Morse Code Emitter

Objectives

In this assignment, you will build a low-powered embedded system for emitting Morse code. More specifically, upon the completion of this assignment, you should able to

- develop software for a μC that runs without an operating system,
- control an LED to meet the strict timing requirements for emitting Morse code,
- configure a USART in UART mode and communicate using it,
- configure a timer,
- write interrupt service routines (ISRs), and
- develop low-power applications.

IMPORTANT This is an individual assignment; you must write all of the code on your own. You may, however, discuss the problem at a very abstract level with your peers. Your source code must be prepared carefully and with attention to both correctness and style. When working on this assignment, please refer to the included marking scheme.

Assigned:	Sep. 30th	Total points:	20
Due:	**Thursday**, Oct. 25 at 11:59 PM	Weight:	20%

Submission

IMPORTANT You must submit your assignment **electronically**. Your **electronic submission** must be submitted to the BlackBoard by 11:59 PM on Thursday, Oct. 25.

Overview of Tasks

1. **Develop a plan** for your program before you start coding it; solving this assignment is non-trivial.
2. **Create skeleton C files** for your assignment that include complete header comments. These comments must include at least
   - your name,
   - the course,
   - the date, and
   - a brief description of the program and/or file (as appropriate).
3. **Write the C source code and an associated `Makefile`** to produce the binary file Image.
   - Develop and test your solution component-by-component.
4. **Create a file name `DEFECTS`.** In this file, clearly document all known defects or clearly state that none exist. This file will be marked for its accuracy.

Details (Image)

IMPORTANT The following International Telecommunications Union (ITU) recommendation

- https://www.itu.int/rec/R-REC-M.1677-1-200910-I/en

applies to Morse code. Your implementation must adhere to only those parts mentioned below.

Your Morse code emitter must receive characters over the UART. You must configure it as follows:

- 2400 baud rate,
- 8 bits per character,
- no parity bit, and
- 1 stop bit.

Bytes received must be buffered.

Your application will start in the *idle* state. When it receives a byte (over the UART) that it can emit, it will enter the *active* state. As the device enters the *active* state, it will emit the "Starting signal" sign. After emitting this sign, it will begin to emit buffered bytes as Morse code. New bytes can be buffered in the *active* state. When it attempts to fetch a byte from an empty buffer, it will return to the *idle* state. As it enters the *idle* state, it will emit the "End of work" sign.

Note that these *idle* and *active* states have no relationship to the μC's CPU state. The CPU must predominately remain in the LPM3 state.

Your program must be able to emit the following characters when they are received over the UART (M.1677, Part 1):

- All letters from 1.1.1 except for an accented e.
- All figures from 1.1.2.
- Only the following character from 1.1.3: Full stop.

Your program will follow all of the duration requirements outlined in M.1677, Part 1, 2.1-2.4. The dot length must be 120 ms.

Your program will emit Morse code as follows:

- dot: the red LED must be enabled for the dot duration and a period (.) must be sent over the UART,
- dash: the red LED must be enabled for the dash duration and a dash (-) must be sent over the UART,
- space between two letters: the LEDs must remain off for the appropriate duration and a space (␣) must be sent over the UART, and
- space between two words: the LEDs must remain off for the appropriate duration and two spaces (␣␣) must be sent over the UART. For simplicity, you may assume that a space will be always received over the UART that marks the end of a word.

All characters must be sent over the UART at (essentially) the same time as the LED changes state.

Checklist

Your submission must…

- ☐ be adequately commented, including a header block for every file
- ☐ avoid busy waiting
- ☐ produce no output when first powered on configure
- ☐ the UART as described
- ☐ appropriately handle all of the described characters
- ☐ ignore all other received characters (Except the space that may be received to mark the end of a word)
- ☐ produce output using the red LED
- ☐ produce output using the UART produce
- ☐ correctly-timed output
- ☐ correctly implement the dot duration (120 ms) and all durations derived from it
- ☐ implement an input buffer of at least 24 characters
- ☐ enter a low-power mode (LPM3) whenever possible
- ☐ clearly document all known defects in a file named DEFECTS

Your source code should be written in a professional manner, and it should be thoroughly tested.

Assumptions

When you encounter ambiguities in the specification, do not hesitate to clarify them with your instructor. Within your code, state all of your assumptions.

Marking scheme

IMPORTANT Submissions that **fail to compile** will not receive more than 50% of the available marks.

Item	Points
Makefile	1
DEFECTS (accuracy)	2
Implementation of the described features (including style)	17
	20