ECSE 324
Laboratory No. 5 Report

Ismail Faruk 260663521

Tanbin Chowdhury 260578441

Introduction

This is the report for Lab 5, implementing a musical synthesizer. The report is broken down into four parts, based on how I approached to create the software.

Part 1: Initialization

Description

This includes variables, arrays for storing notes, configuring timers, implementing timer interrupts, initializing display.

Approach Taken

For starters, variable initialization was a very important step, which included global and local scope, based on their implementation. There were variables defined globally for holding data like scan codes, frequencies, sampling frequency, etc. Local variables included multiple flags for identifying keypress, multiple integer variables used as array pointers, one location pointer variable for keyboard and some for the VGA.

Three timers were configured which would generate interrupts, one for each input/output process: audio, keyboard and VGA.

A signal generator function, which performed linear interpolation for extracting signal values from the wave table was implemented, as required. A wave generator function and eight arrays were used to store the data extracted, each array corresponding to the respective notes. This was done to reduce the task of the ALU once the machine enters the feedback loop.

Challenge Faced

Deciding which variables to be made local and global, some predetermined and some after testing. Also understanding the need to reduce the load on the ALU.

Part 2: Keyboard

Description

This includes identifying keypress and implementing the flags for identifying the notes, all implemented using the keyboard timer interrupt.

Approach Taken

A two-step process was taken used when implementing this part. Every time a scan code was read, the program would refer to the previously read scan code. This would determine what tasks to perform. If the previous scan code was null or a make code, then the new scan code would be a make code and the flag corresponding to the note of that new make code would be triggered. If the previous scan code was a break code, then the corresponding flag would be turned off and the array pointer would be reset. This flag system was implemented to fix the problem of multiple keypress where only one the latest key's make code would be read, while the others' make code would not. Thus, flags were implemented to cope up for that problem.

Volume key identification was also implemented within this part, where volume boundaries were set to prevent distortion.

Challenge Faced

Understanding the requirement of the two-step process and implementing it.

Part 3: Audio

Description

This part was about finding the total sample to write to the audio codec and increment the array pointers, all implemented using the audio timer interrupt.

Approach Taken

Based on the flags triggered by the keypress, the array values of those notes were extracted added to form the total sample. An integer casted modulus was implemented on the array index pointers, based on their frequencies. This was done for a continuity correction for all the notes, so that only the values of the first cycle in the note arrays be extracted. This fixed minor distortion in the signal.

Array index pointers for note arrays were only incremented when the corresponding flags were triggered and the write to audio codec was successful.

Challenge Faced

Understanding the need for multiple array indices for being able to perform continuity correction, understanding continuity correction and incrementing using flags as triggers.

Part 4: VGA

Description

This part was about writing the total sample to the VGA port, implemented using the audio timer interrupt and the VGA timer interrupt.

Approach Taken

Two arrays were used to display the data. One would store the y-coordinates for up to 320 values, triggered by the audio timer interrupt. Once the array is full, within the VGA interrupt, a loop would be used to write for 320 x-coordinates, the corresponding Y-coordinates, and display "white" pixel accordingly. The second array would copy each Y-coordinate value and write null to the previous coordinate in the next iteration. This would thus allow to show the flow of values across the display in a sinusoidal manner.

Challenge Faced

Implementing the array to write previous pixels to null was important. Otherwise, showing a sinusoidal flow in the VGA was not possible. A very precise solution to a difficult scenario.

Conclusion

Thank you for the semester. See you again if I fail. Thoughts and prayers.