Tyler McKean README for Project 3 Stepper Motor and SysTick Timer

BASIC STRUCTURE

This program utilizes the 28BYJ-48 Stepper Motor, Darlington Array, and joystick of the STM32L476 Microcontroller as inputs to do the following tasks:

- Turns the stepper motor exactly 360 degrees clockwise by using half-stepping when the UP button of the joystick is pressed.
- Turns the stepper motor exactly 180 degrees counterclockwise by using half-stepping when the DOWN button of the joystick is pressed.

To achieve these tasks the Clock was enabled to GPIO Ports A and B.

- GPIO Port A is tied to the joystick of the discovery board with pins PA3 being the UP button, and PA5 being the down button.

For the majority of this project, rather than initializing each necessary register in the main function of the project, I declared and used a function that would initialize each necessary register and called it in the main function. To initialize the clocks to GPIO A & B, a GPIO_Clock_Enable() function was made that used the hex value 0x3 to initialize bits 0 and 1 in the Clock register. A similar process was done to initialize the joystick tied to GPIO A and GPIO B pins that were used to move the step motor.

In the MODER of GPIOA, the compliment of the hex value 0xCC0 was used to clear pins 3 and 5 set them as digital inputs. The up and down button are not grounded by a resistor by default like the center button is, so again, the compliment value of 0xCC0 was used in the PUPDR to clear pins 3 and 5. A hex value of 0x880 was used to set pins 3 and 5 as pull down, which grounded the necessary pins.

Another function was declared and made to set all the necessary pins for GPIOB. The necessary pins were set as general-purpose outputs in the MODER by clearing the necessary pins with the compliment of the hex value 0x0F0F0. This value cleared all the necessary bits for pins 2,3,6, and 7 and then individually each pin was initialized as 01 by left shifting a value of 1UL over to the necessary pin locations in the MODER. The four pins of GPIOB were then initialized as open-drain in the OTYPER by using the compliment of the hex value 0xCC0 to set the four pins. Lastly, the four pins were set as no pull up no pull down in the PUPDR by using the compliment of the hex value 0x0F0F0. Now that this function was made, I could simply include the function name, GPIOB_Pin_Init() in the main function and all the necessary pins will be initialized for the project.

Following this same process, I wrote and declared two separate functions called HalfStep360() and HalfStep180() that would turn the step motor 360 degrees clockwise and 180 degrees counterclockwise. Since both functions are required to perform a half stepping of the step motor, an array of 8 values was used for each function to create a step sequence of 8 pulses and had 4 step angles. This array was declared as the hex values: 0x44, 0x04, 0x84, 0x80, 0x88, 0x08, 0x48, and 0x40. A nested for loop was used in each function using the int variables I and j. The j loop would iterate from zero to a specified value in the parenthesis of the function. If that value was 512, the function would step through 360 degrees of the step motor revolution and if that value was 256, the function step through 180 degrees of the step motors revolution. The i loop in each function would step through each value in the half

stepping array and determined which direction the step motor would turn, either clockwise or counterclockwise. For the HalfStep360() function, the i loop iterated from i equals to 7 and decreased each step to 0, which walked through the values in the array from right to left. This results in the step motor turning in a clockwise rotation. Likewise, the HalfStep180() function iterated from i equals 0 and increased each step to 7, which read the array of values from left to right, which resulted in a counterclockwise rotation. Inside both function's i for loop, the ODR of GPIOB was cleared for the necessary bits by using the compliment of the hex value 0xCC. The ODR was then initialized with each HalfStep[i] value. So, when the i loop steps through the array, the pins of the ODR in GPIOB are being initialized to urn the step motor. The SysTick delay function was also included in the i for loop to add some delay between each iteration. For the HalfStep360() function, I set the delay to be 3ms, which would add a 3ms delay in between each i loop iteration, and for the HalfStep180() function, I made the delay a bit slower at 5ms. With all these functions defined, I was ready to code the main function for the project.

Above the main function, all the functions I defined as well as the predefined function were declared so I could use them in my main function. The System_Clock_Init() was initialized first in the main program to set the system clock to 80MHz. From here the GPIO_Clock_Enable(), GPIOA_Pin_Init(), GPIOB_Pin_Init(), and SysTick_Init() functions were all called sequentially at the start of the main program. Next, I made an infinite while() loop that would continuous run. Within this loop, I declare two other while loops that would check for when the up button and down button were pressed. To check when the up button was pressed, a while loop was made to check when the IDR of GPIOA and the hex value 0x8 were used in an AND expression. The hex value represented pin 3, which would be a 1 when the up button is pressed. This loop then continues to state that is the up button is pressed then call the HalfStep360() function and turn the step motor 360 degrees. This was made possible by the value 512 within the parentheses. Another while loop was made to check when the down button was pressed by checking when the IDR of GPIOA and the hex value 0x20 was executed with an AND expression. This hex value represented a 1 for when the down button was pressed, and if the statement was true then call the HalfStep180() function and turn the step motor 180 degrees counterclockwise. This was made possible by the value 256 in the parentheses, which is 512/2, so the step motor would only turn half the total steps than the previous function. This completes the basic structure of the program and the executes the given task for the assignment.

<u>HOW TO TEST</u>

- To test this project, simply plug the USB cable into the STM32L476 into the computer and run the project file in Keil uVision 5.
- After a successful build, click the debug option and press the reset button on the discovery board
- If the UP button is pressed on the joystick, the step motor will turn exactly 360 degrees in the clockwise rotation and then stop.
- If the DOWN button is pressed on the joystick, the step motor will turn exactly 180 degrees in the counterclockwise rotation and then stop.
- By performing these action will prove that the project works correctly.

TEST PROCEDURE - Videos of the testing procedure are included in the project's zipped folder.