Calculators

Zachary Hirtz, Ian Hanna

ECE-3130-001 – Microcomputer Systems

11/27/2024

Introduction

For this project, we decided to program a calculator. This implementation is important for speeding up calculations for accountants and businesses, ensuring accuracy in money computations and equations without human error. The calculator can add, subtract, multiply, and divide multi-digit numbers, following the order of operations. However, it does not support parentheses or negative numbers due to hardware limitations with the limited number of buttons.

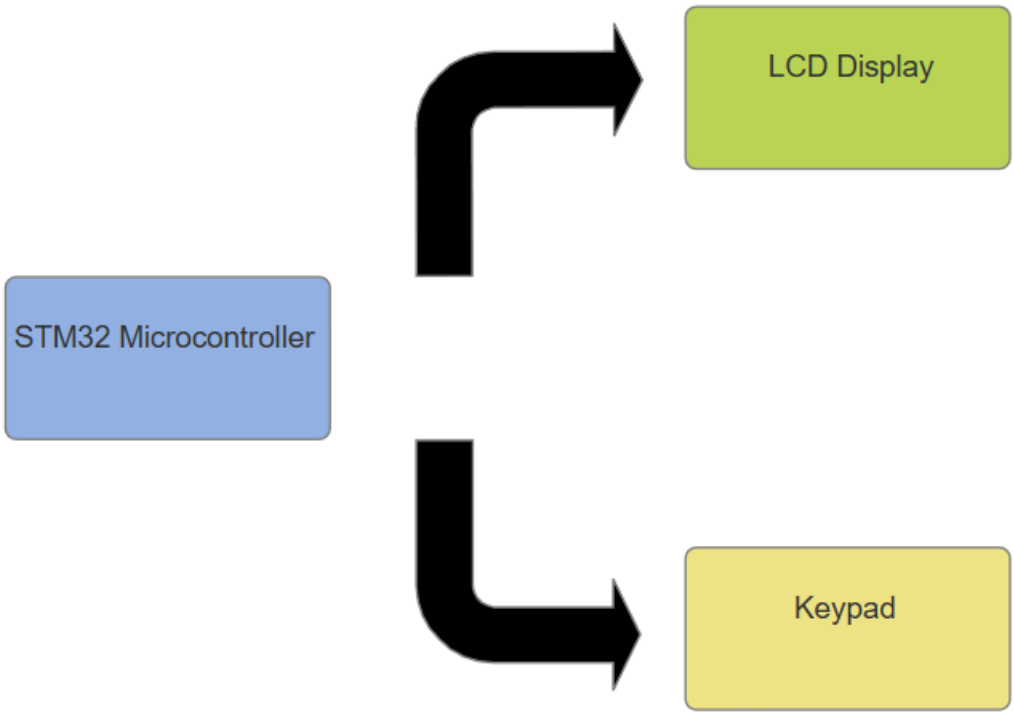
One problem we encountered was handling multiple operations in one calculation. We solved this by storing the numbers and operations in separate arrays. We then processed the operations array, performing multiplication and division first. After each operation, we removed the corresponding numbers and operations from the arrays and stored the result.

The code reads keypad input from the user, displaying the number or operation based on the input. It stores the value in the appropriate array, and if the number is multi-digit, it multiplies the number by ten and stores it when an operation key is pressed. When the equal key is pressed, the program first processes the multiplication and division operations by checking the operations array, updating the number array, and removing the used numbers and operations. It then performs addition and subtraction in the same manner, storing the final result. At the end, the total is printed on the second line of the LCD screen.

This report will go over the specifications of the STM32 Microcontroller and the hardware. Into the pseudocode and flowchart about the software. Into a description of the code and the c code itself. Then the report will go over the analysis of the project and how we tested it to make sure that it works. Finaly into the Description of the Teamwork Experience

Project Specifications and Description

We used a STM32 Microcontroller, a keypad, and an LCD display to create a basic calculator. First, the keypad and LCD are initialized by the STM32, then a function detects for if a key is pressed. If so, the entered numbers and operators are stored as arrays. When the ‘#’ key is pressed, the STM32 performs calculations on based on the numbers and operators. Multiplication and division are of higher precedence than addition and subtraction. The result is then displayed on the LCD display. When the ‘*’ key is pressed, all of the stored values are cleared and the screen is reset.



