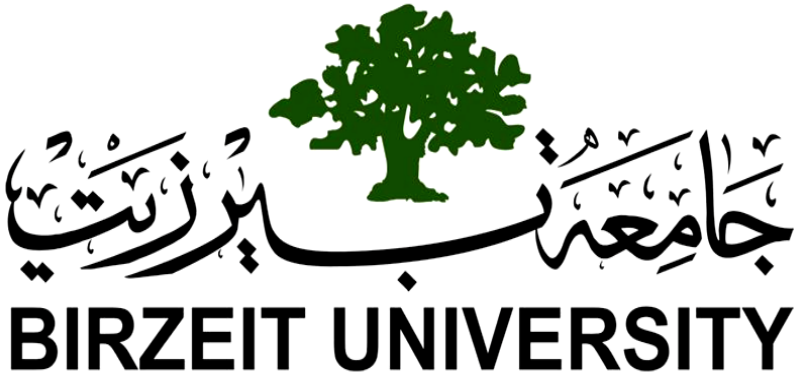

Faculty of Engineering & Technology
Electrical & Computer Engineering Department

Real-Time System and Interfacing Techniques Laboratory-ENEE4113

Report-ToDo#2

A Resettable Counter Up System in LabVIEW

Instructor:
Dr.Wasel Ghanem

Prepared By:
Abd Khuffash-1200970

Section:
3

Date:
7-10-2023

Abstract

This To-do focuses on building a resettable counter using LabVIEW. The counter can increment by 1 or 2 and has a reset function to set it back to zero. The counter's value is capped at 15, and its binary equivalent is displayed on four LEDs. The implementation involves using shift registers, buttons for counting and resetting, and logic to prevent the value from exceeding the limit. This To-do demonstrates basic digital counting principles and how to manage state using LabVIEW in real-time applications.

Table of Contents

Contents

Table of Figures

Theory

In digital systems, counters are fundamental components used to track events, perform timing operations, and generate sequences. A counter works by incrementing or decrementing a stored value based on a clock signal or user input, and it is widely used in electronics, automation, and control systems. In this experiment, a resettable counter is designed using LabVIEW, a graphical programming language commonly used for simulation and real-time control.

Counting Mechanism

The counter allows the user to increment the value either by 1 or by 2, depending on which button is pressed. These buttons control two separate cases in a Count Case Structure, where each case dictates the amount by which the counter is incremented. The counter is capped at 15, and any additional increments after reaching this value will stop.

Shift Registers

Shift registers are used in the loop to retain the value of the counter between iterations. Each time the loop runs, the value in the shift register is updated based on user input (whether the count button is pressed) and passed into the next iteration, ensuring that the counter value is maintained across different cycles of the program.

Boolean Logic and Case Structures

The system makes use of Boolean buttons and case structures to manage control flow:

- **Count Case Structure**: This structure determines whether the counter is incremented by 1 or by 2. When either button is pressed, the respective case is executed, incrementing the counter value.
- **Reset Case Structure**: This case structure resets the counter when the reset button is pressed. Inside this case, the counter value is forced back to **0**, regardless of the current value. This ensures that the system can be restarted at any time.
- **Leds Case Structure**: This case structure determines which LEDs will be illuminated based on the binary representation of the counter's value. Each LED corresponds to a specific bit in the binary output. For example, if the counter reads 3 (binary 0011), LEDs representing LED0 and LED1 will light up, while LED2 and LED3 will remain off.

Binary LED Display

The counter's current value is displayed using four LEDs, which represent the binary equivalent of the counter. When the value is incremented or reset, the LEDs automatically update to reflect the new binary state. A **Leds Case Structure** is used to handle the conditions for displaying the binary values on the LEDs. For example, if the counter reaches its maximum value or certain increments, the case structure decides which LEDs should be illuminated.

Message to the User

An additional feature in this implementation is the Display Message to User function. This provides feedback to the user, ensuring they are informed when certain events occur, such as reaching the maximum count value or resetting the counter. This enhances the usability of the system and provides a user-friendly interface.

While Loop Structure

In implementing the counter with LabVIEW, a while loop is essential to manage the continuous counting process. The loop allows for repeated execution of the counting logic until a stop button is pressed.

