SECTION OVERVIEW

Complete the following objectives:

- Learn to utilize the 16 bit Timer/Counter1.
- Write AVR assembly code to transmit precisely timed patterns.
- Combine the timing and individual patterns to create Morse code messages.

PRELAB

If you consult any online sources to help answer the prelab questions, you must list them as references in your prelab.

1. In this lab, you will be utilizing Timer/Counter1, which can make use of several 16 bit timer registers. The datasheet describes a particular manner in which these registers must be manipulated. To illustrate the process, write a snippet of assembly code that configures OCR1A with a value of 0x3FA5. For the sake of simplicity, you may assume that no interrupts are triggered during your code’s operation.

BACKGROUND

Morse code is a communication method that commonly uses sound or light to transmit messages. The messaging scheme utilizes a system of “dots” and “dashes”. For example, if someone is transmitting Morse code via sound, then a short beep would represent a dot, and a longer beep would represent a dash. If they are utilizing light, then a short pulse of light would be a dot, and a longer pulse of light would represent a dash. Individual characters of the alphabet are encoded as a predetermined pattern of dots and dashes. In this project, we will utilize the Morse code standard that is published by the ITU (International Telecommunication Union). As an example, Table 1 illustrates Morse code patterns that are used for the letters A through D.

<table>
<thead>
<tr>
<th>Letter</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>dot dash</td>
</tr>
<tr>
<td>B</td>
<td>dash dot dot dot</td>
</tr>
<tr>
<td>C</td>
<td>dash dot dash dot</td>
</tr>
<tr>
<td>D</td>
<td>dash dot dot</td>
</tr>
</tbody>
</table>

Table 1: Morse code patterns corresponding to the letters A - D

For a full table of Morse code characters, please see this image from Wikipedia. In this project you will be designing a system to broadcast Morse code messages using the LED lights that are on the ECE375 board. Unless you are implementing the challenge code, we will only use the capital letters A-Z (no numbers, symbols, or other content).

IMPLEMENTATION

In the real world we might want a really big spot light that we could use to transmit Morse code messages. Since we don’t have a spot light, we will use the LEDs on our microcontroller board. When a dot or dash is active, you will simultaneously enable the 3 LEDs which are connected to PB5-PB7. The 3 LEDs will be disabled during a pause, and then re-enabled during the next dot or dash.

Functionality

When the CPU firsts boots, the LCD screen should show the following content to the user:

Welcome!
Please press PD0

This content will remain indefinitely until the user presses the button which is connected to Port D, pin 0. After the button is pressed, this information will be displayed on the screen:

Enter word: A

The second line of the LCD is where the user is able to select the letters that they want to send. This information will be provided by choosing one character at a time.
• Pressing PD7 changes the current character and iterates through the alphabet in reverse order.
• Pressing PD6 changes the current character and iterates through the alphabet in forward order.
• Pressing PD0 confirms the current character and moves to the right.

Special case: if 16 characters have already been selected, pressing PD0 will immediately begin the transmission.
• Pressing PD4 immediately begins transmission of the message that is displayed.

For example, if the display currently shows:
Enter word: A
then pressing PD6 will immediately update the screen to display:
Enter word: B
pressing PD6 again will display:
Enter word: C
if we now press PD7, the LCD will display:
Enter word: B
if we press PD7 two more times, the LCD will display:
Enter word: Z
if we press PD0, the LCD will display:
Enter word: ZA
At this point, we have confirmed our selection of the 'Z' and we can’t go back to change it. However, we are now able to select the second letter. Pressing PD6 four times will result in this:
Enter word: ZE
if we press PD0:
Enter word: ZEA
if we press PD6:
Enter word: ZEB

Finally, the exciting part! We press PD4 and two things immediately happen:
1. All three LEDs connected to PB5-PB7 immediately light up and begin broadcasting the series of dots and dashes that represent ”ZEB”. The lights are illuminated during a dot or a dash, and are disabled during the pauses.
2. The LED on PB4 should be illuminated during the entire transmission to indicate that a message is being sent. Even during a pause, PB4 will remain active.

After the entire message has been transmitted, all LEDs (PB4-PB7) will be disabled. The code will return to the prompt as shown:
Enter word: A
The user can now enter another message to transmit, exactly as before.