Detailed Implementation

Pseudocode

Display to LCD “Enter Numbers”

If (user press any key)

Clears LCD screen

While waiting for user to press a key

If (user input is a number) {

Display on LCD

Take current num time ten and add the number inputted by user}

If (user input is an operand) {

If (number array size is not max) {

store current number in array then index number array makes current number zero}

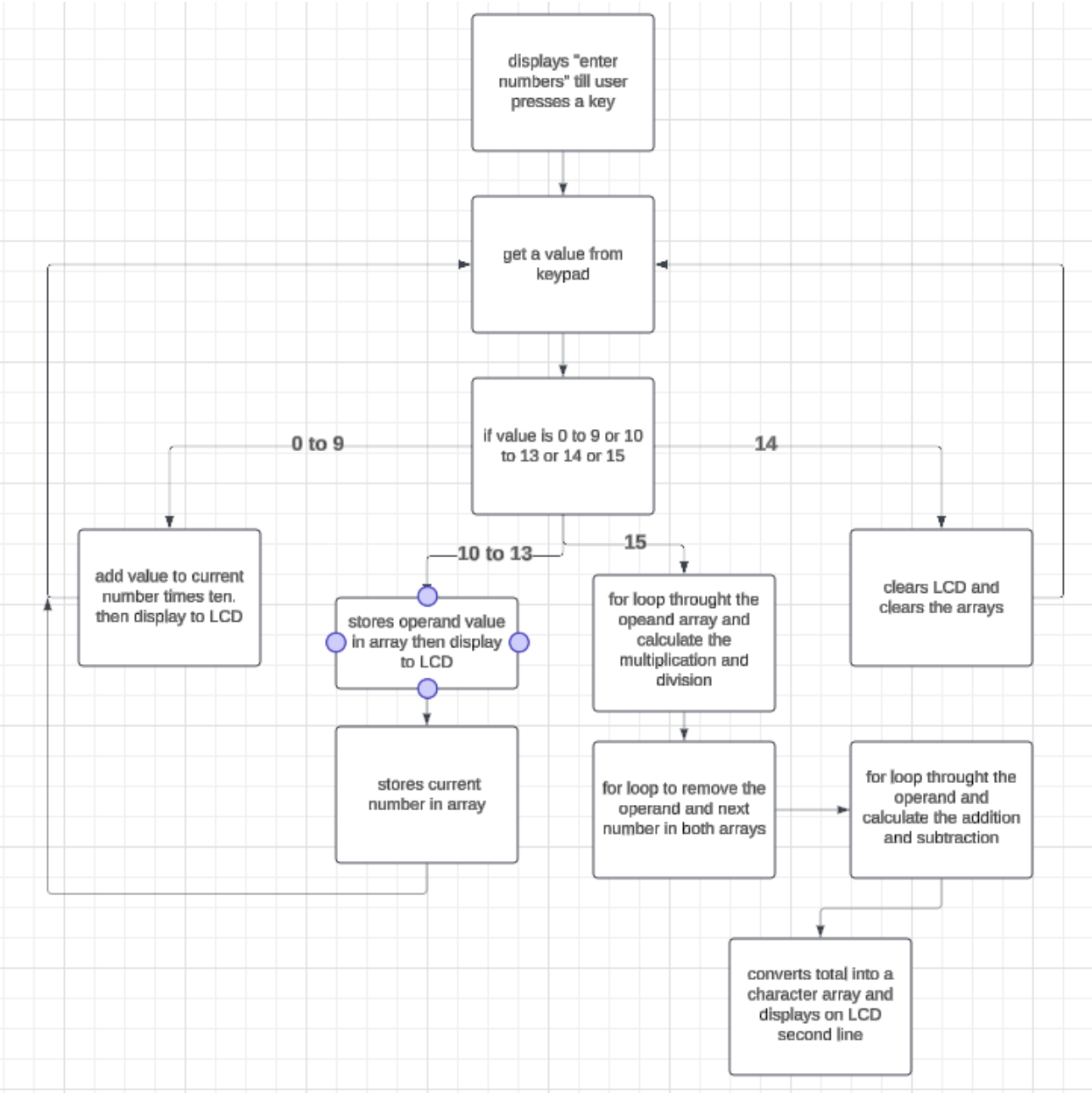
If (operand array size is not max) {

Store current operator in array then index operator array

```
        If (user input is addition, subtraction, multiplication, or division) {
        display the operand depending on what the user inputted}
        }
}
Else If (user inputs the equal key) {
If (number array size is not max) {
Store current number in number array and move to next index in array
        }
}
For (the number of operands in array) {
If (operand is *) {
take the number of the current operand index and multiply it by the number of next operand index}
        Else If (operand is /){
                take the number of the current operand index and multiply it by the number of next operand index}
        If (operand of that index position is * OR /)  {
        Loop through each array and delete the index number and operand
        }
        Decrement operand index and number index
For (the number of operands in array) {
If (operand is plus) {
take total value and add by current number}
Else if (operand is minis) {
takes total value and subtracts by the current number}
}
Print the value on the second line of the LCD screen
If (user input the clear key) {
```

Clears all the arrays numbers and operands}

Flowchart



Interface

For the project, the external interfaces utilized were the keypad and the LCD screen.