This design combines digital counting principles with LabVIEW's ability to handle real-time control and user input. The use of case structures allows for flexible control over different actions (counting or resetting), while shift registers ensure that the counter maintains continuity across iterations. The binary LED display provides an intuitive visualization of the counter's state.

Procedure

At first, a while loop with a button was used to control the circuit. The loop continues to run as long as the stop button is not pressed.

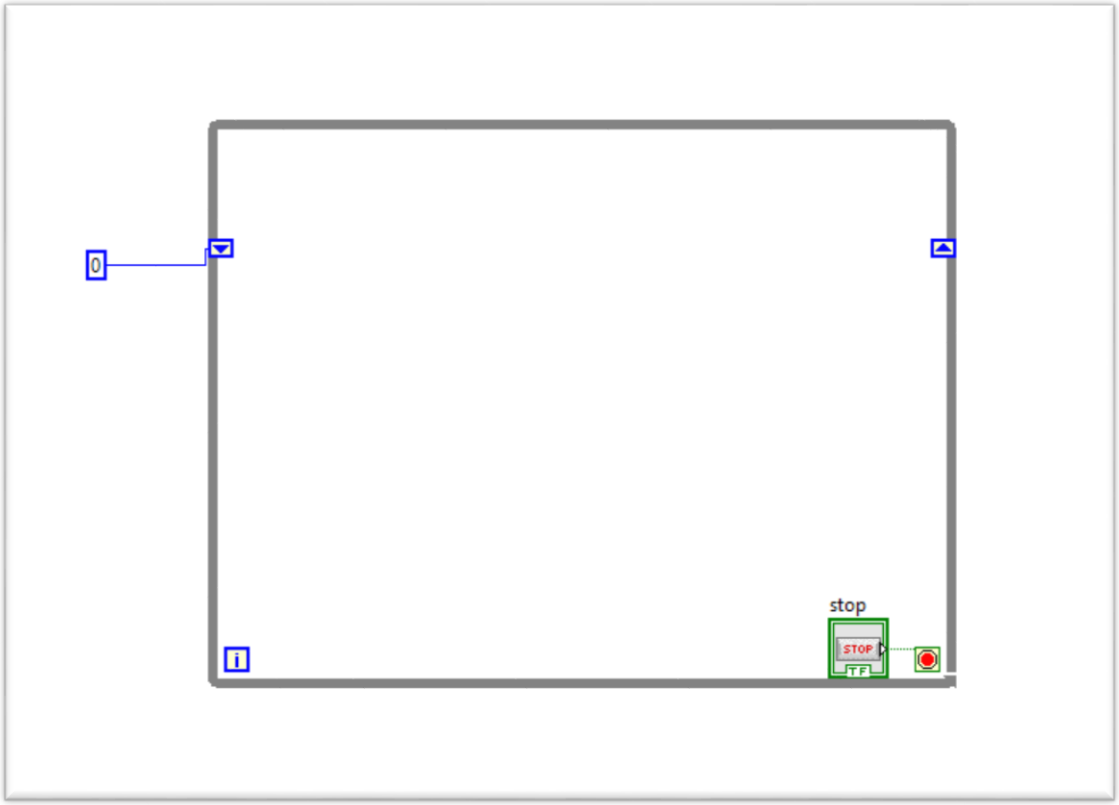

Figure 1. While Loop Structure

The arrows on the border of the while loop represent the input and output of a shift register. The arrow on the left is the input, which is connected to a constant value of 0. This sets the initial value of the shift register, meaning the counter starts at 0 at the beginning of the loop. The value stored in the shift register is then passed through the loop, updating based on the actions inside, such as counting or resetting.

After that, three push buttons were added to the system. The first button is used to count up by 1, the second button is for counting up by 2, and the third button is for resetting the counter. Additionally, four LEDs were used to display the counter's value in binary format, where each LED represents a binary digit of the counter. The picture below indicates for the formatting of these components:

