Assignment Measuring distance with Ultrasonic sensor (Timer Output/Input Compare/Capture) Embedded Systems CB Sem3

Goals

1. Understand input capture function of a timer
2. Handle different events in the interrupt handler
3. Handle timer counter overflow (and/or underflow)
4. Use a timer to measure the timestamp of a signal edge (rising or falling edge) external to the microcontroller

Lab Requirement

1. Measure distance using the ultrasonic sensor and calculate the accuracy
2. Something cool, such as printing distance to the LCD display

Background: Timer Interrupt and Timer Status Register (TIM_SR)

There are two special events:

- During up-counting, **CNT** restarts from 0 after it reaches the **ARR** value. This event is called ***counter overflow*.**
- During down-counting, **CNT** restarts from the **ARR** value after it reaches 0. This event is called ***counter underflow***.

When an overflow or underflow occurs, the timer can generate a timer interrupt if the Update Interrupt Enable (**UIE**) bit is set in the TIM DMA/Interrupt Enable Register (**TIM_DIER**).

In the interrupt service routine of the corresponding timer, you can check the timer status register (**TIM_SR**) to find out what events has generated the timer interrupt.

- If a counter overflow or underflow occurs, the Update Interrupt Flag (**UIF**) is set in **TIM_SR**. The **UIF** flag is set by hardware.
- If a channel is configured as input, a valid transition of an external signal can trigger the timer interrupt. Take channel 1 as an example. If the channel 1 is configured as input capture and the Channel 1 Interrupt Flag (**CC1IF**) in **TIM_SR** is set, the capture on Channel 1 has been triggered and the counter value has been copied to **CCR1** register. The **CC1IF** is set by hardware.

The timer interrupt service routine must clear these flags of **TIM_SR** to prevent it from being called again by the processor. The **CC1IF** flag is automatically cleared if the **TIM_CCR1** register is read. The **UIF** flag must be explicitly cleared by software.

Explanation and hints: Measuring distance with Ultrasonic sensor

Step 1: Set the microcontroller to use the 16MHz clock

Step 2: Set up GPIO PB6 to use TIM4_CH1

Step 3: Configure TIM4_CH1 for Input Capture, where, among the other settings, you do the following settings (Note: this is not the complete list of settings you need to make)

1. Enable the clock of timer 4
2. Set the prescaler register so that it divides the 16MHz clock down to 1MHz
3. Set the auto-reload register to the maximum value
4. Set the input filter duration to 0 in **`CCMR1`**
5. Set the capture to be on both rising and falling in **`CCER`**.
6. Set the input prescaler so that we capture each transition

   ….

Interface with Ultrasonic distance sensor

This part, which is the essence of the assignment, involves capturing a square wave generated by an HC-SR04 Ultrasonic distance sensor. We use PB5 to trigger the ultrasonic sensor and PB6 to capture the echo output of the sensor.

Sensor	Pin	Alternate Function
Trigger	PB 5	TIM3_CH2
Echo	PB 6	TIM4_CH1

Understanding how the sensor works

Documentation for this device can be found in Canvas, what follows is a brief summary.

- The sensor is powered by 5V. Connect the Vcc line to EXT_5V on the STM32 board, and GND to a ground connection on the board.
- While the board runs at 5V, it can be triggered by a 3.3V pulse. Its output is 5V, but many of the inputs on the STM32L board are five-volt tolerant and can handle a 5V input.
- As described in the documentation, to activate the sensor send a high pulse of at least 10µs to the Trigger input. An ultrasonic burst of 40kHz will be emitted, and then the device will return a square wave proportional to the distance to the nearest object.
- The return will be on the ECHO pin, a square wave ranging from 150µs to 25ms (38ms if nothing is in range). To convert this value to cm, divide the time in µs (microseconds) by 58.

