Transforming the PIC32 microprocessor into a musicbox

Christoffer Alenskog
Dante Johansson

December 6, 2015

1 Objective and requirements

The aim of this project is to develop the PIC32 microprocessor into a musicbox. We will connect an external speaker to the board. The user should be able to create sounds by pressing the buttons and switches and then arrange these sounds in any order. The resulting arrangement should then be playable as a song. It should also be possible to upload a file into the memory of the microprocessor which can then be decoded into a song. The requirements for the musicbox are as follows:

- The user should be able to press a button and hear a sound.
- By pressing buttons the sounds can be arranged in a specific order. With specified length and frequency.
- The unit should be able to decode a file containing sound information and convert it to sound. It will be possible to import files and store the sounds in the data memory to use them as preset songs.

Optional features if time allows:

- Using the display to navigate and show information about sounds etc.
- Changing the characteristics of the sounds to make them sound like different instruments like for example drums and synth.
- Adding a microphone to record sound.
- Create random song from designed algorithm.

2 Solution

We took a small speaker and plugged it into the ChipKit board. We wrote some C code to create and play sounds through the speaker on command from the buttons. The sound is generated by alternating the signal to the speaker between one and zero in specific frequencies. We then used a breadboard to connect some LEDs, resistors and buttons. The resistors limits the current so that the pins are not damaged by the speaker and the LEDs don't explode. The

LED's light up when a button on the board is pressed.

By frequently updating the value sent from the buttons it's possible to make the device react when a button is pressed. When pressing a button it can either trigger a tone or some other part of the code. When the switches are changed from their initial position the binary number representation of their signal gets translated into a tone. If the user then press any button used for generating sound this tone will be played and saved for that button. When the switches are all turned off this tone will be played each time the same button is pressed.

It's possible to make preset melodies by including an array containing the necessary information for creating tones with specific length. Each tone from C3 to C6 are defined as integer representations of how many times the timer have to overflow to create each wave for the tone. Tone length sixtenth, eighth, quarter, half and whole notes are also defined as the amount of times the timer have to oveflow to reach that duration.

3 Verification

The verification process have been done with continuous testing to check that all buttons and switches do what they are supposed to do. As well as comparing tones to with other sources. For example when a preset melody is played it's verified that it plays the predicted melody. Each additional functionality has been verified to work as expected by changing parameters and predicting results beforehand.

4 Contributions

Christoffer has written the code that handles playing of melodies and the main loop. Dante has written the code for playing a sound when a button is pressed, changing the tone if any switch is active and the example songs.

5 Reflections

Making sounds was easier than expected. And a lot of fun.