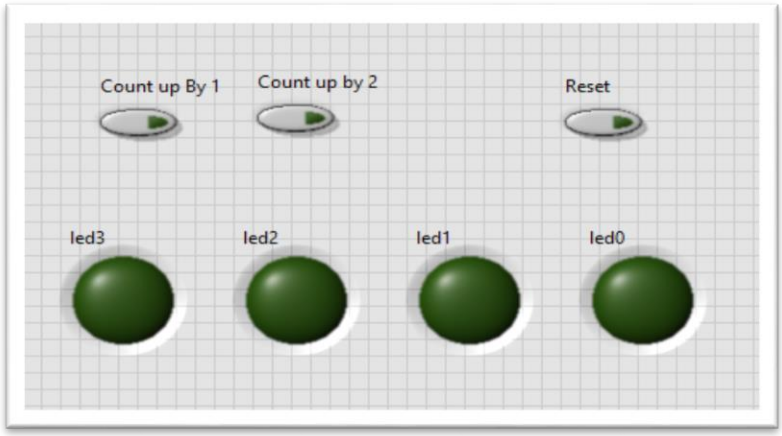

Figure 2. Push Button, and 4 LEDs

For each push button, a mechanical action mode called "Latch When Released" was used. This setting ensures that the button stays "pressed" until it is physically released, and only then does it trigger its corresponding action (such as counting up or resetting). This mode is useful because it prevents multiple accidental activations when the button is pressed and held for too long.

The "Latch When Released" action guarantees that each button performs its task only once, reducing any risk of errors in the counting or resetting process.

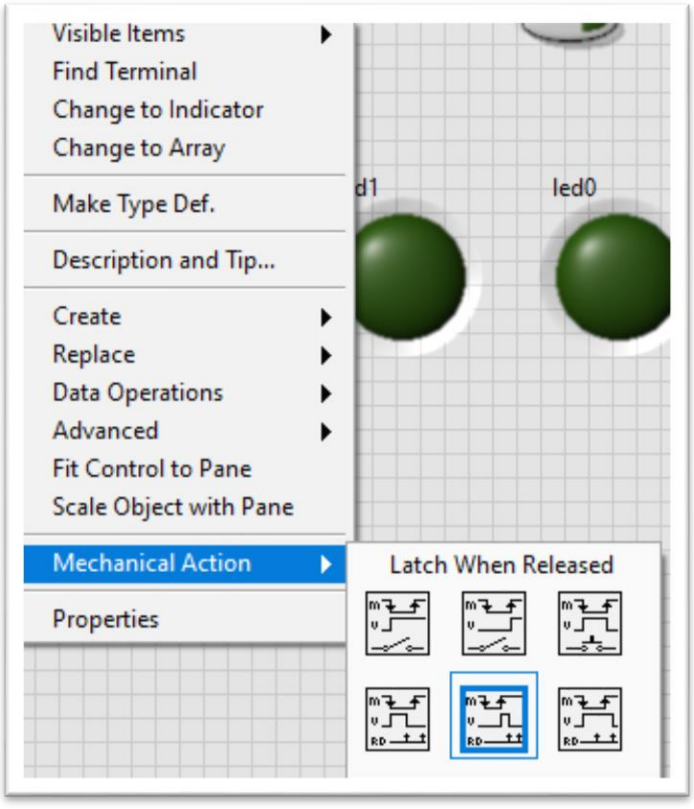

Figure 3. Mechanical Action for Push buttons

For the push buttons logic, the "Count Up by 1" button's output was converted from a Boolean value (True/False) to a numeric value. If the button is pressed, it outputs 1, and if it's not pressed, it outputs 0.

For the "Count Up by 2" button, a case structure was used to check if the button was pressed. If the value is True (button pressed), the program executes the case that adds 2 to the counter. If the button is not pressed (False), the case structure ensures that 0 is added, so the counter remains unchanged. This setup allows for precise control of the counter increments based on the buttons pressed.

