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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A2* | | |
| Homework number: | *HOMEWORK 08* | | |
| Due date: | 26/11/23 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Hui Jiang |  |  | *x* |
| Mattia Sironi |  |  | *x* |
| Gabriele Landi |  |  | *x* |
| Arturo Caliandro |  |  | *x* |
| Luigi Lizzini |  |  | *x* |
| Notes: none | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Encoder with timer and UART 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 as the project 1a:  Then we configured the timer3 as the project 1b also:  In order to avoid using the HAL\_Delay function, we also enabled the timer2 as our timebase:  As this project requires use DMA to transfer the UART data, we also enabled it:  Our final step in the GUI is to enable the following interrupts in the NVIC table:  In the main.c, we first defined a series of global variables(which are the same as those of the Project1a):  As you may notice, we have also commented another kind of definition of the variables counts and oldcounts, that is because when we deal with the problem of downflow and overflow we will use deferent ways to solve it. And we will explain it when we get that point.  In the main, we just started timer2 in interrupt mode and started also timer3 in encoder mode:  Finally, we established the callback function for the timer 2:  The code is actually the same as the that within the infinite while. The code we show here is the version that counts and oldcounts are of type int16\_t. In this case, we don’t need to worry about the problem of overflow and underflow. Because in this case, for example, the compiler can identify 0xffff as -1 instead of in the case of uint16\_t, the compiler will identify 0xffff as 65535. In conclusion,what we want to say is that in the case of int16\_t, the compiler reads the integer using 2’s complement.  Finally, we would like to explain the the role of the polarity in the encoder settings and the working principle of this counter.  In order to explain this, we need to show the results we get:  when the polarity is set as shown below:  we found that when we clockwise rotated the encoder, the counter will increase but when we counterclockwise rotated the encoder, the counter will decrease. This result can be explained by the following diagram:  Image that B is channel 1 and A is channel 2, when clockwise rotated the encoder, we can find that the falling edge of channel 1(B) is always before the rising edge of the channel 2(A), so that makes counter increase.(in this case, we have to watch the diagram from left to right) But when we counterclockwise rotate the encoder, the falling edge of channel 1(B) is always after the rising edge of the channel 2(A), therefore, this will make the counter decrease. (in this case, we have to watch the diagram from right to left)  Now we change our polarity in the following way:  we found that when we clockwise rotated the encoder, the counter will increase but when we counterclockwise rotated the encoder, the counter will decrease. This result can also be explained by the following diagram:  Image that B is channel 1 and A is channel 2, when clockwise rotated the encoder, we can find that the rising edge of channel 1(B) is always before the falling edge of the channel 2(A), so that makes counter increase.(in this case, we have to watch the diagram from left to right) But when we counterclockwise rotate the encoder, the rising edge of channel 1(B) is always after the falling edge of the channel 2(A), therefore, this will make the counter decrease. (in this case, we have to watch the diagram from right to left)  Now we change our polarity in the following way:  we found that when we clockwise rotated the encoder, the counter will decrease but when we counterclockwise rotated the encoder, the counter will increase. This result can also be explained by the following diagram:  Image that B is channel 1 and A is channel 2, when clockwise rotated the encoder, we can find that the rising edge of channel 1(B) is always after the rising edge of the channel 2(A), so that makes counter decrease.(in this case, we have to watch the diagram from left to right) But when we counterclockwise rotate the encoder, the rising edge of channel 1(B) is always before the rising edge of the channel 2(A), therefore, this will make the counter increase. (in this case, we have to watch the diagram from right to left)  Now we change our polarity in the following way:  we found that when we clockwise rotated the encoder, the counter will decrease but when we counterclockwise rotated the encoder, the counter will increase. This result can also be explained by the following diagram:  Image that B is channel 1 and A is channel 2, when clockwise rotated the encoder, we can find that the falling edge of channel 1(B) is always after the falling edge of the channel 2(A), so that makes counter decrease.(in this case, we have to watch the diagram from left to right) But when we counterclockwise rotate the encoder, the falling edge of channel 1(B) is always before the falling edge of the channel 2(A), therefore, this will make the counter increase. (in this case, we have to watch the diagram from right to left)  Finally, we can conclude that when the event defined by the polarity of the channel 1 is happend before the one of the channel 2, the counter will increase otherwise the channel will decrease. And with this way, we can easily determine the direction of the rotation. And we can also see that the different configuration of the two polarity just make clockwise rotatation increase/decrease counter counterclockwise rotation decrease/increase counter. But with any of those configurations of polarity, we can determine the direction of rotation. | | | |
| Professor comments: | | | |