Project: **Simon Says Game with LEDs and Buttons on TM4C123GH6PM**

**Project Overview:**

You will create a memory-based game where the TM4C123GH6PM MCU generates a random sequence of LED flashes (Red, Blue, Green). The player must then repeat the sequence using buttons (SW1 and SW2). The sequence length increases after each correct attempt, and the game becomes faster over time.

**Features:**

* Flashing sequence of LEDs (Red, Blue, Green).
* Player input through buttons.
* Feedback for correct and incorrect sequences.
* Score display via UART.
* Gradually increasing difficulty (faster sequences).

**Components:**

1. **TM4C123GH6PM Tiva C Board**.
2. **Onboard LEDs**:
   * PF1: Red LED.
   * PF2: Blue LED.
   * PF3: Green LED.
3. **Onboard Buttons**:
   * SW1 (Pin PF4).
   * SW2 (Pin PF0).
4. **UART** for score display.

**Summary of Functions:**

1. **PortF\_Init**: Initialize GPIO for buttons and LEDs.
2. **UART\_Init**: Initialize UART for serial communication.
3. **Timer0\_Init**: Initialize Timer0 for millisecond delays.
4. **delayMs**: Function to introduce millisecond delays using Timer0.
5. **generateSequence**: Generate a random LED sequence.
6. **showSequence**: Display the generated sequence using LEDs.
7. **flashLED**: Flash the appropriate LED.
8. **playerTurn**: Get player input and check if it matches the sequence.
9. **getButtonPress**: Detect button press for player input.
10. **gameOver**: End the game and display the final score.
11. **UART\_SendString**: Send a string over UART.
12. **UART\_SendNumber**: Send a number over UART.

**Extensions:**

* **Difficulty Levels**: Adjust the speed and sequence length for different levels.
* **Lap Timing**: Track the player’s timing for each input and display via UART.
* **Multiplayer Mode**: Alternate turns between two players using buttons.

**Simon Says Game: Sequence and Function Documentation**

**1. Sequence Overview**

In the Simon Says game, the sequence represents a series of LED flashes that the player must memorize and repeat. Each sequence starts short (e.g., one LED flash) and grows longer as the game progresses. After each successful attempt, a new random LED is added to the sequence, and the player must correctly recall the entire sequence to continue. The sequence uses three LEDs: **Red (PF1), Blue (PF2), and Green (PF3)**.

* **First round**: One LED (randomly chosen) flashes.
* **Subsequent rounds**: The previous sequence repeats, followed by a new LED flash added to the end.
* **Player's task**: Memorize and input the sequence by pressing buttons corresponding to the LED flashes.

**Function Documentation:**

**2. PortF\_Init**

* **Purpose**: Initialize the necessary GPIO pins for LED control and button inputs.
* **LEDs**: Configures PF1, PF2, and PF3 as output for the Red, Blue, and Green LEDs, respectively.
* **Buttons**: Configures PF0 and PF4 as inputs for SW2 and SW1 buttons, respectively, and sets pull-up resistors for these inputs.
* **Expected Behavior**: After calling this function, the LEDs should be ready for flashing, and the buttons should be configured for player input.

**3. UART\_Init**

* **Purpose**: Set up the UART module for serial communication, allowing the MCU to transmit data (e.g., score or game messages) to a terminal.
* **Expected Behavior**: This function configures UART to transmit game data, such as player scores, to a connected serial terminal (e.g., via PuTTY or a similar tool).

**4. Timer0\_Init**

* **Purpose**: Initialize Timer0 to generate millisecond delays. These delays are essential for controlling the timing of LED flashes and for managing user input timings.
* **Expected Behavior**: After initialization, the timer should generate precise delays when required in the game, allowing for controlled flashing of LEDs and pauses between actions.

**5. delayMs**

* **Purpose**: Introduce delays in milliseconds, using Timer0 to control the timing of LED flashes and ensure proper timing between game events.
* **Usage**: This function will be called when creating pauses between LED flashes (e.g., to make the sequence observable by the player) and when introducing pauses after a player inputs a sequence.
* **Expected Behavior**: Calling this function will pause the game’s execution for the specified number of milliseconds.