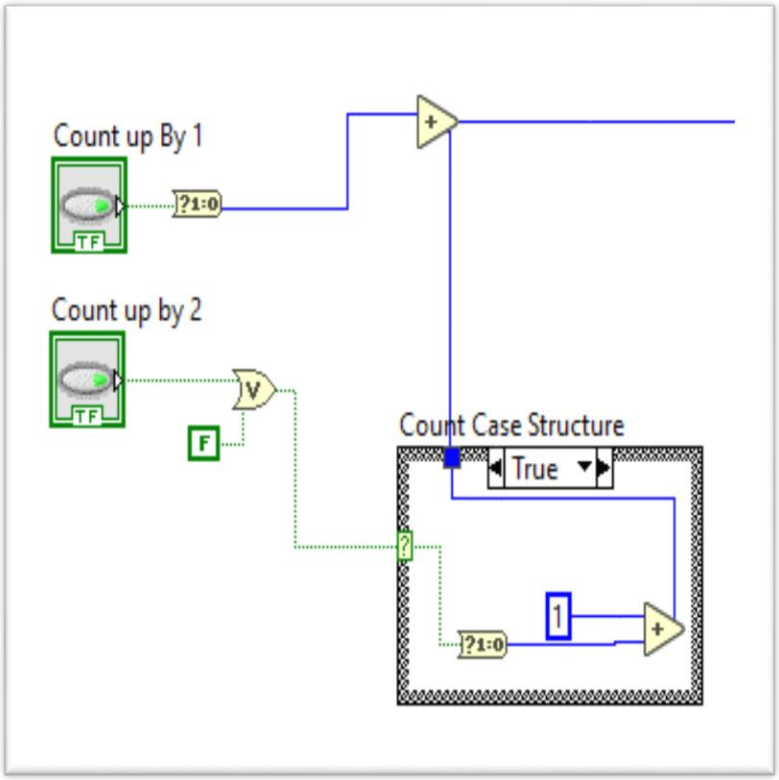

Figure 4. Count Buttons Logic.

The final result of the addition is then connected to the right side of the shift register, updating the counter with the new value. This updated value is also connected to the LEDs Case Structure, which controls the LEDs. Based on the new value stored in the shift register, the case structure determines which LEDs should light up to represent the current counter value in binary.

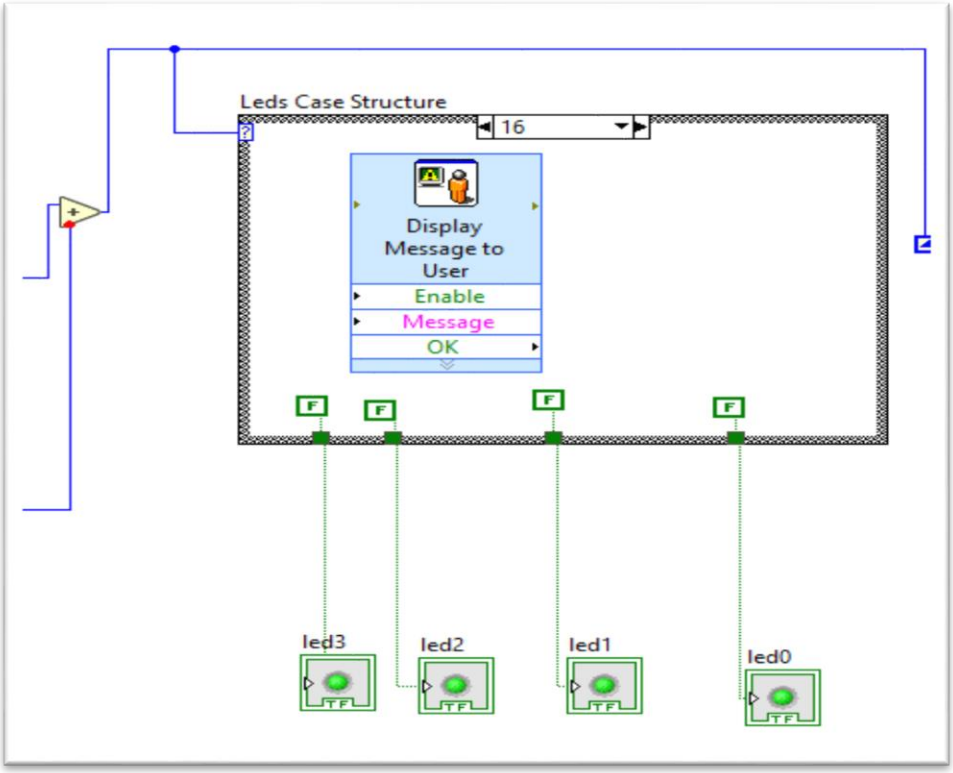

Figure 5. LEDs Case Structure

For example, if the counter reaches the value 4, it will be displayed as 0100 in binary. This means that LED2 will light up, while LED0, LED1, and LED3 will remain off, since the corresponding binary digits for these positions are 0.

