Final Project: Morse Code Interpreter

Introduction

This project uses FRDM-KL25Z as a MORSE code interpreter device. The development of this project involves PWM, TIMER, circular buffers, GPIO, and UART using interrupts.

MORSE code interpreter works here in two ways. When the user taps the button, it will be interpreted as the MORSE code, or when the user enters the string from the terminal it will be translated into the MORSE code and interpreted using LED.

Objective

The goal of this project is to have an interactive command interface, where users will be able to enter the string to be interpreted using MORSE code and the users will also have a provision to enter MORSE code using the tap button. Then the processor converts the input into the dot (‘d’) and dash (‘D’) format and interprets the same using the RGB LED.

Project background

Morse code: This is a method used in telecommunication to encode text characters as standardized sequences of two different signal durations, called dots and dashes[1].

Elements to be developed

1. **Circular buffer implementation.** Here we will be using three circular buffers, among them, two buffers will be used by the UART0 for transmitting and receiving directions. Another one will be used to store the interpreted morse code in the dot (‘d’) or dash (‘D’) format to display using LED.

2. **Configuring UART0.** Configuring the UART0 based on the below-mentioned parameters. Implementation will be done on top of the circular buffers, and it will be completely interrupt-based.

Baud rate	115200
Data size	8
Parity	None
Stop bit	1

3. **Glue code that ties UART communication code to standard C library function**
   After configuring the UART0, the call to standard C library functions like printf() / getchar/ putchar() will be routed to UART0 to the PC or vice-versa.

---

[1] Morse code background

4. **Command processor.** This will be able to accept a string from the user, which will be interpreted as MORSE code and displayed using the LED.

5. **Configuration of GPIO.** This will be configured to accept MORSE code input from the user. Interrupt on either rising or falling edge of the input is set to measure the tap duration to classify the input as a dot or a dash.

6. **Timer configuration.** To measure the duration between the interrupts, a timer system will be set to give ticks every 100ms. Using this we will be able to determine the input is a dot or a dash.

7. **TPM configuration.** This configuration will be set to display a different warning or acknowledgment signals using RGB LED when the user taps the button to enter the MORSE code.

Implementation details

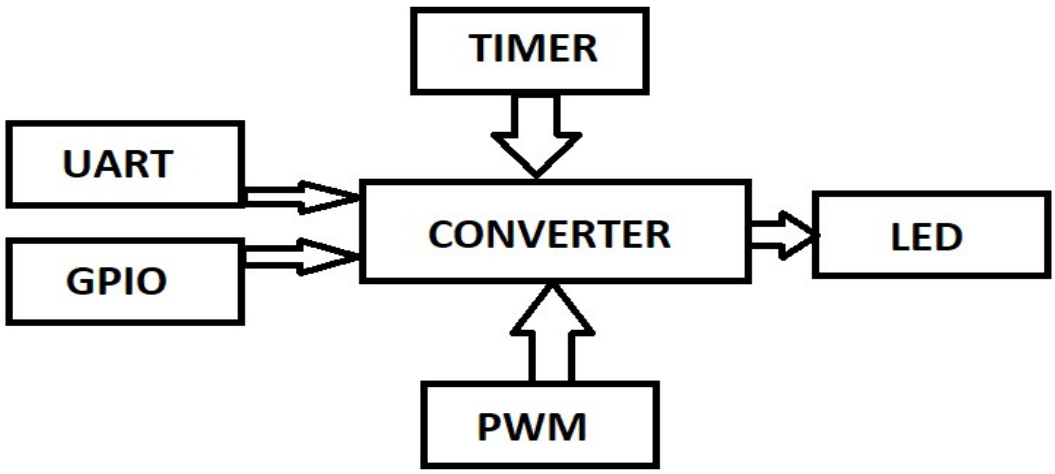

Figure 1: Basic block diagram of peripherals involved in morse interpreter

The basic work of the project is outlined in the above block diagram. Here UART, GPIO, TIMER, PWM, and LED refer to the FRDM-KL25Z peripherals.