Microcontroller resource utilization

For the hardware, the I/O ports used for the keypad were inputs PB8, PB9, PB10, and PB11. The LCD screen I/O ports were set to outputs for PA5, PA10, and PB5. All the code is in

thread mode and it is privileged. For not having interrupts keeping the all the code in the start thread mode and did not change the privileged because it is the all in the main.

Software

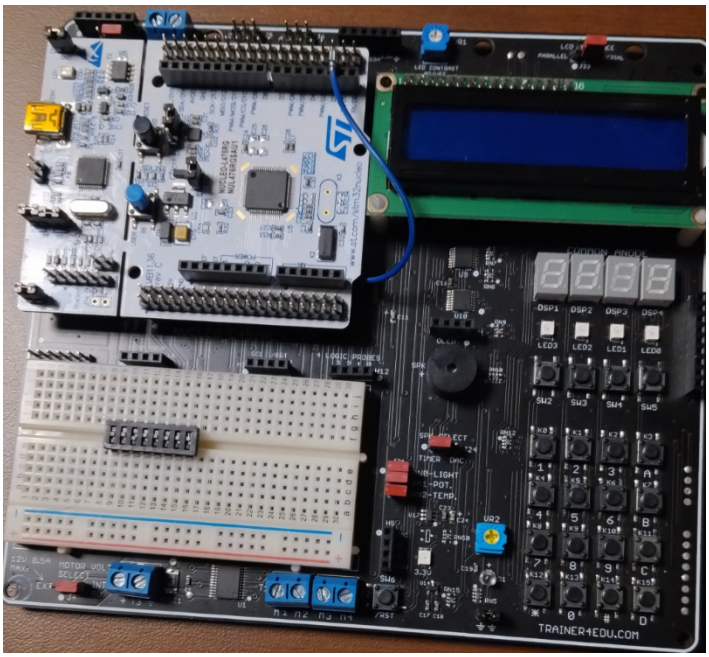
This program calculates multi-digit number equations with the operands of additions, subtraction, multiplication, and division. The software is structured in a while loop that will get an input from the keypad and will take the number and add it to current number times it by ten The calculator is designed to handle multi-digit inputs. When users input an operand, the program takes the operand value and stores it in an operand array, while the current number is stored in a number array. The program continues to loop through this process until the user inputs the equal value, at which point it stores the last current number in the array.

In a for loop, the program calculates multiplication and division first by iterating through the operand array. For each value, if it encounters a multiplication or division operator, it uses the index of the operand array to retrieve the corresponding numbers from the number array, performs the calculation, and stores the result back in the number array. It then removes the used operand and number from their respective arrays.

After completing the loop for multiplication and division, the program enters another loop to process addition and subtraction. Depending on the operand, it either adds or subtracts the numbers, updating the total accordingly.