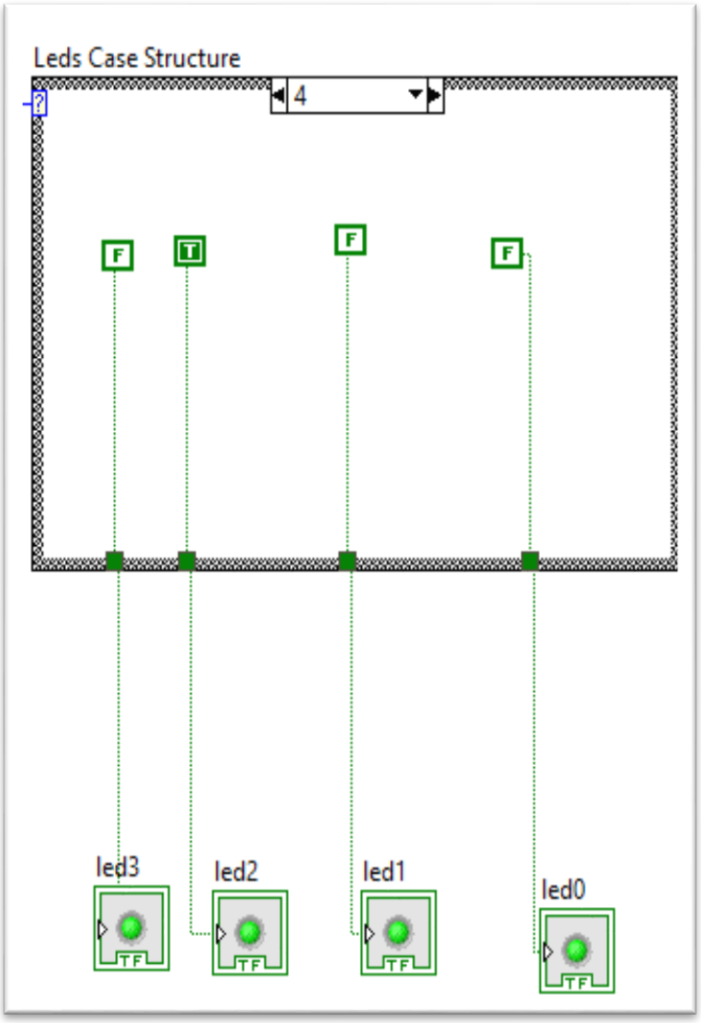

Figure 6. Case 4 of the LEDs

Another example is when the counter exceeds its limit. If the counter reaches 16 (which happens when the "Count Up by 1" button is pressed at the value 15, or the "Count Up by 2" button is pressed at 14) or 17 (when the "Count Up by 2" is pressed at 15), a message will pop up saying “Reached the Limit”. This message informs the user that the counter has gone beyond the allowed range.

