

## Introduction

The document details the latest enhancements.

## Harmonic Pruning

This version improves the algorithm that creates the signal in the Timbre Wavetable – the sound signal you hear when a note is played – with a method I am calling ‘harmonic pruning’.

Refreshing the Timbre Wavetable is computationally expensive, because sinewaves are scaled and summed for every active harmonic. This new code uses lessons from psychoacoustic research about human hearing to ignore the calculations for harmonics that have insignificant loudness. i.e., ‘prune’ these harmonics from the overall sound, because it won’t make any difference when heard.

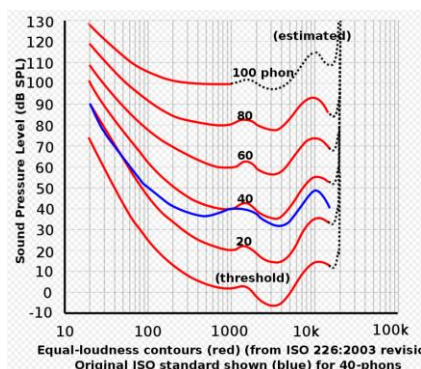
If you want to load version 8 onto the Tone Processors, then either load the hex file or compile and load the source (but you would need the MPLABX Pro licence to get the compiler optimisation required). It is not essential to load Version 8 firmware, the Windows app will still continue to work with the older version.

NOTE: The harmonic pruning mentioned here is in addition to any automatic pruning in the Windows App (which is simply looking at relative harmonic levels, without taking frequency into account). This is an optional feature of the Windows app, under Tools>Settings.

The basic premise for the new algorithm is that we look at the peak harmonic level of a timbre, then inspect every other harmonic and compare to the peak. If the relative level of a particular harmonic is less than a threshold of say ‘a sixteenth’ of the peak level (which is actually -24dB), then it seems reasonable to say that the sound from that harmonic is unlikely to be noticeable, and the harmonic ignored. This is an assumption, and the threshold is open to debate – but the goal is to aggressively ignore the overhead of unnecessary processing.

However, this can be refined further, because of the way humans perceive ‘loudness’ differently across the frequency range of hearing. Research performed back in the 1930s found that the human ear is most sensitive between 2 kHz and 5 kHz (most speech between 300Hz to 3kHz), largely due to the resonance of the ear canal and the transfer function of the ossicles of the middle ear. For frequencies outside of this range the sound pressure level must be higher for us to notice the sound. See the graph below.

[https://en.wikipedia.org/wiki/Equal-loudness\\_contour](https://en.wikipedia.org/wiki/Equal-loudness_contour)



So the frequency of the harmonic matters in determining if we can prune the harmonic. Instead of just using the raw harmonic levels to do the comparison with the peak, we work out a 'perceived loudness value' for each harmonic level, determine the peak loudness value and then prune based on this.

A loudness value in this context takes account of the equal-loudness contour shown above. For example, suppose we had harmonic A at 1.5KHz, raw level 100, and harmonic B at 100Hz, raw level 100. The loudness value calculated for B is calculated to be a lot less than that for A, because we struggle to hear B, compared to A.

The calculation involves simple lookups to make this adjustment. The associated spreadsheet shows the method used to construct the lookups.

The extent of automatic pruning (and CPU cost saving) based on this method is difficult to evaluate, because it depends on relative harmonic levels of the specific timbres used, plus the actual note played.

However, by using test code to toggle a pin to flag pruning, observation shows that typically a significant amount of pruning takes place (roughly 10-20% of the harmonics used in the tests, which were from example patches). This is a big win computationally, which will make the module more responsive and sonically agile on average - which is a good thing!

## Bigger SPI buffer

The buffer has been given an eight-fold increase in size, from 64 words to 512 words. There are situations, especially with dense MIDI processing, where a lot of data is being sent to Tone processors in a very short space of time.