Additional Notes
• Once a character has been confirmed by pressing PD0, there is no capability to go back and change a character that we already selected.
• The LCD display needs to be updated immediately whenever the user provides input.
• You are only expected to broadcast the capital letters A-Z. Do not include support for spaces, numbers, or any symbols (unless you are implementing the challenge code).
• The alphabet needs to wrap around if the user keeps pressing the same button. For example, if the user is pressing PD6, the characters should eventually go in the order: A, B, C, D, E, ... X, Y, Z, A, B, C...
If the user is pressing PD7, the order will be reversed: A, Z, Y, X, W, ...
The code needs to correctly handle any sequence of button presses.

You **must use the Timer/Counter1 module** to manage the Morse code unit timing. You may design your code to use polling or you may use interrupts (either approach is fine). You may not utilize any busy loop for the Morse code delay, although it is allowed to loop if you are monitoring an interrupt flag.

Do not include switch debouncing delays of more than 150ms. A busy loop for debouncing is okay.

Ignore all button inputs while the Morse code message is being transmitted.

Write your code so that it works correctly even if non-relevant buttons are held down (for example, you should be able to hold down PD5 while simultaneously selecting characters with PD6).

The LCD screen must never display symbols, gibberish, or other undesired output.

**Timing Information**

Proper implementation of Morse code is very sensitive to timing. Your project needs to follow these guidelines exactly:

All timing will be based on the concept of a “unit” of time. For purposes of this lab, a **unit of time is equal to one second**. You must configure Timer/Counter1 so that your unit delay is within 100 microseconds of one second.

As taken from the ITU Morse code standard, we have the following standards:

- A dot is one unit of time
- A dash is three units of time
- The space between parts of the same letter is one unit
- The space between letters is three units

If we wish to transmit the message “AD” then we will perform the following steps:

Note that “AD” is: dot dash dash dot dot

- Illuminate the LEDs for three seconds (dash)
- Disable the LEDs for one second (space between parts of the same letter)
- Illuminate the LEDs for one second (dot)
- Disable the LEDs for one second (space between parts of the same letter)
- Illuminate the LEDs for one second (dot)

**Hints**

- Students often ask about the behavior of the LEDs which are connected to PB1-PB3. In some cases the LCD library will illuminate one or more of those LEDs (even if you don’t want them to be enabled). You do not need to worry about this. The project guidelines do not dictate the behavior of any LEDs which aren’t mentioned in the project instructions.
- Remember that PD4-PD7 are NOT implemented as interrupts in the AT-mega128. You will need to use polling in order to check any of these inputs.

**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**, and explains any **problems you encountered**. Your write-up and code must be submitted prior to your final (Week 10) lab session. As usual, NO LATE WORK IS ACCEPTED.

**Study Questions**

No additional questions for this lab assignment.

**CHALLENGE**

In order to earn extra credit, incorporate all of the following features:

Suppose that the user is at the default prompt as follows:

**Enter word:**

A

If the user holds down PD0 for at least two seconds, you will enter a special “Enhanced Mode”. To indicate that the mode is operational, the LCD prompt will change to the following:
ENHANCED MODE:

Note that PD0 retains its normal behavior (confirming the selected character) if the user presses the key for less than 2 seconds. Enhanced mode is ONLY entered after the key is held down for at least 2 seconds.

While the program is operating in enhanced mode, its behavior will remain nearly identical to the original implementation with one small change... the user must now be allowed to choose numeric characters from the selection menu. In other words, if enhanced mode is active, the user can use buttons PD6 and PD7 to select characters A, B, C, D, ... , W, X, Y, Z, 0, 1, 2, 3, ... 8, 9, A, B, ...

In the base implementation, the character selection menu must rollover from Z to A (and vice versa). While operating in enhanced mode, the character selection menu will include the numeric digits before rolling over to A (and vice versa).

As mentioned before, all of the base functionality must be retained. The user can select letters, confirm them, and initiate transmission of the Morse code sequence. Naturally, your code must transmit the proper Morse code sequence even when numeric characters are included in the message.

Additional Notes

- You must use Timer/Counter3 while determining whether PD0 was held down for at least two seconds. Using other timers (or a busy loop) is not allowed. You will not receive bonus credit if you fail to heed this requirement.
- If the program is operating in “normal mode” the user should not be able to select numeric characters such as 0 - 9.
- Even if you decide to implement “Enhanced Mode” you must still support all of the original program requirements that were provided in this lab document.
- The user can choose to return to “normal mode” at any time while they are not transmitting a Morse code message. In order to do so, the user simply needs to hold down PD0 for at least two seconds. When normal mode is re-entered, the LCD panel should return back to the original prompt as shown:

```
Enter word:
A
```

As one would expect, the numeric characters will no longer be available for selection after the user has returned to normal mode.