Name: Tashyab Raj

Roll: 2101214

Date: 14-04-2024

Endsemester Project Report

**Aim:**

To design and implement a Morse code decoder using a PSoC microcontroller.

**Theory:**

Morse code is a method used in telecommunication to encode text characters as sequences of two different signal durations, called dots and dashes, or dots and dashes. In Morse code, each character is represented by a unique combination of dots and dashes, and the characters are separated by spaces. The Morse code alphabet includes letters, numbers, and punctuation marks.

The Morse code decoder operates as follows:

* Serial Communication: The PSoC microcontroller communicates with an external device (e.g., a computer running PuTTY) via a serial communication interface, such as UART (Universal Asynchronous Receiver-Transmitter). PuTTY is configured to send Morse code signals representing characters by transmitting dots (short pulses) and dashes (long pulses) separated by spaces.
* Input Signal Detection: The PSoC microcontroller receives the Morse code signals through the UART interface. It continuously monitors the input signal and detects the timing and duration of each pulse (dot or dash) received from PuTTY.
* Morse Code Decoding: Based on the timing and duration of the pulses received, the PSoC microcontroller decodes the Morse code signals into corresponding characters. It compares the received pulse patterns with a predefined Morse code lookup table to determine the corresponding characters.
* Character Display: Once a complete character is decoded, the PSoC microcontroller displays the decoded characters on an output interface, such as an LCD display or through a serial interface for further processing or display.
* Error Handling: The Morse code decoder implementation includes error handling mechanisms to account for variations in pulse timing and noise in the input signal.

**Pseudo Code:**

Initialize global variables:

morseBuffer of size 100

bufferIndex = 0

charIndex = 0

Define function decodeMorse(morseCode):

Loop through a list of Morse code mappings

If morseCode matches a Morse code sequence in the list, return the corresponding character

If no match is found, return a space character

Define function handleMorseInput(input):

If input is a dot:

Turn on LED

Wait for a short duration (DOT\_DELAY\_MS)

Turn off LED

Display dot on LCD

Send dot over USBUART

Append dot to morseBuffer

If input is a dash:

Turn on LED

Wait for a longer duration (DASH\_DELAY\_MS)

Turn off LED

Display dash on LCD

Send dash over USBUART

Append dash to morseBuffer

If input is a space:

Send space over USBUART

Terminate morseBuffer string with null character

Decode morseBuffer into a character using decodeMorse function

Display decoded character on LCD

Reset bufferIndex to 0

Clear Morse code display on LCD

If input is 's' or 'S':

Send "Hello from PSoC!" over USBUART

Main loop:

Enable global interrupts

Initialize LCD and USBUART

Infinite loop:

Check if USB configuration has changed

If configuration changed, reinitialize USBUART if configuration is enabled

If USB configuration is enabled:

Check if data is available on USBUART

If data is available:

