Sounds Feature

APPLICATION NOTE – EXTRAPOINT 2

Lorenzo Ruotolo – s313207

ARCHITETTURE DEI SISTEMI DI ELABORAZIONE | A.Y. 2022-23

1 Feature Description

This document briefly explains how the sound feature was implemented on the Extrapoint 2 project. It allows the programmer to play a specific sound effect on the on-board loudspeaker. Each sound effect is saved as a variable length notes array on the board ROM. The notes represent the frequency at which the loudspeaker will be driven through a square wave signal.

2 Feature Implementation

To play a sound effect, the related note array must be available beforehand. The array should be a *uint32_t* array containing the series of note to play, expressed as their relative TIM3 match register value that needs to be set to have their specific frequency, taking into account that TIM3 normally runs at 25MHz. Notes *C4* to *B4* are already provided inside the header file *utils.h*. In addition to standard frequencies, the value 0 (*NOTE_PAUSE* in *utils.h*) can be used to implement a pause in the note series and the value 1 (*NOTE_ STOP* in *utils.h*) **must** be used to terminate the array.

With the array ready, the *utils.h* function ***playSong*** can be called anywhere in the program to start playing the sound effect. Its parameters are ***notes***: a *uint32_t* pointer to the notes array, and ***delay_ms***: a *uint32_t* value representing the period of each note in milliseconds. The function uses the match register MR0 of timer TIM3 to manage the note period and change. It also sets *game.currentSong* variables *notesArray* to *notes* and *currentNote* to 0.

Each time TIM3 reaches MR0 value after *delay_ms* milliseconds, its relative IRQ handler ***TIMER3_IRQHandler*** is executed. The handler then checks whether the current note is a *NOTE_STOP*, if not it calls the *utils.h* function ***playNote*** to play the current note of the array. *playNote* takes as a parameter ***note***: a *uint32_t* value relative to the note frequency. The function sets the match register MR0 of timer TIM2 to *note* to correctly drive the loudspeaker. It also checks that *note* value is not below a threshold (200) to filter out very high frequencies, to implement the pause functionality and to also avoid lagging the system with a small period timer generating interrupts.

Finally, each time TIM2 reaches MR0 value, its relative IRQ handler ***TIMER2_IRQHandler*** is executed. The handler will switch the value driven on the loudspeaker trough *LPC_DAC->DACR* between the values 0 and the *game.currentSong* variable *volume* – shifted six positions left. This will generate a square wave on the loudspeaker with the current note frequency. Using square waves instead of sine waves alleviates the load on the processor by generating less interrupts in a note period and does not require the board to store the sine wave values. With that said, this choice will also translate in generating a more “metallic” sounding notes, but that should not represent a problem with this specific application.

3 Quick References

The code relative to the discussed implementation can be found inside these files:

- *utils.h*
- *utils.c*
- *IRQ_timer.c*