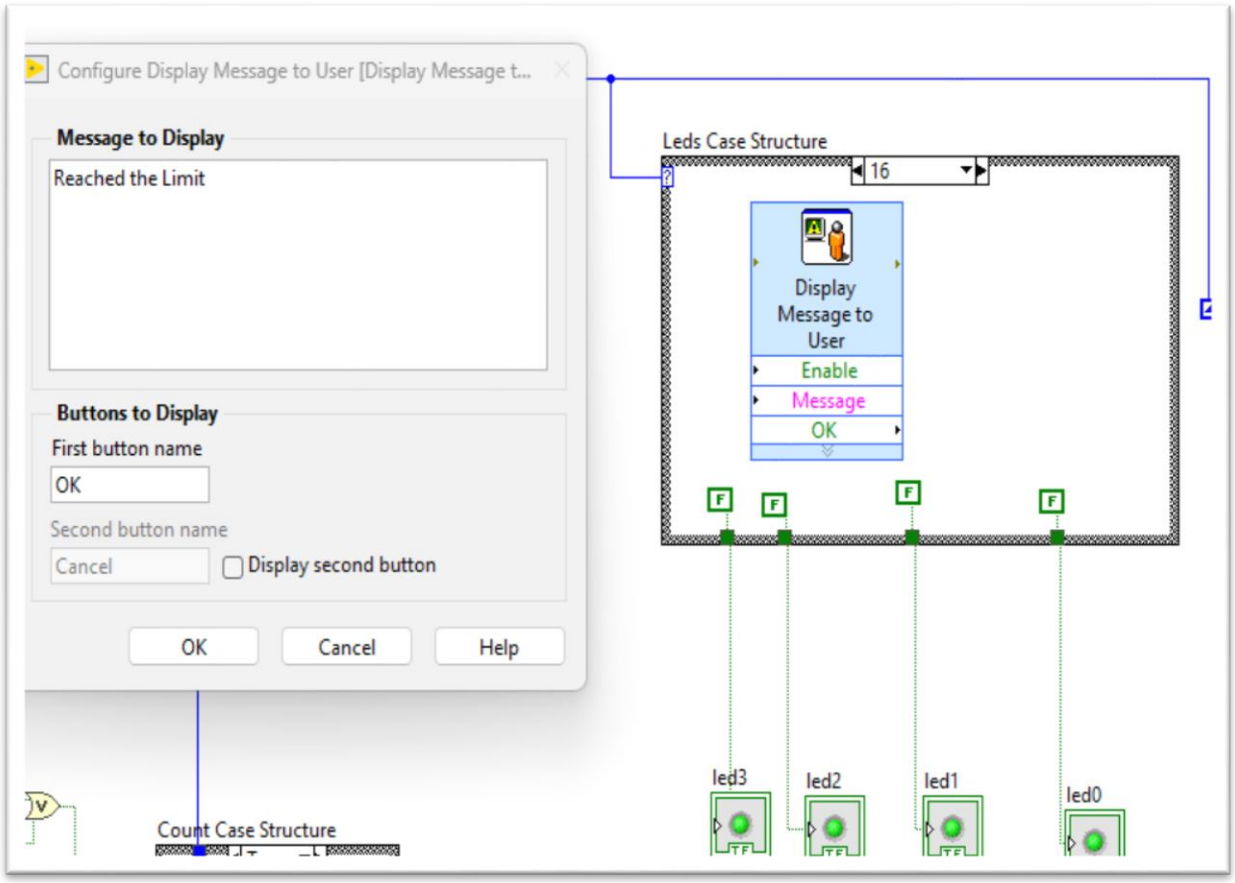

Figure 7.Case 16 of the LEDs

For the Reset Logic, a case structure is used to determine if the reset button is pressed. When the button is pressed, the case structure triggers the reset action. This is done by subtracting the value of the shift register from itself, effectively setting the counter to 0, which "resets" the counter. This ensures that no matter the current value of the counter, it will return to zero when the reset button is activated.

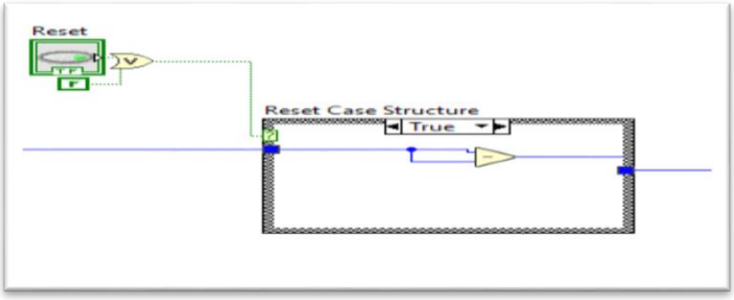

Figure 8.Reset Case Structure True.

If the reset button is not pressed, the case structure ensures that the current value of the shift register is simply added to 0, meaning the counter remains unchanged. This allows the counter to continue counting normally without resetting, as no action is taken to modify the value in this case.