CONVERTER refers to the ARM processor running the firmware to convert user input into the dash and dot format, which will be used to interpret MORSE code using LEDs.

The controller reads the user's input from either terminal (UART) or Tap-button (GPIO). It also uses a timer system developed (Systick) to convert the input to the dot and dash format. Then it displays the interpreted values using PWM (different colors) and RGB LEDs.

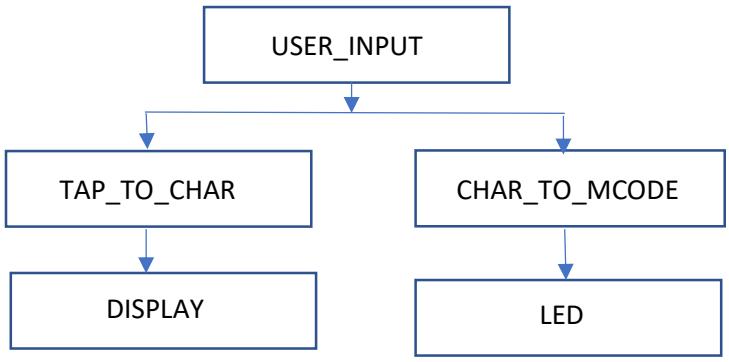

Figure 2: Flow diagram of the MORSE code interpreter

The working of the model will look like the flow diagram mentioned above.

- USER_INPUT: When the user enters a character string through the terminal, that will be interpreted as MORSE code, or when the user taps the button to give MORSE code input, that will be converted to the respective character and displayed on the terminal.

- When the user taps the button to give MORSE code input, it will be captured using GPIO interrupts and the TIMER. Our GPIO interrupt configuration will generate interrupts on both falling and rising edges, that allow us to determine the duration of tap-button hold (/press) using a Systick timer. If the time duration between the successive interrupt trigger (interrupt from falling and the rising edge of the GPIO, considering GPIO line is pulled high) is below 300ms, that is recognized as a dot, or more than 300ms and less than 1 second is recognized as a dash, else an INVALID INPUT.

  If there is no input after 500ms since detecting a dash or a dot, it will be considered as the end of the code for a character.

  These codes will be simultaneously stored in a circular buffer, and once the end of code for a character is detected. The code is dequeued and interpreted as a character and displayed on the terminal by the TAP_TO_CHAR function.

  Simultaneously LED will give real-time feedback to the user as mentioned below.

  While the button is being pressed LED will be GREEN, if the dot is red LED will be RED, if the dash is red LED will be BLUE and YELLOW for the timeout.

- When the user enters a character string from the terminal, the function CHAR_TO_MCODE converts the characters to their corresponding MORSE code, and it will be stored in the buffer. Then the stored MORSE code will be interpreted using LED (RED for dot, GREEN for dash, and BLUE for the end of character).

Testing strategies:

Unit testing will be one of the techniques involved in this process to test the functions within the firmware. Along with that automated testing script will be included, which will run initially during the boot of the program, where it tests with various string inputs and validates for the correct MORSE code output.

Visual validation is done by observing, a test suite running at the start of the program which displays the MORSE code using LED for the letters starting from A to E.

Response to the project approval questionnaires:

- Functionality is included in the implementation section.
- Technologies used GPIO, UART, PWM, Circular buffers, Command Processing, Systick, and Unit testing.
  In assignments we used GPIO interrupts in falling edge configuration, but this project is concentrating on generating interrupts on either edge of the input and calculating the time duration between them, to determine the correct input from the user. Along with other technologies, unit testing is one of the new strategies involved to leverage the testing.
- As mentioned above unit testing will be the one, I want to concentrate on and the references are uCunit or cmocka. For the rest FRDM-KL25Z manual and datasheets will be useful.
- This project doesn't require any hardware except the push button for GPIO.
- Testing strategies are mentioned in the testing strategies section.