Read character from USBUART

Pass character to handleMorseInput function

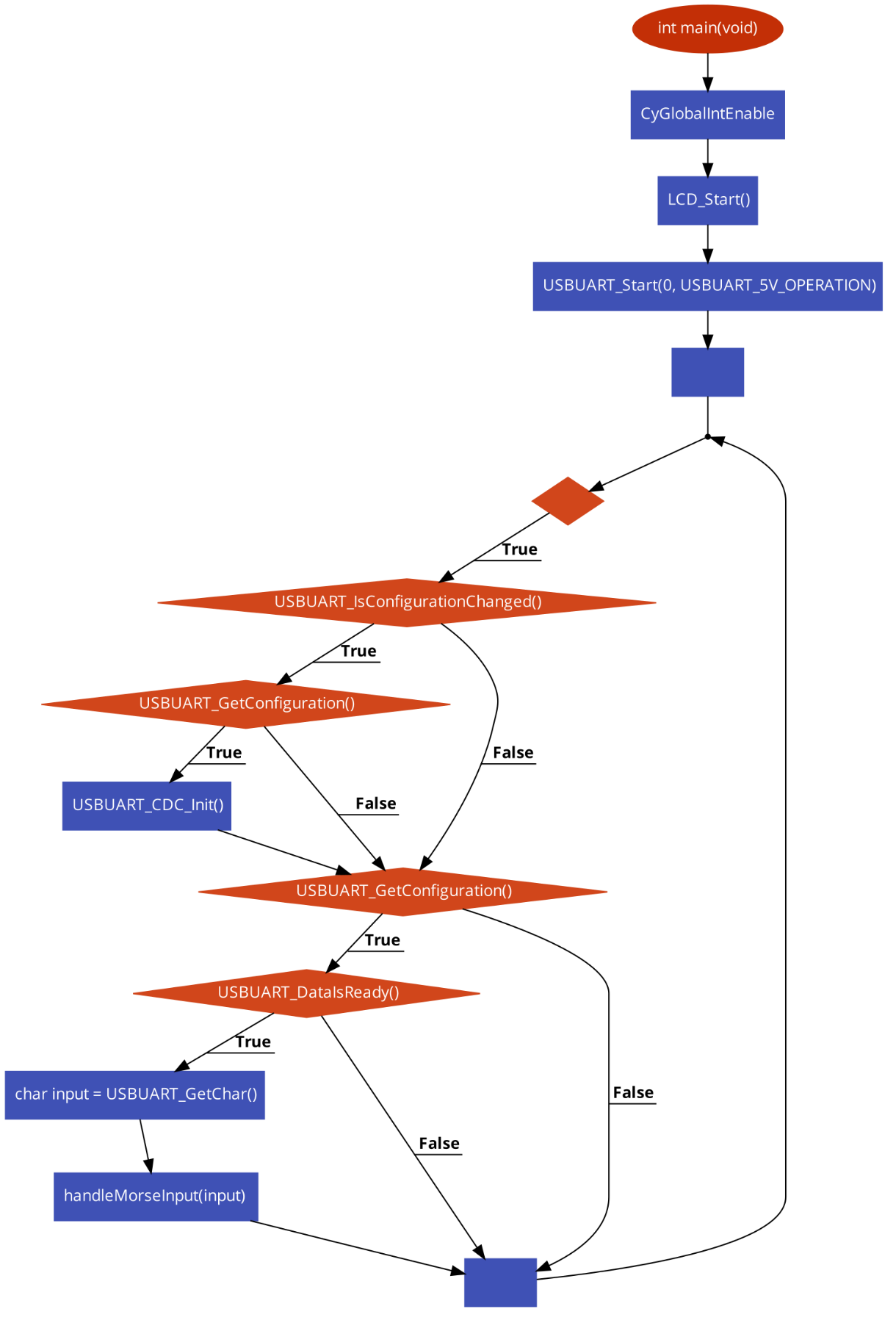
End of main

**Conclusion:**

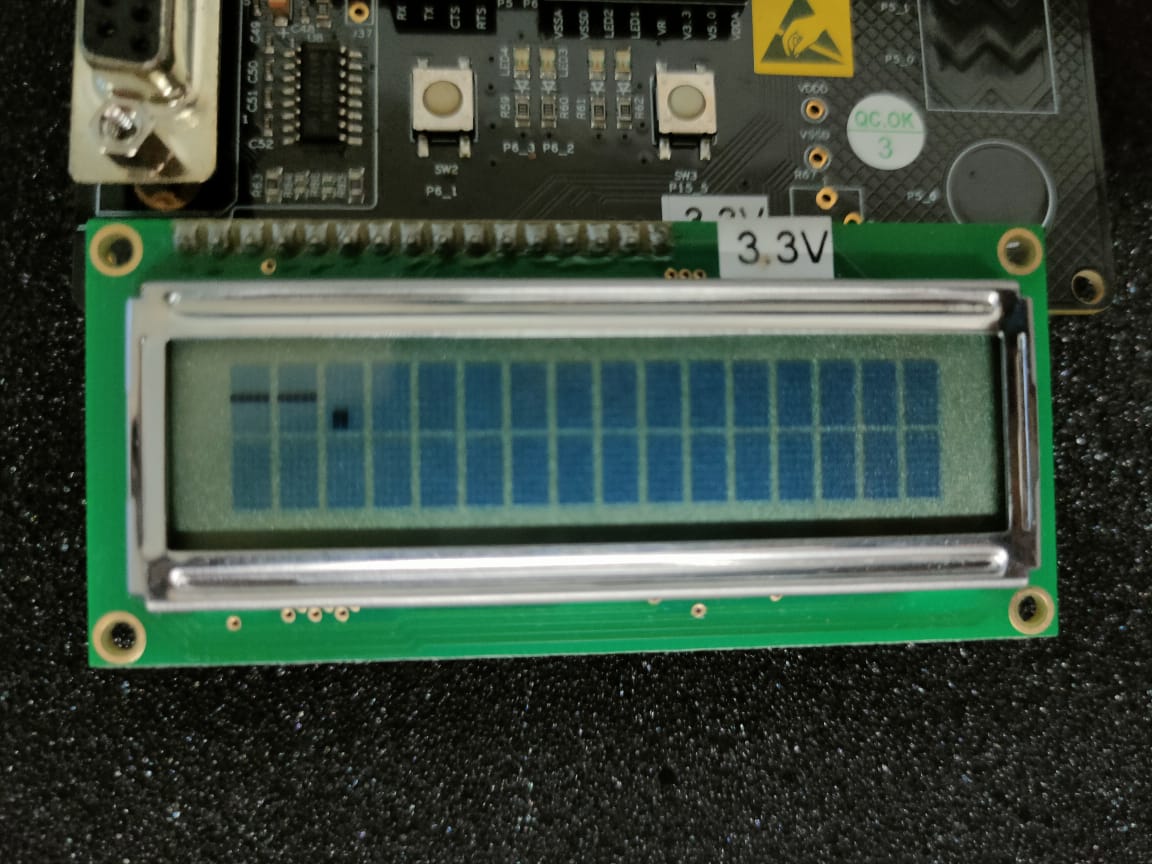
The Morse code decoder project using the PSoC microcontroller has been successfully implemented, providing a practical demonstration of signal processing, serial communication, and character decoding techniques. Through the integration of hardware components such as LEDs for visual feedback and an LCD display for character output, coupled with software functionalities including UART communication and Morse code decoding algorithms, the project showcases the versatility and capabilities of the PSoC platform.

