EE 337: Introduction to Timers
Lab 5

29th August, 2016

This set of experiments has the following objectives:

- Familiarization with the 8051 timers
- Familiarization with interrupts in 8051
- Using timers to estimate signal pulse width
- Using timesr to generate pulses of variable width (PWM generation)

Before attempting these exercises, please go through the notes on timers and interrupts uploaded on moodle.

Introduction

The 8051 microprocessor can interact with the built-in timers either using polling or interrupts. Our objective is to generate a hardware delay using one of the timers (T0, T1, T2), say T0.

1. To achieve a specified delay of D secs, you need to set a value in TH0 and TL0 registers corresponding to the desired `count` cycles.

   Note: Refer to Sec. 4 in the handout on *8051 Timers*.

2. Once this is done and the timer is initialized appropriately and enabled to run, the timer starts its counting process. When the desired number of `count` cycles are executed, the TH0, TL0 registers overflow (FFFFH to 0000H) and the corresponding TF0 flag will be set.

   Note: Refer to Sec.1 (skip Sec.1.1, Sec. 1.2), Sec. 2 in the handout on *8051 Timers*.

For the microprocessor to make use of this delay, there are two possible approaches, which can be either to use polling or interrupt.

- In the polling approach, the microprocessor continuously checks for the status of the TF0 flag to check if the desired delay has been achieved.
- In the interrupt approach, as soon as the TF0 flag is set the timer initiates an interrupt to the microprocessor. Assuming proper initialization of interrupts, the microprocessor recognizes this interrupt from the timer and acts appropriately.

  Note: Refer to the handout on *8051 Interrupts* (skipping Sec.5 for now).

The difference between using an interrupt and polling is, in case of interrupts the microprocessor can continue with its task and suspend temporarily when an interrupt occurs. Whereas, polling requires microprocessor to continuously check for a flag, thus making it busy.

Lab Work:

Delay generation

1. Write code to generate a delay of 1 sec. using T0 in **Mode 1** and polling. *Note:* A single overflow with the maximum count will not give a 1 sec. delay.

2. Write code to generate a delay of 1 sec. using T0 in **Mode 1** and interrupt. *Note:* Again a single overflow with the maximum count will not give a 1 sec. delay and hence an appropriate interrupt service routine (ISR) has to be written.

Pulse width measurement

1. Write code to measure the pulse width (T_{on}) of an external signal connected to the P3.2 of the micro-controller and display it on the LCD.

   (a) We wish to use the gating feature of timer to measure pulse width of an input signal. Configure T0 to be gated by an external signal. *Note*: Refer to Sec. 2 and Sec.6 of the handout on *8051 Timers*.

   (b) Ensure that the timer starts at the rising edge and detects the falling edge using the negative edge triggered interrupt (INT0). To achieve this appropriate configuration of interrupts is required.

   (c) Write an ISR for INT0 to display the count corresponding to the pulse width. The display on the LCD should show "PULSE WIDTH" on the first line and "COUNT IS XXXXXX" on the second line. (Assuming T0 is used, display of count should be of 6 digits: first 2 digits showing the number of times the timer has overflown, next 2 digits showing TH0 value, and the next 2 digits showing TL0 value).

   (d) Find the pulse width in seconds (corresponding to the COUNT), maximum and minimum possible pulse widths that can be measured using this setup. Check these values with your RA/TA.