|  |  |
| --- | --- |
| **Mark** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A2* | | |
| Homework number: | *HOMEWORK 012* | | |
| Due date: |  | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Hui Jiang |  |  | *x* |
| Mattia Sironi |  |  | *x* |
| Gabriele Landi |  |  | *x* |
| Arturo Caliandro |  |  | *x* |
| Luigi Lizzini |  |  | *x* |
| Notes:  none | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name |  | | |
| Not done | Partially done   (major problems) | Partially done   (minor problems) | Completed |
|  |  |  |  |

|  |
| --- |
| We have done the whole homework, here is our explanation:  **Part 1:**  Project 2a: as this project is done during class and explained by the professor, so we don’t explain here again, actually our code for this project is the same as the one of professor.  Project 2b: although this project is also like the Project 2a which is also done during class and explained by the professor, but we have make some optimizations as shown below: |
| We have changed the function playnote: as you can see, the green part that we have make commented, it’s the version of the professor which is just copied from the function MAX\_TIM1\_Init and changed a few rows. Our version is to use a few functions to change the AUTORELOAD register and COMPARE register and reset the timer counter every time play a new tone as shown in the row: 166 168 169. Those functions we have found in the document:”HAL functions.pdf”, and here are the extracts of those functions: |
| And we have proved it using the board.  Project 2c(also recognized as Homework 02 part 1):  Starting from this project, we have encountered a lot of difficulties and make us think a lot.  Firstly, we have configured the board as shown below: |
| As you can see, we have configured the pin PA9 which is connected to the speaker as TIM1\_CH2 in order to later generate PWM signal and the pin PA8 which is connected to the microphone as GPIO\_EXTI8 in order to detect sounds and in consequence generate a interrupt.  Secondly, we configured the TIM1\_CH2 as shown below:    Here it really doesn’t matter the value of the AUTORELOAD register and the pulse value(the value of the register COMPARE), because we are going to overread them with different values according to the tone it’s going to play. We just fix the prescaler to 99(in the formula become 100), therefore our calculation will be easier.  Before we explain the third step, we want to explain a curious thing that we have encountered: In our first idea is based on the code of the previous project, we just need to put the playsong function into the HAL\_GPIO\_EXTI\_Callback function, then everytime there is a sound, triggers the callback function and as this function contains the function playsong, therefore, it will play the sound. This idea seems right, but its not true, according to our test, if do in this way, after the microphone detect the first sound from the environment, the speaker will play the first sound of the first tune forever. Thus, this makes us think the reason, and finally we have found the problem, if we don’t do anything, in the NVIC table, the EXTI interrupt and the Time Base: System tick timer which is triggered by the function HAL\_Delay) have the same priority. Which means: when the microphone detect the first sound, cause the EXTI interrupt, in consequence, execute the code within the function callback, however, in this function, we have playsong function, which contains loads of HAL\_Delay function which causes the the Time Base: System tick timer interrupt, as they have the same priority, we imagine that these two interrupt execute at the same time, and what does that mean? Means that when the speaker begin to play the song after the microphone detect the first sound, the microphone can detect the song from the speaker, then trigger EXTI interrupt again and begin to paly the song again, and then these happens again and again. Therefore, we use the method as shown below to fix this problem:    We changed the priority of the EXTI interrupt, thus one the speaker is playing the song, the microphone can’t trigger the EXTI interrupt. For doing this, we need to change the Priority group as 1 bits for… and then in the row EXTI line[9:5] interrupts, change the value to 1 which means a lower priority.  Finally, in the main.c, we maintain everything as the previous project but add the callback function:    As the professor commented on the previous lecture, we add steps for the clear flags. And we have also proved it using our board.  Homework02 2:  As these project is just do again the project2c without using the HAL\_Delay function, the configurations is almost the same as the previous project. As the hint, we are going to use the timer instead of the HAL\_Delay, so need to configure another timer, and we have used the timer1 channel 2 to generate the PWM signal, so we choose the timer 2 to replace the HAL\_Delay:    We have fixed the Prescaler to 8399 in order to make later our calculation easier. And the value of the counter period really doesn’t matter, because we are going to overread it according to the duration of the tune it’s going to play. And other configurations on the board is the same as the previous project(although also this time we don’t need to change the priority in NVIC, but better we talks about it later).  Secondly, we define some global variables in the main.c:    The variable song\_playing identify if the song is over or nor, 0 means is over and 1 means not. The variable index identifies the order of the tune that it’s playing.  Then every time there is a sound, triggers the callback function:    And callback function firstly check if the previous song is over or not, if it’s not, it will do nothing, and that’s the reason why we don’t need to change the priority on the NVIC board.and as we have initialized the song\_playing to 0, so if it’s the first time detect the sound, it will execute correctly. Then if the previous song is over, it will execute the function playing\_init:  Then, the function change the value of song\_playing to 1, which means now we are playing the song and initialize the index to 0 and finally call the playnote function:  The first thing we do in this function is to stop the PWM(just in case, maybe it’s not necesarry), and then chec if the song is finished or not, if it’s finished, the program will stop the timer 2 and clear the falg of the timer2 and put the song\_playing to 0 in order to indicate to the other part of the program the precious song is over. And return. If it’s not over, it will set the timer1 channel 2 to generate the pwm signal according to the tune is playing. After doing that, the program will set the timer2 in order to counter to the end of this tune and generate a interrupt and in its callback function, trigger to play the next song. As the prescaler is 8399(row 125), the counter plus 1 every 0.1ms. therefore, in the line 127, we set the period based on the duration of every tone. Thus, when the counter counts until this number, will trigger its callback function as shown below:  Then the program will play another tone, and another tone until the song is over. And if the microphone detects another sound, the program will do the same thing that we have mentionaed so far. And we have also proved it with our board.  Here are the descriptions of those HAL functions we have used in this project: |
|  |
| Professor comments: |