Lab 6 Timer Applications

Image from huckleberry.com

Section 1 Lab Objectives

When this lab exercise is completed, the student should be able to use timers:

1. Use timers to read the distance between objects.
2. Use timers to generate an interrupt and start a process.
3. From STM: The STM32 timer peripheral was conceived to be the keystone peripheral for a large number of applications: from motor-control applications to periodic-events generation applications.

Section 2 PreLaboratory Preparation

Prior to your scheduled laboratory meeting time the following items need to be completed.

On Line Learning

1. Watch *Pulse Width Modulation (PWM) - Electronics Basics 23* `https://www.youtube.com/watch?v=GQLED3gmONg`
2. Watch the first $6\frac{1}{2}$ minutes of *Getting Started with STM32 and Nucleo Part 6: Timers and Timer Interrupts — Digi-Key Electronics* `https://www.youtube.com/watch?v=VfbW6nfG4kw`
3. Watch *STM32F4-Discovery and 2 HC-SR04 ultrasonic sensors* `https://www.youtube.com/watch?v=5usorjLqtHk`

Preparation for the Prelab Quiz

The quiz will be available in lab for the first ten (10) minutes of your laboratory session. You may use any of your prelab preparation as a reference while taking the quiz.

Section 3 Ultrasonic Sensor

Follow the instructions in the pre-lab video *STM32F4-Discovery and 2 HC-SR04 ultrasonic sensors* with the following modifications.

1. When you start the project, select our Nucleo-L476RG (not the F4 Discovery) and accept the default configuration.
2. In the Clock Configuration shown in figure 1, we *only* need to adjust APB2. Change the APB2 Prescaler to /2 to make APB2 match the video value of 40MHz.
3. The default settings for our board includes the UART, so you can skip the setup of the UART shown in the video.
4. Change TIM8 Trigger Source to TI1FP1.
5. For the trigger signal use timer 16 (instead of TIM11). Setup channel 1 for PWM generation. Later in the setup, remember to use htim16 instead of htim11.
6. The video is using two ultrasonic sensors and we will use one. Channel2 Input Capture Mode is not needed and the variables echo2 and dis2 are not needed.
7. Ignore the code creating printf.
8. TIM8 configuration is shown in Figure 2 and TIM16 configuration is shown in Figure 3.
9. Listing 1 displays the code implemented in the main(void) function utilizing UART2. To use snprintf, make sure to include the stdio.h library in your code.

