Final Project

Submit Assignment

Due Thursday by 11:59pm **Points** 156 **Submitting** a file upload **File Types** pdf and asm

ECE375 Final Project

Submit your work by 11:59pm on Thursday, December 10th, 2020.

Late submissions will be deducted 25 points per day.

Reminder: please check the errata included at the bottom of this page.
Your submission should consist of two items:

1. *a single .asm file containing your AVR assembly source code (I will provide the LCD driver file)*
2. *a PDF document containing your typed report*

Your grade will be reduced if you do not use the assembly code template file that is provided.
Please take the time to read the entire documentation.

Introduction

Morse code **(https://en.wikipedia.org/wiki/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, the table below illustrates Morse code patterns that are used for the letters A through D.

Letter	Pattern
A	dot dash
B	dash dot dot dot
C	dash dot dash dot
D	dash dot dot

For a full table of Morse code characters, please see **this image from Wikipedia (https://upload.wikimedia.org/wikipedia/commons/b/b5/International_Morse_Code.svg)** .

In this project you will be designing a system to broadcast Morse code messages using the LED lights that are on the ECE375 board. We will only use the capital letters A-Z (no numbers, symbols, or other content).

Project 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 will only be broadcasting the capital letters A-Z. Do not include support for spaces, numbers, or any symbols.
- 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, Z, ...
- The code needs to correctly handle any sequence of button presses
- You **must use the Timer/Counter1 module** to manage the Morse code 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 10ms. 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, hold down PD5 while trying to select characters with PD6)
- PD0, PD4, PD5, PD6, and PD7 must be configured in input mode with the internal pull-up resistors enabled. Note: PD5 will only be used during testing to verify that your code is correctly ignoring its value.
- 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. 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

For example purposes, let's assume that a unit of time is defined as one second. 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 one second (dot)
- Disable the LEDs for one second (space between parts of the same letter)
- Illuminate the LEDs for three seconds (dash)
- Disable the LEDs for three seconds (space between letters)
- 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)

Modes of Operation

There is a reserved constant in program memory (1 byte) named UserMode. If UserMode has a value of 0x01, then one unit of time will be **1 second**. If UserMode has a value of 0x00, then you should treat one unit of time as **200 milliseconds**. The reason for this requirement is that it allows us to test your code in a more rapid fashion.

Your code **must** support operation using both of the modes as described above.

I don't expect your unit time to be exactly 200ms or exactly 1 second, but it needs to be as close as you can get it (within 100 microseconds).

Exam Material

Required template file - **available here**
Note: While testing your code, I will provide the LCD driver file that was used in Lab 4. Do not include it with your submission.

Extra Resources

While working on the exam, students are welcome to use code from the ECE375 slides, previous homework, or previous labs. It's perfectly fine to recycle code that you have written previously. I encourage you to be a strategic programmer and write your code in a modular fashion!
Design twice, code once.

Written Report

Document your work in a well-written report. Be sure that your report includes all of the content that is listed in this section. Present your work to the reader in a professional manner. I suggest that you use images, charts, diagrams or other visual techniques to help convey your information to the reader. Make your report something that is intriguing and interesting.

- Explain how you implemented your Morse code transmitter. You should provide enough information that a knowledgeable programmer would be able to draw a reasonably accurate block diagram of your program.
- Did you use interrupts in your code? Did you implement your code in a polling fashion? Explain the rationale behind your decision.
- What Timer/Counter mode did you use for Timer/Counter1? What prescaler and timer/counter values did you choose in order to create the 1 second unit time? What prescaler and timer/counter values did you choose in order to create the 200 millisecond unit time?
- What were the primary challenges that you encountered while working on the project?
- Is there anything you would design differently if you were to re-implement this project?
- Proofread your report! Better yet, ask another person to proofread your work. You will be marked down for spelling or grammatical mistakes.

There is no arbitrary minimum length requirement on the report. For example, if you are able to document your work and answer all of the questions in a two page document, that is perfectly fine. If you need five pages, that's perfectly fine too.

I will review your assembly code separately, so there is no need to include the source code within your final project report (unless you are describing certain snippets of the code).

Where to get help?

If you are working on the design of your final project, feel free to stop by office hours and ask questions. Since this is the final project, I expect students to demonstrate initiative while testing and debugging their code. We are happy to provide feedback and answer questions, but if you encounter a bug in your code, you will need to use your debugging skills to resolve it.

