## 1 Background

For the purposes of this homework, you will be implementing a simple Morse Code Translator in your MSP-EXP430G2 Launchpad[1]. Morse Code is a method used to transfer text, usually as a series of tones. The code was most commonly used in combination with telegraph machines. The International Morse Code provides an encoding of the basic Latin alphabet as a series of signals called *dots* and *dashes*, where a *dot* is a short pulse and a *dash* is a long pulse. The International Morse Code is shown in Table 1.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | •– | K | –•– | U | ••– | 1 | •––––– |
| B | –••• | L | •–•• | V | •••– | 2 | ••––– |
| C | –•–• | M | –– | W | •–– | 3 | •••–– |
| D | –•• | N | –• | X | –••– | 4 | ••••– |
| E | • | O | ––– | Y | –•–– | 5 | ••••• |
| F | ••–• | P | •––• | Z | ––•• | 6 | –•••• |
| G | ––• | Q | ––•– | | | 7 | ––••• |
| H | •••• | R | •–• | | | 8 | –––•• |
| I | •• | S | ••• | | | 9 | ––––• |
| J | •––– | T | – | | | 0 | ––––– |

Table 1: International Morse Code

To send a message in Morse Code, we use the encoding provided in Table 1 and the following set of timing rules:

- The duration of a *dot* is a single unit.
- The duration of a *dash* is three units.
- The duration of the space between parts of each letter is a single unit.
- The duration of the space between letters is three units.
- The duration of the space between words is seven units.

For example, to send the standard International Morse Code Distress Signal, *SOS*, we translate each letter using Table 1 and then employ the timing rules to obtain the message in Figure 1.


(a) Rendered in Morse Code (b) Rendered in Morse Code with Timing Added

Figure 1: The Standard International Morse Code Distress Signal

[1]If you do not own one of these boards, you may obtain one from TI's online store at https://store.ti.com/ or the UCF Bookstore.

## 2 Requirements

You will be implementing a Morse Code Translator in your MSP-EXP430G2 Launchpad. Because the board does not have a built-in way of generating a tone, we will be using `LED1` to show Morse Code instead. You are free to use C, MSP430 assembly, or a combination of the two to complete this homework. You are free to use functions in the C library provided by TI if you deem them necessary. You are not allowed to use any other external library[2].
The message string is guaranteed to have the following characteristics:

- It will be of at most 63 characters in length.
- A carriage return `\r` character will demarcate the end of a UART input.
- The end of input character is counted as part of the length of the message string.
- You must echo the string in the serial terminal as it is received by your program.
- Your program is free to ignore punctuation characters in the input string. Your program is also free to ignore any digits. Any punctuation or digits in the input string may count towards its length.
- Upon receiving the end of input character, the program must start "transmitting" the Morse Code encoded string through `LED1`. You must use the Timer_A module to blink the LED.
- The program must not accept any input while it is transmitting.
- The UART module must operate at the following configuration: 9600 8N1 (9600 baud, eight data bits, no parity, one stop bit)[3].

You may work in groups of up to four people towards the completion of this homework.

### 2.1 Bonus Points

If at any time when blinking the Morse Code the button in `P1.3` is pressed, the program must send through the UART the string encoded in Morse Code. Transmission through `LED1` must not be interrupted while this is taking place.

## 3 Development Guide

The following is a small series of hints that could help you complete this homework:

- You may use multiple source files instead of one monolithic source file if you feel it will simplify your work.
- Remember that the default frequency of the MSP430 in your Launchpad is of 1MHz. You do not need to change this behavior.
- Configure the USI as an UART. Use Table 15-4 in document number `slau144` to configure the baud.

[2]For users of the `msp430-elf` toolchain, you are free to use `newlib` and `libgloss` as these become your C library and runtime environment.

[3]Originally, this was set to 19200 8N1, but we found out that the Launchpad's UART is unstable at these speeds.

  - Full operation of the Universal Serial Interface (USI) as UART is described in Chapter 15 of document number `slau144`.
  - Configure the USI to trigger an interrupt whenever a character is received.
  - Store any incoming characters in a buffer.
  - If the end of input character is received, disable the interrupt.
  - Test your code at this point by displaying the received string in the serial terminal upon receiving the end of input character.
  - Reenable the interrupt when you are finished showing the string.

- Configure the Timer_A module in *Up Mode*. Have it trigger an interrupt every 250ms. This will be the length of a unit.
  - Full configuration of the Timer_A module is found in Chapter 12 of document number `slau144`.
  - Use Table 12-1 and equation 1 to configure the Timer_A module to the required frequency.

$$f_{timer} = \frac{f_{input}}{N \times (TOP - 1)} \tag{1}$$

    Where $TOP$ is the value in `TACCR0` and $N$ the prescaler used for the input clock.
  - Have the Timer_A module trigger an interrupt at this point.
  - Test your code by using the interrupt handler to blink a LED in your board.

- Write code that maps the Latin alphabet to Morse Code. You can test this code in your own computer without the need of a Launchpad.

- Integrate the UART code.
  - Upon receiving the end of input character, disable the RX interrupt and enable the Timer_A module.
  - Navigate the input string character by character.
  - Each time the Timer_A module triggers an interrupt, continue on the translation of the current character by turning on or off the LED.

## 4 Submission

To submit your project, pack your source files and compress them. Submit the resulting file to WebCourses. Ensure that the name of all your team members are in every source file. You only need to make a submission per team. Non-trivial parts of your code should be commented. For example, you do not need to describe in minute detail how you set up a Timer or I/O port, but you should state how you perform the translation between the string and Morse Code.