :
Before we setup the rest, let me explain the working order of the hacked squeezelite etc…
If we look at the exec files we have in the filters dir.
exec_44100.py shown below:
```
from subprocess import *
import time

from websocket import create_connection
ws = create_connection("ws://127.0.0.1:3011")

ws.send("setconfigname:/home/tc/DSP_Engine/filters/null_44100.yml")
ws.send("reload")
```
The hacked-up squeezelite sends command to execute this python script when samplerate changes to 44100. The same when eg. changed to 96000Hz, the exec_96000.py is executes from within squeezelite. I have provided possibility to change samplerates from 44100 upto 384000Hz in squeezelite, but then some extra exec_*.py files have to be created ofcause.

Setup the player in pCP webinterface:
Now try to setup squeezelite-custom in the pCP.
(Shown with my settings)
(pCP will proberly complain about big size file, but nevermind that now)

The output settings must be set to **squeeze** (our loopback which catches the LMS stream)



Remember to Set Binary for our custom squeezelite
It proberly takes some trial this setup part, but I had it working following my instructions so it's doable for sure ;-)

Tweaks and finishing up:
I have also provided a script called filter.sh
The script makes it possible to change filters when testing different configurations and FIR's, BiQuads etc… The filter is executes like this:
/home/tc/filter.sh nofilter
/home/tc/filter.sh filter
/home/tc/filter.sh filter_1
/home/tc/filter.sh filter_2
/home/tc/filter.sh filter_3
/home/tc/filter.sh filter_4
/home/tc/filter.sh filter_5
The script is selfexplaining, but you have to edit it to make it work as expected.

Before we finish this, lets cleanup all the un-needed files we have.
cd /home/tc (Important!)
rm *.py
rm *.yml
rm *.tcz
rm squeezelite-custom
sudo filetool.sh -b

Good luck.

Jesper (lykkedk@diyaudio.com)