ME 344 Lab 1 Report

Christian Carbeau, Jay Parmar, and Winslow Griffen
January 26th, 2025

Introduction

In this lab, we delve into the fundamentals of LabVIEW. While the lab introduces essential concepts such as navigating the interface, data flow programming, and creating Virtual Instruments, the primary focus involves designing a sequencer that cycles through three steps with visual indicators. Specifically, we will program three Boolean LEDs and a string indicator to simulate a “Let’s Go Duke!” sequence, where each LED lights up in turn while displaying a corresponding message. This sequencer will operate continuously until stopped, demonstrating key LabVIEW features like loops, case structures, and state machines.

VI Design

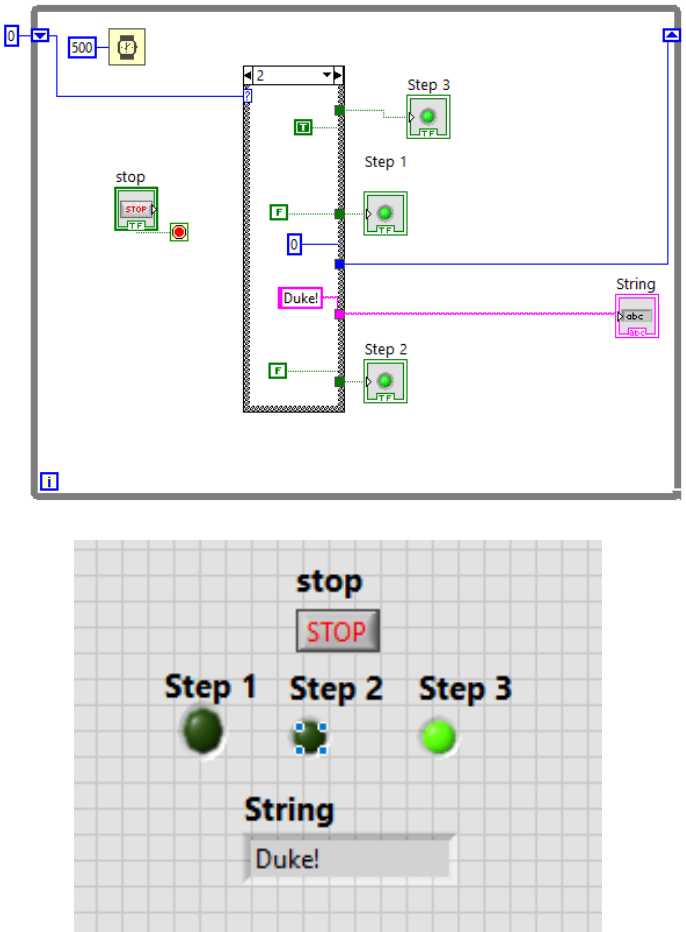

Figure 1: Block Diagram and Front Pannel of the VI Design Challange

A loop with a shift register is made around the entire program, the program initially starts with a constant 0 placed into the register. A 500 ms timer delegates the delay between loops. A stop button is attached to stop the program at any given point. A case structure is used to control the LED sequence and respective string output. Three shared LED outputs, “Step 1,” “Step 2,” and “Step 3” are placed outside of the case structure along with a string output titled “String.” Case 2, shown here, has three boolean constants that are wired to the LEDs outside of the structure and a string constant reading “Duke!” which is wired to the string output. We have the Step 3 LED wired to a true constant and the others are false. Additionally, there is a constant that refers to the number of the next case within the case structure, which is wired to the register in order to execute the following case in the next iteration. Cases 0 and 1 are formatted the same, with the string constant being “Let’s” and “Go” respectively. The boolean is ‘True’ for Step 1 LED in case 0 and ‘True’ for Step 2 LED for case 1, with all other LEDs being false. The constant number is 1 in case 0 and 2 for case 1.

Questions

1. **Block Diagram and Front Panel**

The front panel is the equivalent of a UI, where users can interact with buttons and features that are coded and wired in the block diagram, which serves as the motherboard of system.