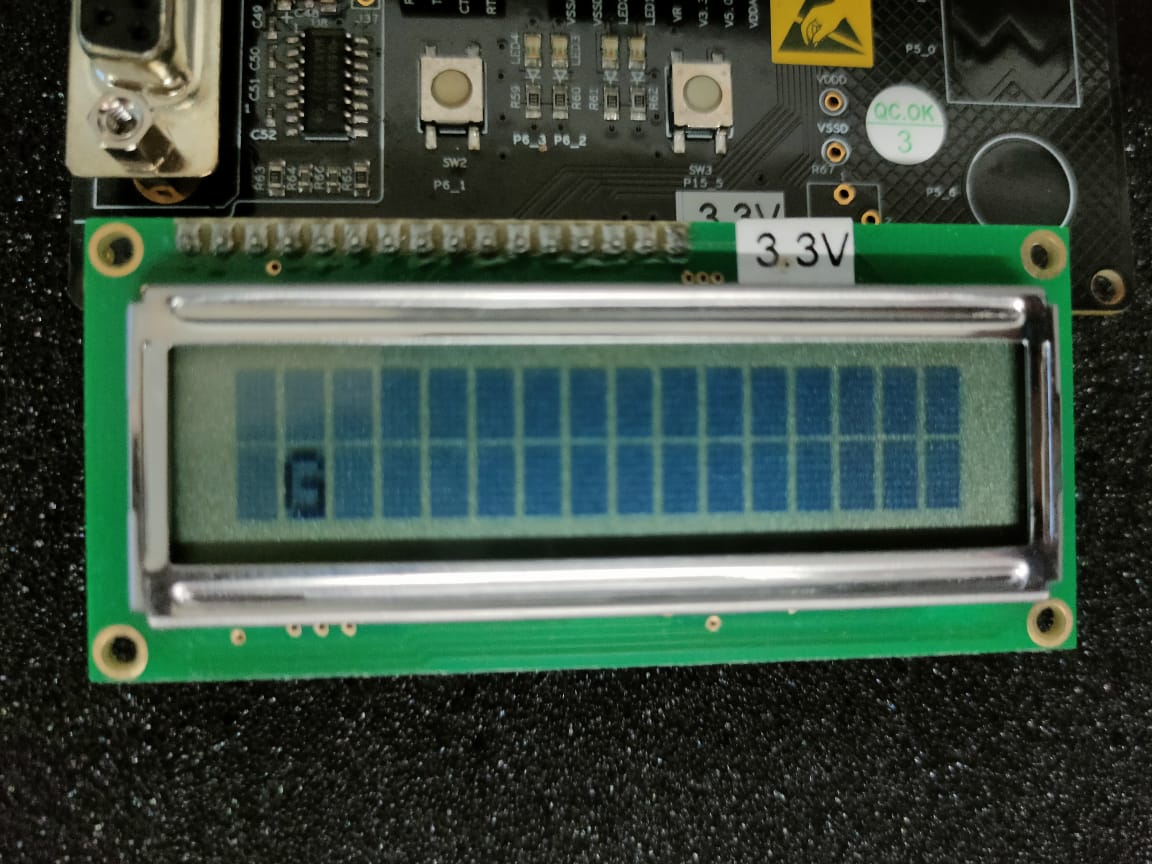
Throughout the development process, various challenges were encountered and overcome. These challenges included designing efficient algorithms for Morse code decoding, implementing error handling mechanisms to account for variations in input signals, and integrating multiple hardware components to provide a seamless user experience.