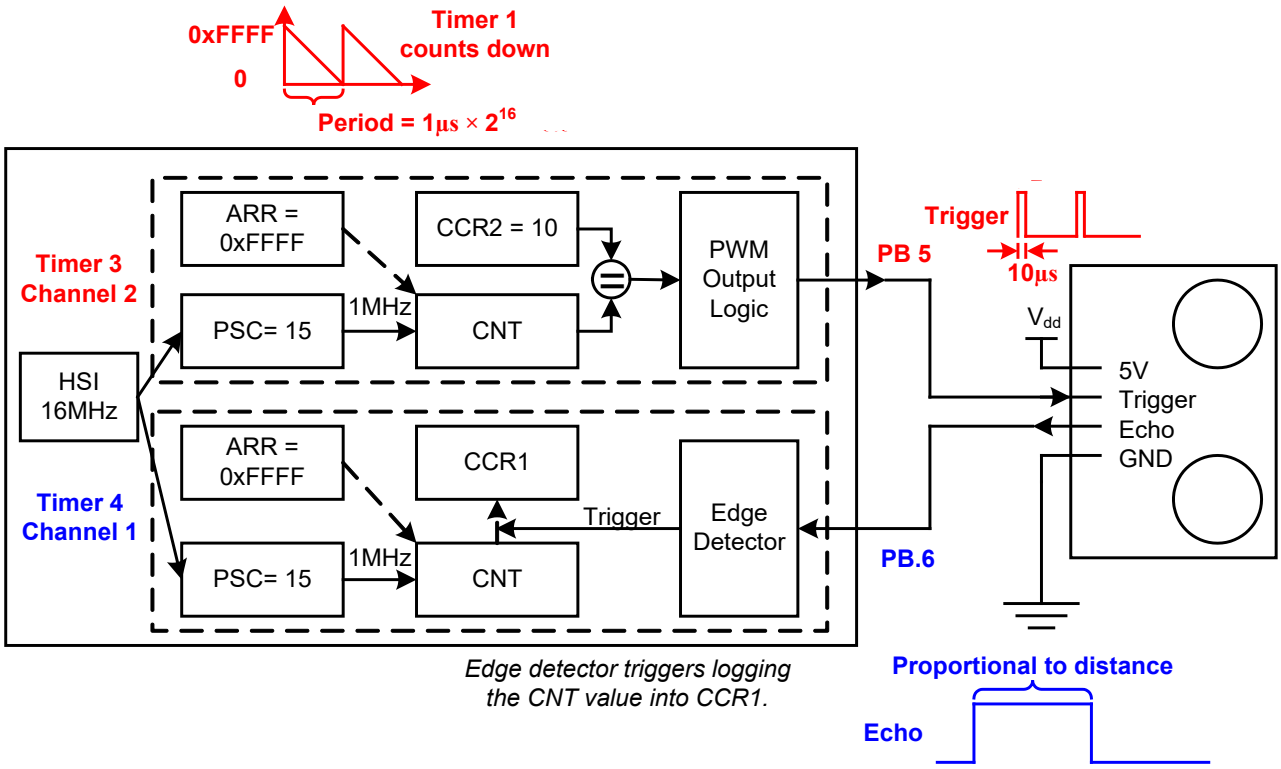

Figure 1. Connection and configuration diagram for interfacing ultrasonic distance sensor

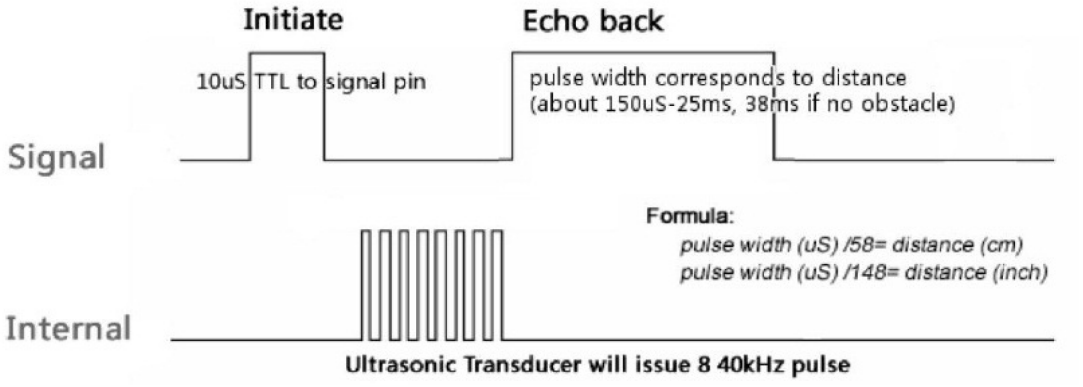

Figure 2. Trigger signal, internal burst with 8 40kHz pulse, and the echo signal

Generating trigger signal

Generate the 10μs pulse on trigger on PB5 (give your calculation in the document). This will internally trigger an ultrasonic burst of 40kHz audio.

- If you want to verify the signal is working, hook it to a logic analyzer (or oscilloscope) and make sure you are getting a 10 μs pulse every 65 ms on PB5.

Capturing the echo signal and computing the distance

Conduct timer capture on PB6 to capture the return signal. The result will be a square wave proportional to the distance to the nearest object.

- The return on the ECHO pin will range from 150 μs to 25 ms (38 ms if nothing in range).
- To calculate the distance:

$$distance\ (in\ cm) = \frac{time\ (\mu s)}{58}$$

- Store the result in an integer variable.
- You can check the result in the debugger. Directions for this should be similar to those for measuring the 1Hz signal.

Optional: Interface with Ultrasonic distance sensor

Do something cool! You can come up with something on your own, but here is a list of ideas you can use.

- Print the distance to the LCD
- Have the LEDs turn on/off at certain distance (for example: put your hand closer than 1m and the red LED goes on, closer than 0.5 m and the green LED goes on)

Submission

The grading for this challenge will be based on your design, implementation and documentation. The reader should be able to verify the design process and the design choices based on objective arguments, test results, measurements, etc. Use pictures and diagrams to clarify your setup / measurements and design. Fully functional code without the required documentation has not much value.

Your submission is a zip file that consists of:

1. All code
2. Proper documentation consisting of:
   2.1. Title page with date and name
   2.2. Short introduction
   2.3. Your design: all diagrams with short description and well justified design choices
   2.4. Your calculations for the values you use to configure timers in your design
   2.5. Your testing strategy: How you tested your implementation including proof (how you validated the results and so on)
   2.6. Reflection on what you have learned
   2.7. Bibliography (sources) in APA style (https://www.bibme.org/citation-guide/apa/)
3. A short video (max 4 min) where you demonstrate and clearly explain your solution