Audio Library Processor Usage & Interrupts

Every audio object consumes processor time as it manipulates data. You can use AudioProcessorUsageMax and other functions to monitor how much of the processor's power is being used.

The AudioNoInterrupts function allows you to briefly suspend the audio library, which allows you to change multiple object's settings and have them all take effect at the same time when the library resumes.

Usage

AudioNoInterrupts();

Disable the audio library update interrupt. This allows more than 1 object's settings to be changed simultaneously.

AudioInterrupts();

Enable the audio library update interrupt. Any settings changed will all take effect at the same time.

AudioProcessorUsage();

Returns an estimate of the total CPU time used during the most recent audio library update. The number is an integer from 0 to 100, representing an estimate of the percentage of the total CPU time consumed.

AudioProcessorUsageMax();

Return an estimate of the maximum percentage of CPU time any audio update has ever used. This function is the most useful for assuring the audio processing is operating within acceptable limits. The number is an integer

from 0 to 100, representing an estimate of the percentage of the total CPU time consumed.

AudioProcessorUsageMaxReset();

Reset the maximum reported by AudioProcessorUsageMax. If you wish to find the worst case usage over a period of time, this may be used at the beginning and then the maximum can be read.

anyObject.processorUsage();

Returns an estimate of the total CPU time this particular object used during the most recent audio library update.

anyObject.processorUsageMax();

Returns an estimate of the maximum CPU time this particular object has ever used during the most any audio library update.

anyObject.processorUsageMaxReset();

Reset the maximum returned by processorUsageMax, for this particular object. Reporting for all other objects is unaffected.

AudioConnection objects do not have any functions. They are simply created in your sketch, after the audio objects, to define the data flow between those objects.

Example Program

TODO: an example showing varying memory usage...

Understanding Audio Library Scheduling

TODO: conceptual model...

TODO: multi-level interrupt priority - update low priority, data interrupts

high priority, most other arduino libs medium priority

interrupt latency and compatibility with other libraries

update responsibility of input and output objects.