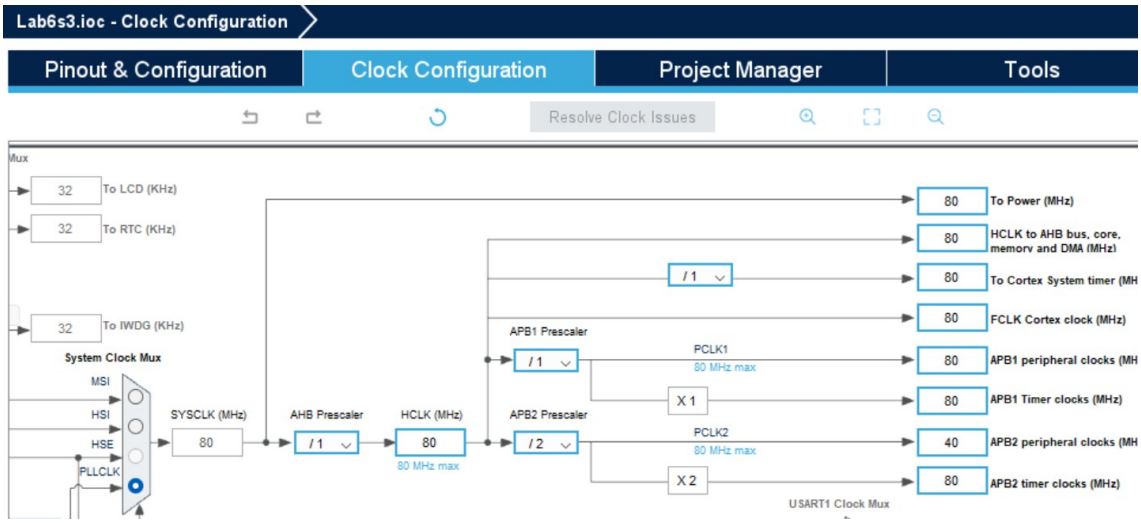

Figure 1: APB2 ioc file adjustment

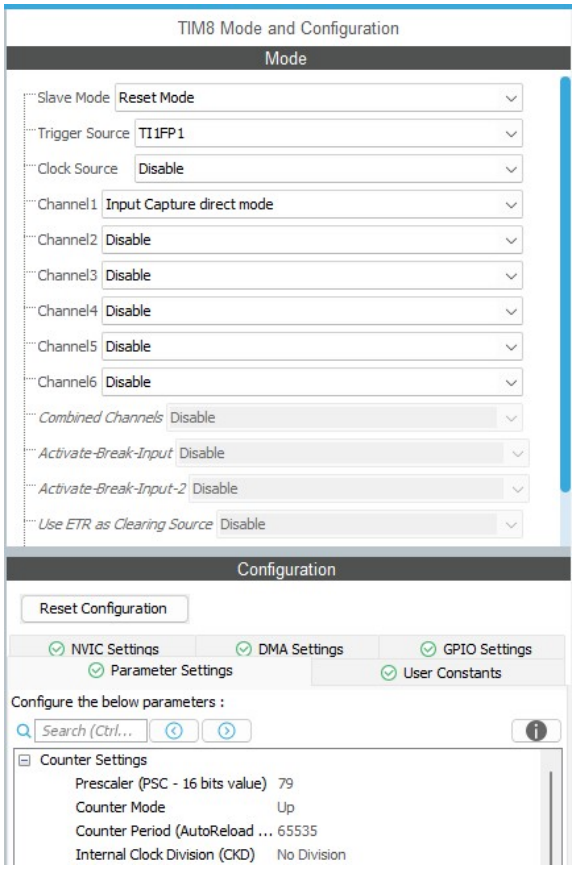

Figure 2: Echo Pin - Timer 8 Channel 1

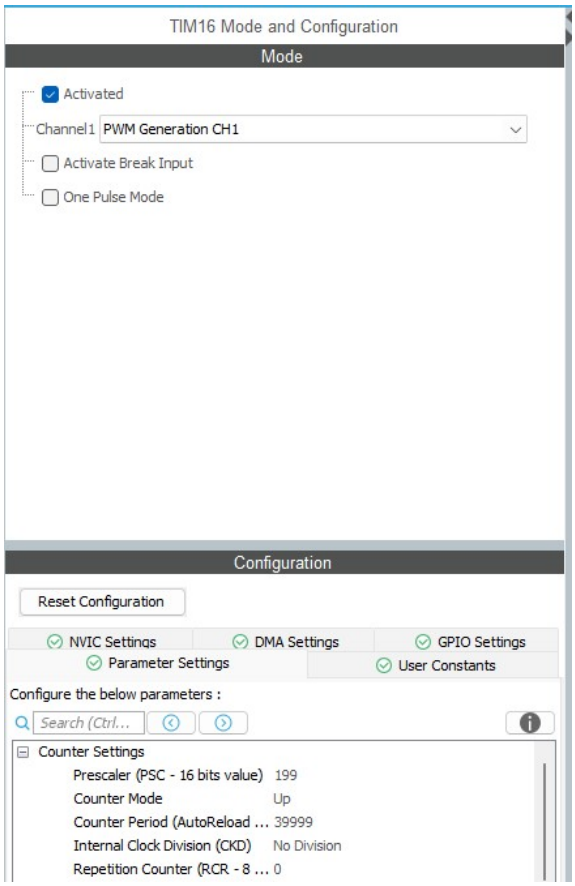

Figure 3: Trigger Pin - Timer 16 Channel 1

```
  /* USER CODE BEGIN 2 */
//  TIM11: Trigger
//  Trigger on for 10us and a period of 100ms.
//  80MHz/200(Prescal)=400KHz or 2.5us
//  2.5us*40,000(ARR) = 100 ms
//  CCR1 set to 3 making 2.5us*(3+1) = 10us pulse
  HAL_TIM_Base_Start(&htim16);
  HAL_TIM_Base_Start(&htim8);
  HAL_TIM_PWM_Start(&htim16, TIM_CHANNEL_1);
  HAL_TIM_IC_Start(&htim8, TIM_CHANNEL_1);
  TIM16->CCR1 = 3;
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    if (HAL_GetTick() - delay >= 100){
      delay = HAL_GetTick();
      echo1 = HAL_TIM_ReadCapturedValue(&htim8, TIM_CHANNEL_1);
      dist1 = echo1 / 58.0f;
      dist1_int_part = (int)dist1;
      dist1_frac_part = (int)((dist1-dist1_int_part)*1000);
//    snprintf is designed to safely format and store a string into a provided buffer, ensuring that the buffer's boundaries are respected, thus preventing buffer overflows.
      snprintf(buffer,sizeof(buffer), "echo = %d, distance = %4d.%04d\r\n",(int)echo1, dist1_int_part,dist1_frac_part);
      HAL_UART_Transmit(&huart2, (uint8_t *)buffer, sizeof(buffer), 1000);
    }
    /* USER CODE END WHILE */

```

Listing 1: main(void) code

Section 4 PWM - pulse width modulation

1. Find the STM32L476 datasheet by searching for STM32L476xx.

2. Create a new project.

Within the STM32 microcontrollers' documentation, a general purpose timer peripheral is always named "TIMx timer", where "x" can be any number and it does not reflect the number of timer peripherals embedded by a given microcontroller. For example, the STM32F100 microcontrollers embed a timer peripheral named TIM17, but the total number of timer peripherals embedded by these microcontrollers is less than 17.

3. Setup one of the 32 bit counters to turn drive an LED with a PWM signal. Set the on-time to 2 seconds and the period to 10 seconds.

One of the 32 bit counters shares resources with UART2. To use this timer, UART2 must be turned off.

Study the signal sent to trigger in Section 3. Modify the prescaler (PSC - 16 bits value) and Counter Period (AutoReload Register - 32 bits value) in the ioc and CCR1 (Capture/Compare Register) in main.

In compare mode the timer generates an output signal or trigger an event when the timer counter reaches a certain value. In this mode, the CCR holds the value to be compared with the timer counter. When the counter matches the value in the CCR, an output action (like toggling a pin, generating an interrupt, etc.) is performed.

4. Connect an external LED to the timer output pin. Verify on time and period.

5. Create a task that reads the blue user push button. When it is pressed change the LED on time to 8 seconds.

If the PWM period is fast enough, the flashing on and off of the LED will not be noticable. The brightness of the LED can be changed by changing the duty cycle.

6. Reduce the timer prescaler (PSC) to 79 (80MHz/(79+1)=1MHz). Reduce the autoreload register (ARR) to 99 ((99+1)/1MHz = 0.1ms period). Set the CCR to change between 19 and 79 when the button is pressed (20% to 80%).

7. To get a sign-off provide the total number of counters and demonstrate the LED changing brightness.

Section 5 Sign-offs

Name:_____

Section 3: Measuring distance with Ultrosonic Sensor. Date .

Section 4: Total number of counters and LED changing brightness. Date .