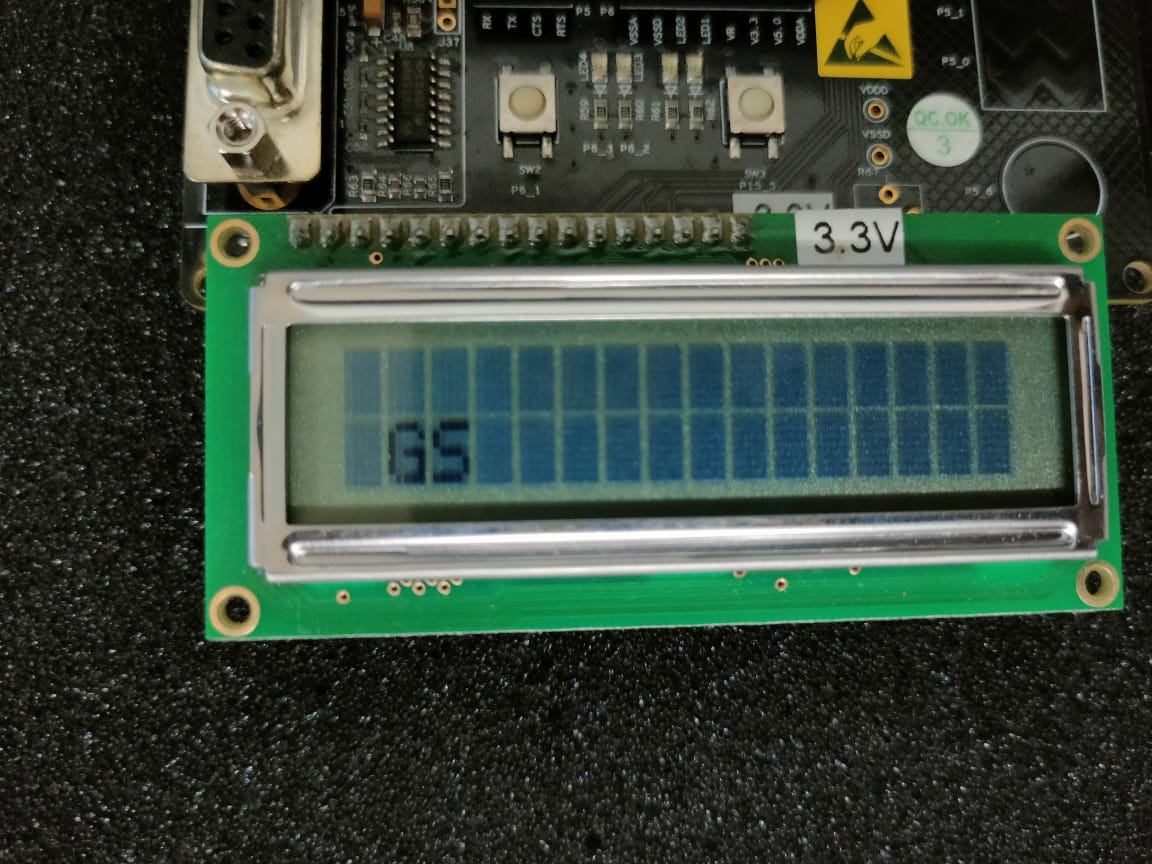
**Flowchart:**

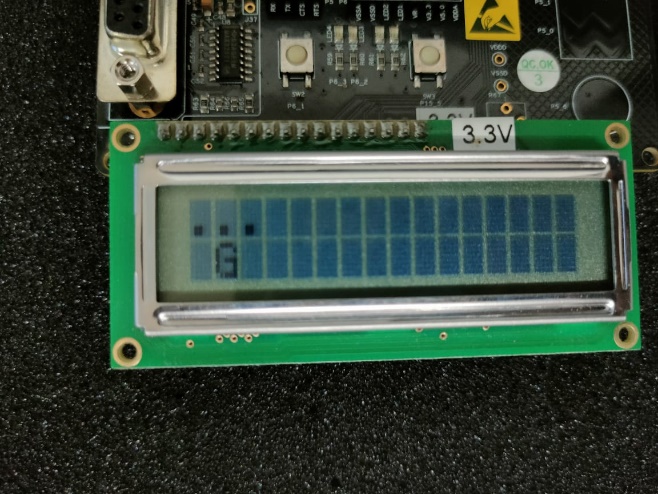


**Output:**

****

****

****

****