If you have a question regarding the project requirements, please let me know (either during lecture or send me an email). If the instructions are unclear, I can update the documentation and add an entry in the errata. **justin.goins@oregonstate.edu (mailto:justin.goins@oregonstate.edu)**

Grading Rubric

Implementation and Functionality

(17 pts) USER INTERFACE

- PD0, PD4, PD5, PD6, and PD7 are each configured in input mode
- Code does not cause a debouncing delay of more than 10 milliseconds (debouncing is completely allowed, you just can't consume more than 10 milliseconds).
- The program functions correctly even if non-relevant buttons are pressed
  - An example: when the code first boots, pressing PD4, PD5, PD6, and PD7 should have no effect (since the program is waiting specifically for PD0).
  - Another example: if PD5 is held down while pressing other buttons, it's value should be ignored.
- PB5-PB7 are correctly illuminated when a dot or dash is transmitted. They need to be disabled when a dot or dash is not being transmitted.
- The LED attached to PB4 (see errata, it's okay if you used PB0) correctly serves as a "transmission active" indicator. It is illuminated when the Morse code transmission begins and it remains illuminated until all characters have been completely transmitted.
- Button presses on PD0, PD4, PD5, PD6, and PD7 are completely ignored during the Morse Code transmission.

(30 pts) PROGRAM FUNCTIONALITY

- After booting, program shows correct greeting on LCD screen and waits to proceed until PD0 is pressed

```
Welcome!
Please press PD0
```

- Code displays the letter selection prompt as expected:

```
Enter word:
A
```

- User can press PD7 to iterate through the alphabet in reverse order
- Pressing PD7 correctly wraps around from A -> Z
- User can press PD6 to iterate through the alphabet in forward order
- Pressing PD6 correctly wraps around from Z -> A
- If there are fewer than 15 characters displayed, pressing PD0 locks the current character and allows the user to select the next character.
- If 16 characters have already been selected, pressing PD0 immediately begins Morse code transmission of the displayed message.
- The code functions correctly even if the user presses an arbitrary sequence of valid button combinations.
- Pressing PD4 immediately begins transmission of the message that is currently displayed (even it consists of fewer than 16 characters).
- After each full message is transmitted, the code returns to the letter selection prompt. The user can then select and transmit another word.

(9 pts) LCD CONSIDERATIONS

- The LCD screen never displays unexpected symbols, gibberish, or other undesired output
- The content of the LCD screen is immediately updated when the user performs a relevant action
- The LCD screen correctly displays the selected message as the user continues to add additional characters

(18 pts) TIMING

- Timer/Counter1 is used to manage all aspects of the Morse code transmission
- A unit delay is correctly generated as 1 second when UserMode has been programmed with a value of 0x01 (if the delay is skewed by more than 100 microseconds you will lose points).
- A unit delay is correctly generated as 200 milliseconds when UserMode has been programmed with a value of 0x00 (if the delay is skewed by more than 100 microseconds you will lose points).
- A dot occupies one unit of time
- A dash occupies three units of time
- The space between parts of the same letter is one unit of time
- The space between letters occupies three units of time

(26 pts) MORSE CODE ACCURACY

- Each letter of the alphabet (A-Z) is correctly transmitted in Morse Code using the ITU (International Telecommunication Union) standard.

Written Report

- (12 pts) Report provides reasonable explanation of the student's design. There should be enough information that a knowledgeable programmer would be able to draw a reasonably accurate block diagram of the program.
- (12 pts) Student describes the usage of interrupts and/or polling in their code. Where were interrupts used? Where was polling utilized? What is the justification for the selected implementation?
- (3 pts) Which Timer/Counter mode did you decide to use for Timer/Counter1?
- (6 pts) What prescaler and timer/counter values did you choose in order to create the 1 second unit time? Justify your choices by showing the mathematical calculations that were used to derive your selected values. What is your exact computed delay?
- (6 pts) What prescaler and timer/counter values did you choose in order to create the 200 millisecond unit time? Justify your choices by showing the mathematical calculations that were used to derive your selected values. What is your exact computed delay?
- (7 pts) What were the primary challenges that you encountered while working on the project?
- (5 pts) Is there anything you would design differently if you were to re-implement this project? How could your design be improved? Please provide an explanation and don't just give me a "yes" or "no" answer.
- (5 pts) Report is written in a professional manner and does not include spelling mistakes or grammatical errors.

Errata

This section will be updated when changes or clarifications are made. Each entry here should have a date and brief description of the change so that you can look over the errata and easily see if any updates have been made since your last review.

November 24th - Posted the original documentation and fixed a couple typos.
December 3rd - Important correction: The original instructions told you to use PB0 to indicate whether a transmission was active. Instead, you should use **PB4** for that purpose. PB5-PB7 will be used as the three LEDs which indicate dots and dashes. The instructions have been updated and we will discuss this in lecture. If you've already implemented the original version, that's perfectly fine, but implementing the new instructions may involve less work.
December 3rd - Fixed Canvas due date so that the assignment is due at 11:59pm on Thursday (earlier it said AM rather than PM).
December 4th - Posted grading rubric for final project.