ARM Project #4

This project aims to program blinking LED3 and LED4 on the STM32 Value Line Discover kit at a selectable frequency using Timer 3.

1. Start from a copy of the example project titled “SysTick” open in your Keil program. Just copy the whole project into a folder and call it TIM1 before opening and modifying it. Remove the part that toggles LED3 and LED4 in main() using the delay() function to disable the SysTick-based LED toggling.
2. Add ARM Timer standard drivers by “stm32f10x_tim.c” to the StdPeriph_Driver group in your Project window. This is done by right-clicking the group and selecting “Add Files to Group”. This file is located with all other standard peripheral drivers in the Libraries\STM32F10x_StdPeriph_Driver\src folder. Read the definition of the timer drivers in that file.
3. Include "stm32f10x_tim.h" in both your User group source files (main.c and stm32f10x_it.c) to allow calling its functions within both files.
4. Set the following parameters for your timer: counter value, autoreload value and prescaler value according to the rate required for LED blinking. In this project, the requires rates are:
    a. 1 Hz – 50% duty cycle.
    b. ½ Hz – 25% duty cycle.
5. Enable Timer 3 Update Interrupt from TIM3_DIER.
6. *Enable Timer 3 Peripheral clock in APB1 peripheral clock enable register (RCC_APB1ENR). (See Hints)*
7. *Enable Timer 3 TIM3 gloabal Interrupt in NVIC. (See Hints)*
8. Open “stm32f10x_it.c” from the User group and update it by adding your interrupt handler function for Timer 3 interrupts. Follow the instructions at the end of this file to know how to do that.
9. Write your Timer 3 overflow interrupt handler to include the following:
    a. Reset the Update interrupt flag bit in TIM3_SR register.
    b. Toggle LED3 and LED4 just like you did in previous projects.
10. Start Timer 3 by setting the Counter Enable bit in TIM3_CR1.
11. Compile and load your code into the kit.
12. Go to debug mode and run the code. Show TIM3 in a System View window and check out its values by stopping execution and inspecting the present values in all TIM3 registers. You can modify any of them and run the code again and the

program will use the modified values. Reset the CPU to return to the original program values.

Hints:

- A working function that performs steps 6 and 7 above by enabling Timer 3 clock and interrupt is as follows:

```
void TIM_Config(void)
{
  NVIC_InitTypeDef NVIC_InitStructure;

  /* TIM3 clock enable */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

  /* Enable the TIM3 gloabal Interrupt */
  NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

- Use initial values of the Counter and Autoreload registers as 0xFFFF and the Prescaler as 0x100 to get near the rates required by the project.

Note: *Modify only files in the "User" group of your project and never change the standard peripheral drivers since they will affect other programs not just the one you are working on at the time. Modification of such files will result in deducting points off your project grade.*