**6. startGame**

* **Purpose**: This is the main game loop that manages the progression of the game. It handles generating the LED sequences, flashing them, accepting player inputs, and checking for mistakes.
* **Expected Behavior**: The game begins by generating a new sequence, showing it to the player, and then prompting the player to repeat it. The sequence gets longer after each successful attempt, and the game continues until the player makes a mistake.

**7. generateSequence**

* **Purpose**: Generate a random sequence of LEDs that the player needs to memorize and repeat. The sequence is stored in an array, with each element representing a particular LED (Red, Blue, or Green).
* **Sequence Representation**:
  + 0 = Red LED (PF1).
  + 1 = Blue LED (PF2).
  + 2 = Green LED (PF3).
* **Expected Behavior**: Each time this function is called, it generates a new sequence by appending one more random LED to the existing sequence. The sequence is randomly generated to provide variability in each game round.

**8. showSequence**

* **Purpose**: Display the generated LED sequence to the player. Each LED in the sequence flashes for a brief period, with a short delay between each flash.
* **Expected Behavior**: The function will cycle through the current sequence, flashing each LED in turn, and pausing briefly between flashes to allow the player to observe the full sequence.
* **Visual Feedback**: The player should see the LEDs light up in the correct order, allowing them to memorize the sequence.

**9. flashLED**

* **Purpose**: Flash a specific LED based on the input parameter (Red, Blue, or Green). This function is called by showSequence to light up the appropriate LED in the sequence.
* **Expected Behavior**: The specified LED will turn on for a set amount of time (e.g., 500 ms) and then turn off, simulating the flash of an LED as part of the game’s sequence.

**10. playerTurn**

* **Purpose**: Wait for the player to input the sequence of LED flashes using the buttons. This function checks if the player’s input matches the generated sequence.
* **Input Method**:
  + SW1 (PF4) represents the Red LED.
  + SW2 (PF0) represents the Blue LED.
  + Additional buttons or delays may be used for other colors.
* **Expected Behavior**:
  + The player presses the buttons in the order corresponding to the LED sequence.
  + The function verifies each button press against the sequence. If the player inputs the correct sequence, the game progresses. If not, the game ends.

**11. getButtonPress**

* **Purpose**: Detect and return the player's button press. This function is used during the player's turn to register which button (SW1 or SW2) was pressed and return the corresponding value (0, 1, or 2).
* **Expected Behavior**: This function continually monitors the buttons and returns when a button is pressed. It maps button presses to LED colors (Red for SW1, Blue for SW2) and communicates that back to the game logic.

**12. gameOver**

* **Purpose**: End the game and notify the player that they have made a mistake. This function provides visual feedback (e.g., flashing all LEDs) and displays the player’s final score via UART.
* **Expected Behavior**: When the player makes a mistake, this function will flash all LEDs several times to indicate game over. It will also send the player’s final score to the terminal via UART.

**13. UART\_SendString**

* **Purpose**: Transmit a string of characters (e.g., a message or score) over UART to a connected serial terminal.
* **Expected Behavior**: This function will send the provided string to the terminal, allowing for communication between the Tiva C board and the player via a serial interface.

**14. UART\_SendNumber**

* **Purpose**: Convert a number (e.g., the player's score) to a string and send it via UART.
* **Expected Behavior**: When the game is over, this function will convert the player's score into a string and transmit it over UART, so the player can see their performance on the terminal.

**Sequence Flow:**

1. **Generate Sequence**:
   * A random LED sequence (Red, Blue, Green) is generated and stored.
   * The first sequence starts with one LED. For each new round, the sequence grows longer.
2. **Show Sequence**:
   * The generated sequence is displayed one LED at a time with a short pause between flashes.
   * The sequence is shown to the player to memorize.
3. **Player Input**:
   * The player inputs the sequence by pressing buttons.
   * The game compares the player's input with the stored sequence.
4. **Check Input**:
   * If the player inputs the correct sequence, the game adds a new LED to the sequence and repeats.
   * If the player inputs the wrong sequence, the game ends, and the final score is displayed.

**Visual Representation:**

* **LEDs** flash in the order of the sequence (Red, Blue, Green).
* **Player Inputs**: The player repeats the sequence by pressing buttons:
  + SW1 (Red LED),
  + SW2 (Blue LED).
* If the player repeats correctly, the game continues with a longer sequence and faster flashing. If the player is incorrect, the game flashes all LEDs to indicate the end.

Top of Form

Bottom of Form