EE 337: Introduction to Timers
Lab 5

29^{th} 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 timers to generate pulses of variable width (PWM generation)

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

1 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.

2 Lab Work

2.1 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.

2.2 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.

3 Post-Lab Work

The 89C51 device in Pt51 has a Timer 2 (T2) in addition to the T0, T1 timers. This T2 can generate PWM signals over a range of frequencies, with 50 percent duty cycle (i.e., $T_{on} = T_{off}$). By setting appropriate values in T2 related registers, a PWM signal of some specified frequency will be generated on the P1.0 pin of the device.

For details about configuring T2 to generate a PWM, refer to the section "PROGRAMMABLE CLOCK OUT" in the handout on *Introduction to Timer-2 in 8052*, uploaded in moodle.

In this part of the lab, we wish to measure the pulse width of the PWM output from the T2 timer. One such application of this would be speed-control of motors, i.e., where you desire a motor to run at a specific speed, and monitor or control to maintain that speed.

This can be done using the following steps:

Step. 1 Configure T2 to generate a 1kHz PWM signal. Verify the frequency by observing the PWM signal on a DSO. *Refer to Sec. 3.1 for template code.*

Step. 2 Modify your code for pulse width measurement (written in Sec. 2.2) to measure the pulse width of the PWM generated in Step 1 (i.e., using T2).

Repeat the above steps for PWM signals of maximum and minimum possible frequencies in Pt51.

3.1 Template code

Use the following code for T2 initialization and PWM generation.

```
; Defining Timer-2 registers
 T2CON   DATA    0C8H
 T2MOD  DATA  0C9H
 RCAP2L  DATA    0CAH
 RCAP2H  DATA    0CBH
 TL2     DATA    0CCH
 TH2     DATA    0CDH
; Defining interrupt enable (IE) bit
 ET2     BIT     0ADH
; Defining interrupt priority (IP) bit
 PT2     BIT     0BDH
; Defining P1
 T2EX    BIT     91H
 T2      BIT     90H
; Defining timer control (T2CON) register bits
 TF2     BIT     0CFH
 EXF2    BIT     0CEH
 RCLK    BIT     0CDH
 TCLK    BIT     0CCH
 EXEN2   BIT     0CBH
 TR2     BIT     0CAH
 C_T2    BIT     0C9H
 CP_RL2  BIT     0C8H

 ORG 0000H
 LJMP MAIN

 ORG 0100H
;Timer-2 initialization
 T2_INIT:
 T2MOD,#XXH;
 CLR/SETB EXF2;

/* Next, disable baud rate generator */

/* Next. ignore events on T2EX */


;Initialize values in TH2, TL2 depending on required frequency
 MOV TH2,#XXH;     /* Init msb_value */
 MOV TL2,#XXH; /* Init lsb_value */
;Reload values in RCAP
 MOV RCAP2H,#XXH;     /* reload msb_value */
 MOV RCAP2L,#XXH;     /* reload lsb_value */
 CLR/SETB C_T2;             /* timer mode */
 CLR/SETB CP_RL2;           /* reload mode */
 CLR/SETB ET2;              /* clear timer2 interrupt */
 RET
```

```
;--------------- MAIN STARTS HERE --------------------
 MAIN:
;Port initialization
 LCALL T2_INIT
 OVER: SJMP OVER

 END
```