

**Additive Synthesizer with GUI**

1. **Overview**

For the final project, the board was used for creating an additive synthesizer. The synthesizer parameters were mapped to GUI (graphic user interface) created in Visual Basic. In order to facilitate user input music keys, the keyboard was used via serial port.

1. **Hardware components**

* OMAP-L137
* Keyboard
* Data Cable
* PC

**Keyboard -** The keyboard input is implemented via serial port from the computer in which Code Composer Studio is running to the board. A Visual Basic application is used in order to read keystrokes from the keyboard and send them via the serial port to the board. In order to read these keystrokes UART (Universal Asynchronous Receiver/Transmitter) is used. Since the UART interface runs as a separate task, the keyboard input is synchronous and, therefore, the board is capable of taking keystrokes while it is doing audio processing.

1. **Software components**

****

1. Oscillators

A sound synthesizer (often abbreviated as "synthesizer" or "synth") is an electronic instrument capable of producing a wide range of sounds. In my implementation, I created a piano that can reproduce 9 different notes. The user, using the graphic interface, can choose and use up to 3 different waveforms.

* Sine
* Square
* Triangular

A volume control for every waveform was added to.

1. Octave control

In music, an octave is the interval between one musical pitch and another with half or double its frequency. In my implementation, I create an octave control, which allows the user to change between different octaves starting from middle C (C4).

1. Audio in

For my audio in channel I used a pass-through function which allows the user to play input audio while using the synthesizer.

1. Delay effect

Delay effects store the input signal—and hold it for a short time—before sending it to the effect input or output to create the sound of a repeating, decaying echo. In my implementation I have added delay effect that can be applied on the reproduced audio from the synthesizer or the incoming audio from the audio in channel. I also added a dry/wet slider, which allows the user to control the percentage of the applied delay, and feedback slider, which allows the user to control the delay feedback length.

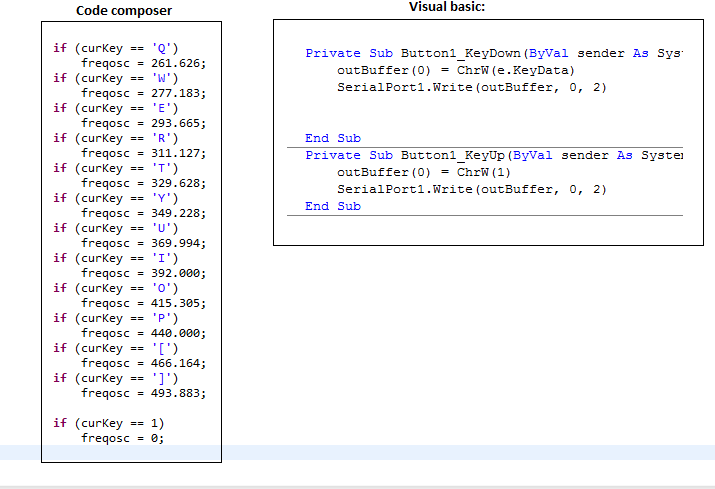
1. Amplifier (Master Volume)

For every output component, there is a gain which can be used to emphasize or depreciate a certain sample using sliders next to them. But a master volume slider was added also for preventing any kind of clipping in the output audio. In our case the amplifier was modeled with the following linear equation:

Fp[out] = g \* Op[out]

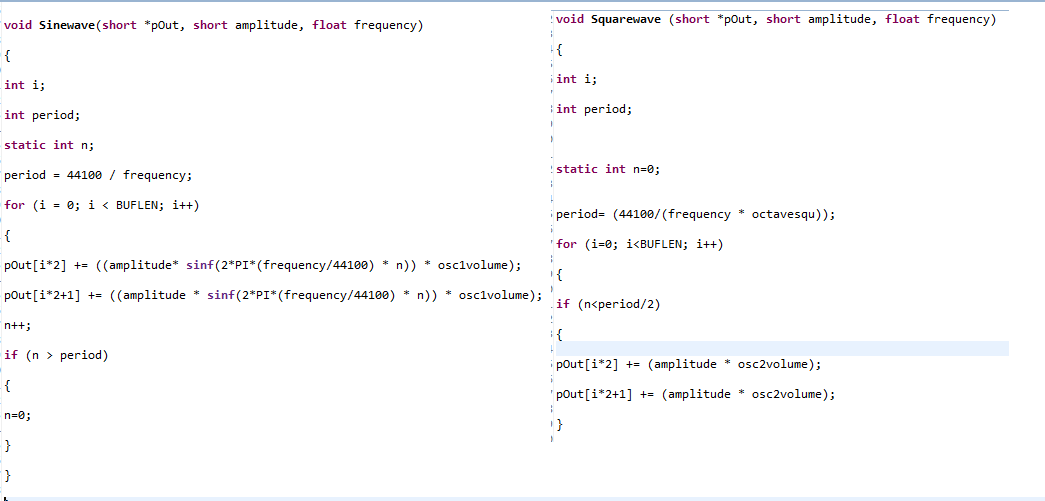
1. **Code explanation**

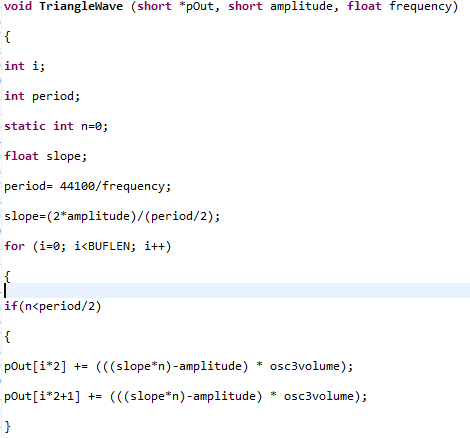
For this project I started with the keyboard template for lab 9. The first thing I modified in it is the different keystroke inputs from the keyboard responding on the music keys. Meanwhile the visual basic code from lab 9 was modified to respond on these changes.



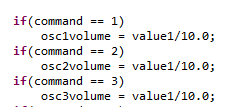
In visual basic we can see that every time when a button is pressed it sends the value of this button to outBuffer(0) but when its released sends 1 which in our C code sets the frequencies of the oscillators to 0.

After the keyboard code was added, the functions for the 3 different waveforms were added. (See on the next page)



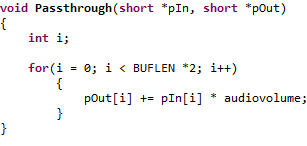


Inside every waveforms equation an oscillator volume control float value can be noticed as a multiplier. These values are the gain controls for the different waveforms.



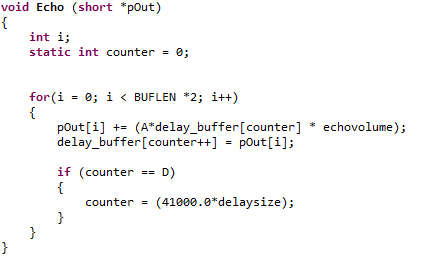
All this values are received from visual basic using slider controls.

For the n channel (pass-through) I used a modified version of the code from the very first lab which looks like this:



As it can be seen I have added a volume control multiplier.

The delay effect I used for my synthesizer is similar to the one we develop in one of the labs:



As we can see I added a volume control and delay size control which can be controlled from the GUI with a slider.

At the end comes the amplifier. Till this point we have added all the functions together by using “+=” which of course will create a problem with a clipping when everything is activated. To resolve this problem an amplifier is added which is basically a function which multiplies the whole sum of everything in output. The code of the function looks like this