2. **Mouse Icon Shape**

When wiring, the mouse takes the form of a spindle or spool of wire.

3. **Wire Color Significance**

The wire colors indicate the data type that that wire is carrying. For example, blue wires carry integer data, green wires carry boolean data, orange wires carry floating point data, and pink wires carry strings.

4. **Controls and Indicators on the Block Diagram**

The two can be differentiated visually because in the block diagram controls have data terminals on the right and indicators have terminals on the left. This allows wires to flow from left to right from controls to indicators in a visually easy way.

5. **Block Input and Output Locations**

True: inputs are typically on the left side of the block and outputs are typically on the right.

6. **LabView PEMDAS**

Part A: The subtraction occurs first because the wires dictate data flow from left to right and the subtraction block is to the left of the multiplication block.

Part B: Because the subtraction and addition blocks are not wired together and occur as the first blocks on the left in their respective sequences, they should run simultaneously.

7. **Wire Classification**

A is a 1D Array, B is a 2D Array, and C is an element data type, as distinguished by the number of wires and wire thickness.

8. **Element Display**

Because the index calls for “3”, the system will output “2010” because it is the 4th item in the array that aligns with an index of “3” (since the indexing system starts at zero.

9. **Block Analysis**

The shown block represents a split 1D array. The inputs are an array and index and the outputs are the first and second subarray. The function “divides array at index and returns the two portions with the element of index at the beginning of second subarray”.

10. **Find New Block**

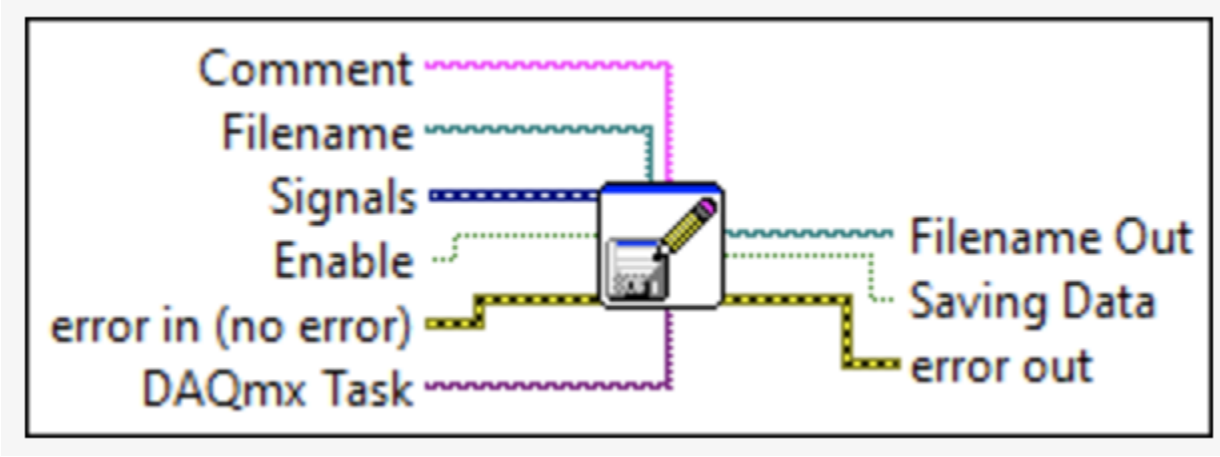

Figure 2: Write a Measurement File Function

The photo above displays an image of the “Write to a Measurement File” Function which is a part of the file I/O library. The block allows the user to write data to a file that occurs in your system to a file. This function will be useful in the future for some of our labs as we will need to record certain circuit metadata. The Context Help window displays the message “Writes data to text-based measurement files (.lvm), binary measurement files (.tdm or .tdms), or Microsoft Excel files (.xlsx).” for this block.

Authorship

This report was written in collaboration between Christian, Jay, and Winslow. Contributions were even between each member: Christian focused on the introduction and last question, Jay on the VI Design Challange, and Winslow on the rest of the questions.