REAL-TIME SIGNAL PROCESSING SYSTEM PROPOSAL AND IMPLEMENTATION

Carlos Gómez Moreno

Audio Department
Sinixtek ADTS, s.l.
carlos@sinixtek.com

ABSTRACT

This paper intends to describe the desirable features of a complete, powerful and highly customizable real-time audio algorithm implementation system, and to provide the guidelines to its implementation. The goal is to design a platform by means of which new sophisticated audio algorithms can be developed, tested and used with the minimum effort. The idea is to build large complex processing systems based on elemental building blocks which may interact in any possible manner. This way, by connecting existing, proved modules, such as filters, noise gates, or any new specific module, complex processes can be achieved and tested in a real-time environment with the minimum possible effort. The building-block philosophy would also make such a system very suitable for educational purposes, as it would make possible to 'hear' in real-time a particular complex algorithm with and without one of its blocks (a filter, for example), thus showing its importance.

1. SPECIFICATIONS

The main desirable features for a flexible and powerful audio processing system as described above are the following:

1.1. Flexibility

Connections between the different building-blocks must be arbitrary and unlimited, no topology pattern or block count limit should be imposed by the system. The connections must at least include signal connections and control connections. The latter may be defined as special connections that allow one building-block to change the parameters of other in real-time. Optionally, one block may have multiple signal input, output and control output ports.
It must be possible to change the parameters of any building block in real-time. This is done with two purposes: he first is to allow the user to make changes to the audio process and hear the effects immediately; the second is to allow automatic parameter modification. This way one building-block can be designed to change other block's parameters sinusoidally, and be reused both in a 'Flanger' or 'Tremolo' application.

1.2. Dynamic loading – Development community

The user must be capable of loading the unit with newly developed or third-party algorithms, and the developer must be able to hot-replace and debug new code. All this must be achieved without re-compiling the system core. The main objective is to promote an open code interchange community, by means of which new, complex algorithms can be developed very easily by building up on basic, tested blocks.

1.3. Portability

The unit must be able to function autonomously. This is due to the need to test algorithms in particular environments and conditions, with real-world signals. However, extra functionality may be provided when connected to an external device, such as a computer.

1.4. Programming simplicity

When the development of a new, specific building-block is needed, it must be programmed. The developer does not need to know the intrinsics of low-level programming. The system must provide him with a plain, standard interface where all architecture-dependant operations (such as I/O) are handled by the system. Ideally, the only knowledge required by the programmer is Digital Signal Processing.

1.5. Soft transitions upon process change

When the user requests a process change, the transitions between the old and the new process should be smooth and instant. This is particularly important when the process to be replaced has memory, because the discontinuity is more evident and unpleasant in such cases.

1.6. Efficiency

The intelligence and the code size in the unit must kept at a minimum in the real-time (audio processing) thread, leaving the as much calculations as possible to other low-priority threads. The system must be speed-optimized for patch execution.

2. IMPLEMENTATION

This section discusses the implementation details of the different elements that must be present in the designed real-time audio algorithm system.

2.1. Audio algorithms

An audio algorithm is basically a 'black-box' which is capable of producing outputs, either as a result of an operation on its inputs, or in an input-independent way (this would be the case of generators). These operations may depend on parameters the algorithm