|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B7* | | |
| Homework number: | *HW02* | | |
| Due date: | *October 12th* | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| 1 *Giampà Simone* |  |  | *x* |
| 2 *Massa Giacomo* |  |  | *x* |
| 3 *Raduzzi Lucafrancesco* |  |  | *x* |
| 4 *Micelli Johanna* |  |  | *x* |
| 5 *Galimberti Claudio* |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | HW02: play a song when the microphone triggers | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| First of all, we set the speaker (PA9) in TIM1\_CH2 mode, then we enabled the PWM Generation on channel 2 with a PSC = 100-1 for the timer1, and internal clock source.  In the main file we have defined:   * a struct note containing the tone and the duration of each note for the song * all the duration parameters for each note in the pre-processor directives * a *tempo* variable set to 150ms corresponding to the duration of the shortest note in the song (the duration of all the other notes is a multiple of this variable)   We created an array of struct note song containing the sequence of the notes and their duration for our song. The playNote function sets the period of the timer with \_\_HAL\_TIM\_SET\_AUTORELOAD(&htim1, note.tones) and the pulse as half of the period with \_\_HAL\_TIM\_SET\_COMPARE(&htim1, TIM\_CHANNEL\_2, note.tones/2).  After that, it starts the PWM generation for playing the note, waits for the duration of the note HAL\_Delay(time\_constant \* note.duration), and then stops before the next note.  To play the whole song we defined a function playSong, iterating over all notes.  MICROPHONE TRIGGER  We enabled the microphone (PA8) in GPIO\_EXTI8 mode. After that, we configured the interrupts in the NVIC section and set the GPIO mode of the PA8 configuration as external interrupt mode.  In order to generate the interrupt that triggers the start of the song, we exploited the HAL\_GPIO\_EXTI\_Callback function.  A binary flag is used in order to prevent accidental triggering of the microphone when the song is being played. When the microphone interrupt is generated, the interrupt handler of the microphone is disabled with HAL\_NVIC\_DisableIRQ(EXTI9\_5\_IRQn).  When the song finishes, we have ensured that we can replay the song by triggering the microphone again. So the interrupt handler is reactivated after the song has finished playing with HAL\_NVIC\_EnableIRQ(EXTI9\_5\_IRQn) | | | |
| Professor comments: | | | |
| Project name | HW02: play a song with timer interrupt instead of delay | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| Starting from the first part of the project we had to replace the delay function using a timer. In order to do so, we enabled timer 2 on channel 2 (pin PC7) and. After that, we enabled the TIM2 global interrupt in the NVIC section.  When the timer2 expires, the function HAL\_TIM\_PeriodElapsedCallback is called. A binary flag is used to allow concurrent execution in the main loop of the main function, to switch to the next note in the song. When the timer interrupt callback is executed, it signals the main loop in the playSong function to switch to the next note. This is done as an alternative to the delay function used in the first part of the project.  The playSong function works similarly to the previous part of the project. Every time the playnote function is called, it starts the PWM generation with timer1 (used for modulation of the frequencies of the notes) and at the same time, it also starts the timer-based interrupt generation with timer2, using HAL\_TIM\_Base\_Start\_IT(&htim2).  We set the timer 2 auto reload register to change the duration of the notes with \_\_HAL\_TIM\_SET\_AUTORELOAD(&htim2, time\_constant \* setNote.duration). We reset the counter with \_\_HAL\_TIM\_SET\_COUNTER(&htim2, 0) and in the end we clear the interrupt flag with \_\_HAL\_TIM\_CLEAR\_FLAG(&htim2, TIM\_FLAG\_UPDATE) to avoid pending interrupts. | | | |
| Professor comments: | | | |