Assignment – ESE S4 Embedded Systems Software – week 10

Interfacing to an Ultrasonic Range Finder

In this week's lab, we will use a timer module to both generate an output pulse and measure an input pulse. The output pulse will trigger an ultrasonic range finder to make a measurement which will be reported back to the STM microcontroller as a pulse with a variable width. The software you will write will measure the width of this returned pulse and display the distance to an object in front of the sensor.

Portions of this lab document come from Zhu

- Lab_09_TIM_InputCapture (Ultrasonic Distance Sensor)

As always, when building circuits and when working with active circuits, wear safety glasses.

Due: ***Monday*** 6 April 2020
Submit your code listing, schematic, and required scope screen shots to the eConestoga dropbox by 11:59 pm

Submit lab report to dropbox including:

- Your commented c source code and header files
- Schematic (sketch) showing the connections from the Discovery board to the ultrasonic sensor

Demo: video, zoom meeting (demo on your webcam or screen share)

References:

Zhu, Chapter 15 General-purpose Timers. Esp. 15.4, 15.4.2

The basic steps of the lab are:

1. Connect an external LCD module to your Discovery Board as covered in previous labs. Use one of these previous Keil projects as a starting point. Alternatively, you can display the distance data on a terminal window on your computer using the virtual serial port in STlink.

2. Connect an ultrasonic distance sensor to the STM32 timer module input and output pins as specified below. Write software to generate a trigger signal and measure the echo generated. The echo measurement code ***must*** use interrupts.

3. Write software that will initialize all the hardware and then loop continuously reading the echo pulse width. Calculate the distance to the object in cm and display the result on the display.

You will need to write the following:

- A function to initialize a timer channel to generate a trigger signal as outlined below

- A function to initialize a timer channel for input capture as outlined below.

- An interrupt service routine to respond to the input capture event and calculate the length of the measured echo in timer ticks.

- A main() to do the following:

    - Call functions to initialize the ports, LCD module and/or serial port, timer modules, and NVIC for interrupts.

    - Loop forever doing the following:
        - Reading the echo pulse width from a global variable updated by an ISR
        - Calculate the corresponding distance to the object in front of the sensor in cm (do this in the mainline since ISR's are supposed to be fast and not do floating point math)
        - If the pulse width indicates that the sensor didn't receive an echo, display a "no echo" message on the LCD. Otherwise, display the measured distance value in cm.

To receive maximum credit, use good coding style, appropriately separate your code into modules, and comment as outlined for previous labs.

Interface with Ultrasonic distance sensor

To determine the distance of an object from the ultrasonic distance sensor, your software will need to capture a square wave generated by an HC-SR04 Ultrasonic distance sensor. We will use PB.6 to trigger the ultrasonic sensor and PA.0 to capture the echo output of the sensor.

	Pin	Alternative Function
Trigger	PB.6	TIM4_CH1
Echo	PA.0	TIM2_CH1

- The sensor is powered by 5V. Connect the Vcc line to EXT_5V on the STM32L board, and GND to a ground connection on the board.
- While the sensor 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 sensor documentation, to activate the sensor send a high pulse of at least 10us to the Trig 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 150us to 25ms (38ms if nothing is in range). To convert this value to centimetres use the speed of sound (343 m/s) to determine the total time for the sound to reach the target object and echo back, and then divide by 2.

Trigger the Sensor

Configure Timer 4, Channel 1 as the trigger output. Modify your RC Servo code to generate a 10 us pulse every 0.5 s (500,000 us). You will need to change the clock period so that the 16 bit timer can count to 0.5 s. This trigger will free-run once configured and started.

Measure the Echo Pulse Width

Configure Timer 2, Channel 1 as the echo input. Use slave mode with reset as outlined in Section 15.4.2 of the textbook. Use a timer clock frequency of 1 MHz so the timer value that you read back will be directly in microseconds. Configure the timer channel to generate an interrupt request on the completion of each echo pulse measurement, and the NVIC to respond to this interrupt request.

In the interrupt service routine for Timer 2, read the CCR1 register to automatically clear the interrupt request flag and get the width of the echo pulse. Put this value into a volatile global variable that will be read by the foreground loop calculate and display the distance to the target.

Configure Timer 2 Channel 1 as input capture using Slave mode, 1MHz clock

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UIFREMAP	Res	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Value																																
0x04	**TIMx_CR2**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI1S	MMS[2:0]			CCDS	Res	Res	Res
	Value																																
0x08	**TIMx_SMCR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res	SMS[2:0]		
	Value																																
0x0C	**TIMx_DIER**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Value																																
0x10	**TIMx_SR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC4OF	CC3OF	CC2OF	CC1OF	Res	Res	TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Value																																
0x14	**TIMx_EGR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TG	Res	CC4G	CC3G	CC2G	CC1G	UG
	Value																																
0x18	**TIMx_CCMR1** Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC2M[3]	Res	Res	Res	Res	Res	Res	Res	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Value																																
	TIMx_CCMR1 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Value																																
0x1C	**TIMx_CCMR2** Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC4M[3]	Res	Res	Res	Res	Res	Res	Res	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Value																																
	TIMx_CCMR2 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Value																																
0x20	**TIMx_CCER**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC4NP	Res	CC4P	CC4E	CC3NP	Res	CC3P	CC3E	CC2NP	Res	CC2P	CC2E	CC1NP	Res	CC1P	CC1E
	Value																																