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

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Keyboard with timer interrupt and debounce | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| Explanation:  We successfully completed the homework.  Next, we explain all the steps for completing the homework:  Project 1b:  First, we have configured our board as the previous project:  Then, we configured the timer 2 as shown below:  As (8399+1)\*(99+1)/84e6 = 0.01s = 10 ms, in our case, we will generate a interrupt every 10 ms.  In order to have a better performance, we enabled the DMA for UART2\_TX:  Before we go to the code, we enabled the following interrupts:  In the code, we first defined these GPIO ports in order to facilitate the code:  Then we created a series of global variables we are going to use:  Most of the above variables were been used also in the previous project(the code showed us during the lecture), but for every button, we have defined a variable debounce\_time in order to deal with the debounce problem.  In the main, we initialized the three arrays we have defined. As you may notice, we initailized all the elements of debounce\_time to 1 which we will explain it when we get that point. As the column\_index initial value is 0, so in the main, we seleted the first column in order to make the callback fucntion of timer in its first iteration could read the values of first column. Then we started the timer 2 in interrupt mode:    Finally, in the callback function of the timer:  We first read the 4 rows of the selected column(as we mentioned,in its first iteration,it would read the 4 rows of the first column). Then we created a for loop: we first check if the corresponded button is being pressed, if it’s the case, we check if its previous value is not 1(wasn’t being pressed), if it’s also the case, we check if the corresponded button has experienced the defined debounce compensation time(in our case:40ms, we say in advance that with this time, we got a very good performance), if it is also the case, we can send the corresponding “symbol” using uart dma which will not block until it finished its work thus we get a better performance. If the first two previous conditions is completed but the last one, we put the corresponding debounce\_time in 0 therefore after 10ms\*4, it will completed, then send the symbol. And if the button is not being pressed, we just put this ack to 0. Finally, we selected the next column in order to read its value next time the callback function is being called(after 10ms,this also give time to the correspoding GPIO PIN to reach GPIO\_PIN\_SET).  Finally, the code works as we expected: | | | |
| Professor comments: | | | |