Audio Virtual DSP Framework

Introduction

This is a framework for implementing an Audio virtual DSP capability, platform independent, with a concept of predefined macros or opcodes generated by an encoder, and a customizable runtime for executing these opcodes on any platform.
This results in the possibility to make a complex filtering or multiway cross-over model with a single description file, and to execute the resulting opcodes within a single program.
No user friendly graphical interface is provided due to the large possibilities for customization, but the dsp program is quite easy to describe with function helpers. Then GCC is used as the mean to interpret this program and generate the list of opcode for the runtime.

Creating a dsp program

Each supported opcode or dsp macro is defined in the file dsp_header.h
Due to the relatively large number of opcodes and the difficulty to interlink code, data and parameters consistently, an encoder is provided.
This is implemented as a list of C helper functions that can be called conveniently and then organized into a dsp program for a linear execution (no loop) at the sample rate frequency.

Each opcode is provided with one or many C helper function for generating the opcodes and their pointers to the parameters, or for generating the parameters with a proper structure and having them grouped in multiple "param" area.
All these function calls can be grouped in sub-programs for creating complex and flexible crossover models that will be sequenced and called by the main encoder function.
Some example are provided in the folder *dspprogs*, like a typical 2 way cross-over with bass management. This program will be linked with the main *dspcreate.c* provided in folder *encoder*.

Once the dsp program is described with the C functions, the *dspcreate.c* shall be compiled with GCC, including some other complementary files. The command line can be seen in the file *dspcreate.mak* (folder *build*)

During the compilation, some traditional error may appear if the user dsp program is not clean from a pure C declaration perspectives.
Once the file has been compiled into an executable, just run this executable to finally generate a file containing the expected list of dsp opcode.

During this execution, some messages appear in the console, showing progress of the opcode generation, and resulting program size. In case of inconsistency in the parameters definition, some error message will appear. Then you have to modify the dsp program and re-launch the GCC command line (or makefile) before re-executing the *dspcreate* executable file. This is a 3 step approach : create, compile, generate.

It is possible to combine multiple dsp program within *dspcreate.c* and to manage the behavior of the code generation phase by providing command line parameters, like cross-over frequency or the number of channels, or IO allocations. In fact the power of the C language can be used to make the code modular and the generator flexible.
The executable could be used by a end-user to generate his/her own final dsp program according to his/her custom parameters, while keeping a solid core code for the target platform.

Executing a dsp program

The opcode file generated in the previous step now needs to be interpreted by a runtime and the runtime needs to be fed by an audio source and its outcome shall be routed to a proper sound device.

The file *dsprun.c* is an example of an implementation for linux, using *alsa* loopback solution.
This program requires parameters including the opcode binary file, the sampling frequency, and the names of the pipe sourcing sinking the audio flow in the context of *alsa*.

The file *dsp_runtime.c* is the opcode interpreter, it is provided with couple of other files to support the key features (fir, biquads, maths…) and can treat either float or integer samples and can calculate the output in either float, double or 64bits integer.
Depending on the target platform some optimization can be done to maximize cpu performance.

Runtime parameters changing

For this framework, the model is that any parameter for the DSP shall be predefined and known either at the compilation stage with gcc, either and latest at the generation state via the optional command line parameters. This might not be flexible enough. For example, one may want to modify a gain, or a cutoff frequency, or a delay during runtime or when listen live program.

To make this feasible at some stage, all the opcodes parameters can be declared in “param” sections anywhere in the program, and a specific dump file can be generated with a list of symbols and their address, relative to the start of a given “param” section.

Procedure to read and write within a “param” section will be provided so that one can give the possibility to modify the parameters during runtime, by the mean of a command line program, or a graphical user interface. The intend of this framework is not to provide solution for this but to be ready supporting any of them.

<u>What’s next to come</u>
We hope this beta version will highlight the benefit of this concept and will bring contributors for enriching the dsp features, encoder, runtime and Operating system interfaces. The author is working on making this solution available for a DAC manufacturer and with help of community would like to make it efficient for the raspberry pi platform (or other soc / nuc) as a nice alternative to ladspa plugins.

We hope to find help and contributors to encapsulate this runtime within Audio Core and/or VST plugins!