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

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | SPI with LED Matrix using DMA | | |
| 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:  Firstly, we have configured our board the same as the project 1a(the one that explained during the lecture):  Then we configured our SPI1 as shown below:  As in this project, we need to use the SPI\_DMA, so we also added the DMA:  In order to avoid using HAL\_Delay function, we configure two timers for different purposes:  We configured the timer 2 with a period 4ms with the same purpose as the previous project.  We configure the timer 3 with a period 1s in order to control the shift frequency of the symbols(the symbol that is printing in the LED matrix)  Before we switch to the main.c, we enabled a series of interrupts as shown below:  After doing that, we went to the code, we first define a series of global variables:  Where variables H,U,I represent the letter “H”,”U”,”I”respectively. The variable display\_content repreent the pixels’ value for the symbol/letter is printing on the LED matrix. The variable column\_index represents the column of the symbol is going to print on the LED matrix. Finally, the variable order indicates the index of symbol is printing on the LED matrix, in our case, when 0, means the board is printing the letter “H”, 1, means the board is printing the letter “U”, 2 , means the board is printing the letter “I”.  In the mean, we just need to start timer2 and timer3 in interrupt mode:  Then we created the callback function for the timers:  If is the timer 3, we just change the variable order in order to print in the following order:  H U I H U I H U ……….  Ann put the correspondence pixels’ value in the variable display\_content  If is the timer 2, we just transmit the corresponding column of the display\_content. As HAL\_SPI\_Transmit\_DMA is in the non-blocking mode, which means the processor will process the code in the next line while the transmit operation is still operating in the DMA, therefore we can’t change the value of the variable column\_index in the next line, otherwise something strange would happen. So we put the part of the operation of column\_index in another callback function:  As you can see, this callback function is called once the transmisson is finished, so in this case, we can change the value of the variable column\_index.  Finally, the code works as we expected. | | | |
| Professor comments: | | | |