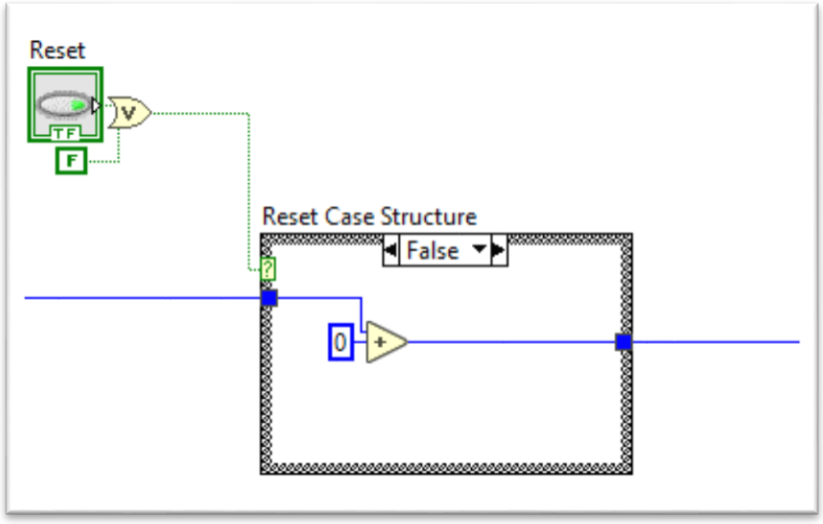

Figure 9.Reset Case Structure False.

Overall Design:

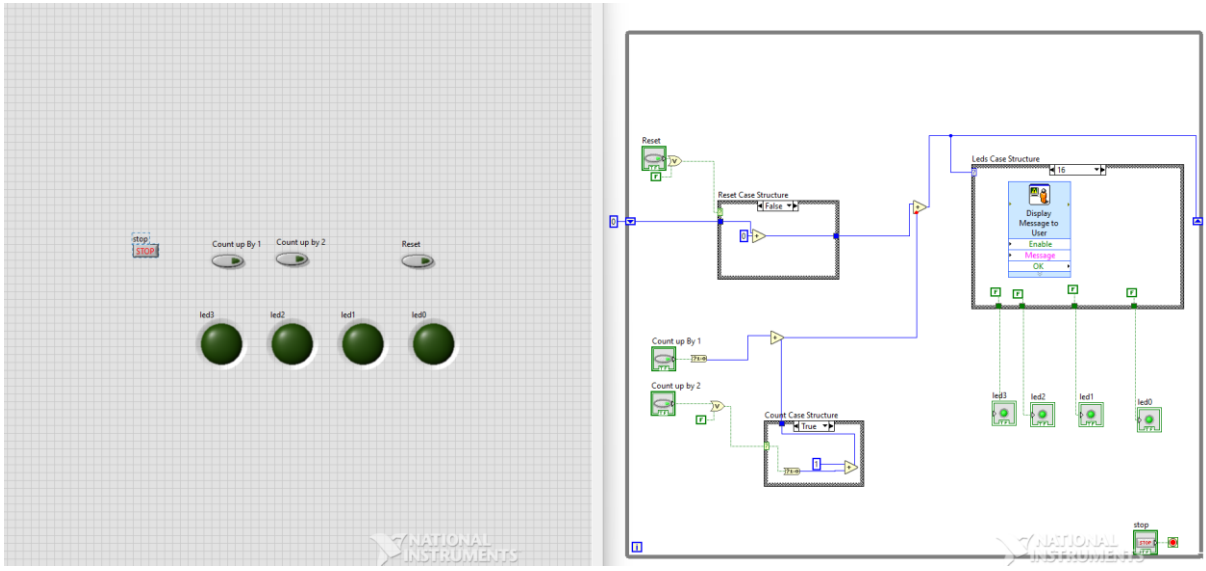

Figure 10.Overall design

A video was recorded for this project, and you can access it using the following link:

https://drive.google.com/file/d/1aBRZYf5MPQX3B2lPjBZeGFNy7SpW1zoo/view?usp=sharing

Conclusion

In conclusion, the implementation of the resettable counter system in LabVIEW demonstrates essential concepts in digital logic, user interface design, and programming. By integrating various components such as push buttons, shift registers, case structures, and LEDs, the project showcases how user inputs can dynamically affect counter operations. The use of a mechanical action for the buttons ensures reliable counting, while the binary LED display provides an intuitive visualization of the counter's value.

The counter logic effectively handles both incrementing and resetting the count, emphasizing the importance of feedback mechanisms through messages when limits are reached. Overall, this project not only reinforces the theoretical principles of counting systems but also highlights practical applications in embedded systems, paving the way for further exploration and development in digital electronics and real-time programming. The skills gained through this exercise will be beneficial in future projects, enhancing understanding and proficiency in using LabVIEW and similar environments.

ChatGPT was utilized for rephrasing various text elements to enhance clarity and readability. Its ability to generate alternative phrasing has proven helpful in ensuring that the content is conveyed effectively while maintaining the original meaning.