SECTION OVERVIEW

Complete the following objectives:

- Learn how to configure and use the 8-bit Timer/Counters on the ATmega128 to generate pulse-width modulation (PWM) signals.
- Use the upper half-byte and lower half-byte (otherwise known as the upper and lower nibbles) of an I/O port for two different tasks.

PRELAB

To complete this prelab, you may find it useful to look at the full ATmega128 datasheet. If you consult any online sources to help answer the questions below, you must list these sources as references in your prelab.

1. List the correct sequence of AVR assembly instructions needed to store the contents of registers R25:R24 into Timer/Counter1’s 16-bit register, TCNT1. (You may assume that registers R25:R24 have already been initialized to contain some 16-bit value.)

2. List the correct sequence of AVR assembly instructions needed to load the contents of Timer/Counter1’s 16-bit register, TCNT1, into registers R25:R24.

3. Imagine Timer/Counter0 (an 8-bit timer) has been configured to operate in Normal mode, and with no prescaling (i.e., \( \text{clk}_{T0} = \text{clk}_{I/O} = 16 \, \text{MHz} \)). The decimal value “128” has just been written into Timer/Counter0’s 8-bit register, TCNT0. How long will it take for the TOV0 flag to be set? Give your answer as an amount of time, not as a number of cycles.

BACKGROUND

Timer/Counters are one of the most versatile and commonly-used components of a microcontroller, and have many uses in embedded applications. In basic terms, a Timer/Counter is simply a register whose contents are regularly incremented (or decremented) at a specified, configurable frequency. If you know the rate at which this incrementing or decrementing is occurring, then you can compare the contents of the Timer/Counter’s register at two different points in a program, and calculate how much time has elapsed!

Together, the width (in bits) of the Timer/Counter and the frequency of the clock supplied to the Timer/Counter determine its range (the largest interval that can be measured) and resolution (the smallest interval that can be measured). The ATmega128 microcontroller features two 8-bit Timer/Counters and two 16-bit Timer/Counters, which can each be configured to run at one of several available clock frequencies.

Beyond simply measuring an interval of time, Timer/Counters can also be configured to take some action based on an observed interval, such as toggling an I/O pin after a certain amount of time has passed. Changing this interval has the effect of varying the duty cycle of the waveform being generated by the I/O pin - this is a technique known as pulse-width modulation (PWM). For more details on how the ATmega128 microcontroller can generate PWM signals, please refer to Timer/Counter0’s “Fast PWM Mode” subsection on pages 98-99 of the ATmega128 datasheet.

PROCEDURE

For this lab, you will need to write an assembly program that allows the speed of the TekBot to be adjusted based on user input on Port D. The speed itself will be modified by using both of the 8-bit Timer/Counters in Fast PWM mode, and driving the right and left Motor Enable port pins with the PWM signals created by the Timer/Counters. By varying the duty cycle of the output PWM waveforms, you will be able to modify the speed of the TekBot’s motors.

You are not required to implement any BumpBot behavior in this lab, so the TekBot can just be configured to move forward indefinitely.

Some design decisions have been left up to you for this lab (such as how to read the Port D pushbuttons), but you must adhere to the following requirements:

1. Your TekBot needs to have sixteen equidistant speed levels, with Speed Level 0 being completely stopped, Speed Level 15 being full speed, and Speed Levels 1 through 14 in between. Refer to Table 1 (see next page) for more details.

2. A user must be able to modify the speed in 4 different ways: (1) increase speed by one level, (2) decrease speed by one level, (3) immediately increase
<table>
<thead>
<tr>
<th>Speed Level</th>
<th>TekBot Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>100% (255/255)</td>
</tr>
<tr>
<td>14</td>
<td>~93.3% (238/255)</td>
</tr>
<tr>
<td>13</td>
<td>~86.7% (221/255)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>~13.3% (34/255)</td>
</tr>
<tr>
<td>1</td>
<td>~6.7% (17/255)</td>
</tr>
<tr>
<td>0</td>
<td>0% (0/255)</td>
</tr>
</tbody>
</table>

Table 1: Equidistant Speed Levels

speed to the highest level (full speed), and (4) immediately decrease speed to the lowest level (stopped).

3. A single button press on Port D must result in a single action; for example, if the user presses the “decrease speed by one level” button, your program should smoothly decrease the speed by just one level, not several levels in rapid succession.

Unlike the previous lab, it is acceptable to use a small delay from within an ISR to meet this requirement. (A solution that meets this requirement while avoiding the use of within-ISR delay is still preferable from a design perspective, but not required to receive full credit for this lab.)

4. The speed levels must not wrap around from 15 → 0 or 0 → 15. In other words, if the user requests a speed increase and the TekBot is already at full speed, the speed must not wrap around to a lower speed. Similarly, the TekBot must not go from stopped to a higher speed if a decrease is requested.

5. For the user to visually assess the current speed level of the TekBot, you must use the mega128 LEDs connected to the lower nibble (pins 3:0) of Port B to display a 4-bit indication of the current speed level. For example, “0000” (all 4 LEDs off) will represent Speed Level 0, and “1111” (all 4 LEDs on) will represent Speed Level 15. This 4-bit indication of speed level must not interfere with the motor control signals on pins 7:4 of Port B.

STUDY QUESTIONS / REPORT

A full lab write-up is required for this lab. When writing your report, be sure to include a summary that details what you did and why, explains any problems you may have encountered, and answers the study questions given below. Your write-up and code must be submitted by the beginning of next week’s lab. Remember, NO LATE WORK IS ACCEPTED.

Study Questions

1. In this lab, you used the Fast PWM mode of both 8-bit Timer/Counters, which is only one of many possible ways to implement variable speed on a TekBot. Imagine instead that you used just one of the 8-bit Timer/Counters in Normal mode, and every time it overflowed, it generated an interrupt. In the overflow ISR, you manually toggled both Motor Enable pins of the TekBot, and wrote a new value into the 8-bit Timer/Counter’s register. (If you used the correct values, you would essentially be manually implementing a fixed-frequency, variable duty-cycle output signal.)

Provide your best assessment (in 1-2 paragraphs) of the advantages and disadvantages of this new approach, in comparison to the original PWM approach used in this lab.

2. The previous question outlined a way of using a single 8-bit Timer/Counter in Normal mode to implement variable speed. How would you accomplish the same task (variable TekBot speed) using one or both of the 8-bit Timer/Counters in CTC mode? Provide a rough-draft sketch of the Timer/Counter-related parts of your design, using either a flow chart or some pseudocode (but not actual assembly code).

CHALLENGE

To receive extra credit for this lab, you will need to implement a new feature that supplements your regular Lab 7 code. Use one of the 16-bit Timer/Counters to keep track of how many seconds have elapsed since the TekBot’s speed has been modified, and continually display the most up-to-date count as a 10-bit binary number on the LCD.

Demonstrate the correct operation of this new feature to your TA, explain your design in the lab write-up, and be sure to also turn in your challenge .asm file as part of the online submission process.