# NPTEL-WEL Summer Workshop on Micro Controller

Lab-3:Timer Programming

In this set of experiments, we learn how to use built-in hardware timers in 8051 to generate delays without tying up the processor.

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

1. Write a subroutine which will use a 16 bit value stored at 81H/82H in the indirectly addressable memory to program the timer T0 in order to generate a delay proportional to this count.

   Recall that the 8051 timers count *up*. These generate an interrupt (if enabled to do so) when the count wraps around from 0FFFFH to 0000. If we want a timer to time out after n cycles, it should be loaded with -n (2's complement of n).

   So the subroutine should subtract the stored 16 bit number from 0000H and load the result as the initial count in T0.

   (While debugging the program with single stepping, you could initialize locations 81H/82H with 0001, which results in loading the timer with -1 = 0FFFFH, so that it overflows at the first increment. In actual use, a different count will be stored, of course.)

2. Write a program which will use the above subroutine to blink LEDs such that these are ON for one second and OFF for one second endlessly. Adjust the timer count and the number of times the delay subroutine is called to make the ON and OFF period as close to 1 second as possible. (Measure the time over a large number of cycles to make this adjustment).

3. Program timer T0 to generate the maximum delay and cause an interrupt on time out. The interrupt service routine should increment a counter till it reaches 40. Every time the count reaches 40, an LED should be toggled and the count reset to 0.
   By measuring the time taken by the LED to toggle 20 times, adjust the timer count, such that the toggle time is as close to 1 second as possible. (20 toggles should take 20 seconds).

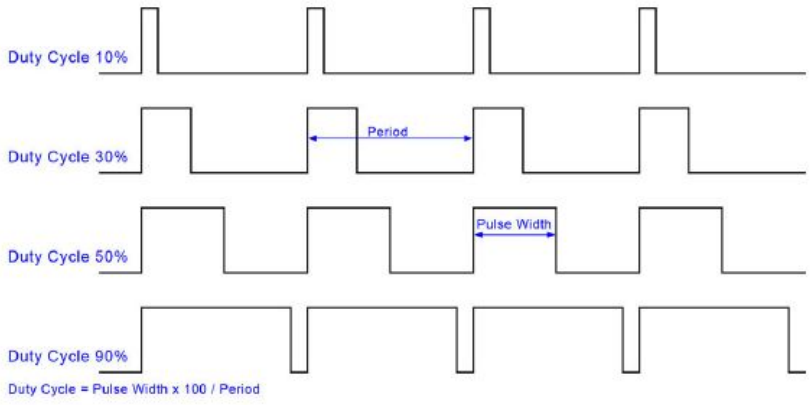


Figure 1: Pulse width modulation

| P1.0-2 | Duty Cycle(%) |
|---|---|
| 000 | 20% |
| 001 | 30% |
| 010 | 40% |
| 011 | 50% |
| 100 | 60% |
| 101 | 70% |
| 110 | 80% |
| 111 | 90% |

Table 1: Mapping of switch positions to duty cycle.

## Pt-51 clock

(a) You will configure the timer T0 to generate a pulse width modulated (PWM) signal whose duty cycle is controlled by the user. Figure 1 shows typical PWM signals. Read the switches P1.0-P1.2 to accept the duty cycle from the user. Write a program to generate a PWM with the duty cycle corresponding to the switch positions as indicated in Table 1.Set the frequency of PWM signal be 0.5 Hz and use the LEDs P1.4-P1.7 (though one is sufficient) to monitor the generated output. The PWM information has also to be written out on the LCD in the following format. First row of the LCD should display the duty cycle in percent as D̈uty cycle: xx ;̈ where xx depends on the switch value. Second row should display the pulse width as P̈ulse width: tttt;̈ where tttt represents tttt x $10^{-3}$s.

(b) Using the 1 second timer developed as homework, implement a clock which displays the time on the LCD unit. (This will involve separate memory locations for seconds, minutes and hours). At the end of 1 second interval, advance the seconds count.

When it reaches 60, reset it to zero and advance the minutes count. when that reaches 60, reset it to zero and advance the hours count. Reset the hours count to 01 when it reaches 13. Initialize the hours, minutes and seconds count to 12, 59 and 30 respectively. Display the time on the LCD in hh:mm:ss format.

(c) Use timer T1 to debounce the switches on Pt51 board. Program T1 for a delay of 20 ms. Read the switch twice at intervals of 20 ms. Reject the new value if it is different and re-read after another 20 ms. If two values are the same, accept it and report it as the switch status.

(d) Use the debounced switches to implement a facility for setting the clock. If Switch 0 is on, the program should enter the time set mode. In the time set mode, every toggle on switch 1 should advance minutes by 1 while every toggle on switch2 should advance the hours by 1. Returning switch 0 to off should start showing time from the set value onwards and the usual clock display as implemented in the earlier experiment should run. In this mode, check the status of switch 0 every minute and go to the set mode if switch 0 is ON.