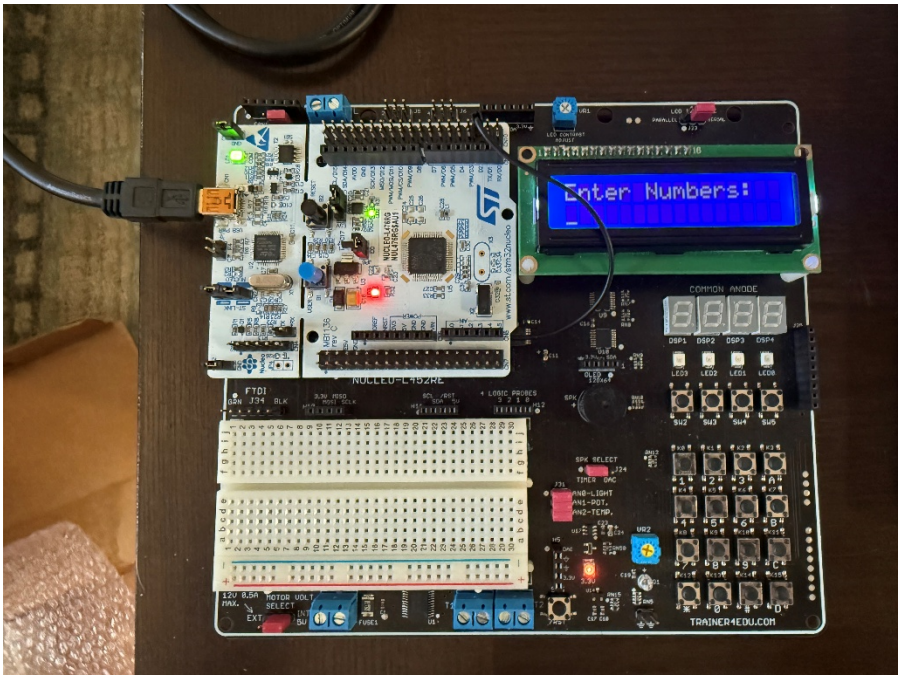
Finally, the total integer is converted to a char array, which is then printed on the second line of the LCD screen, allowing the program to exit the while loop.

Main.c code given

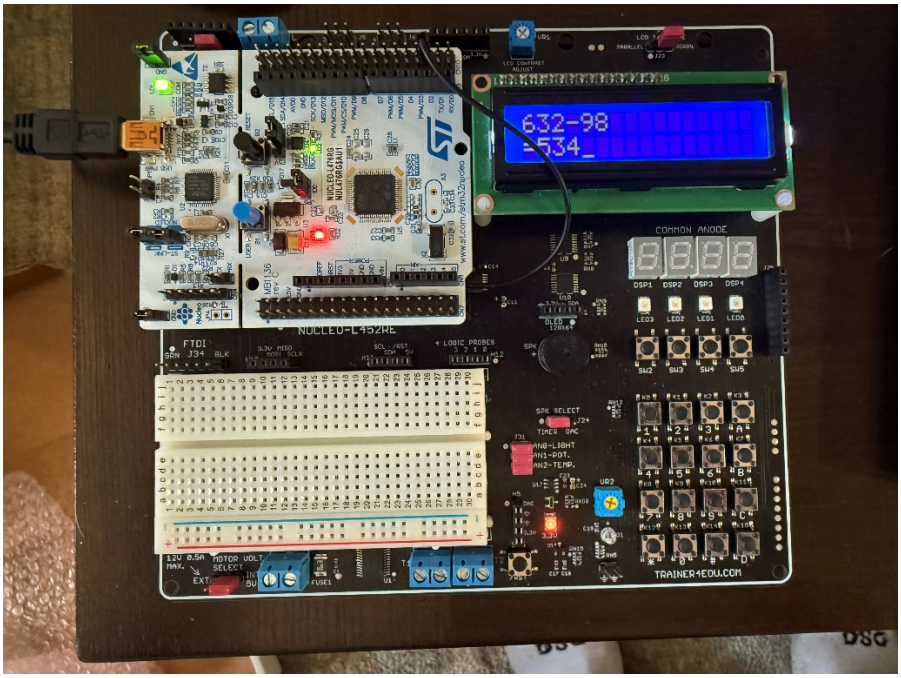


Analysis

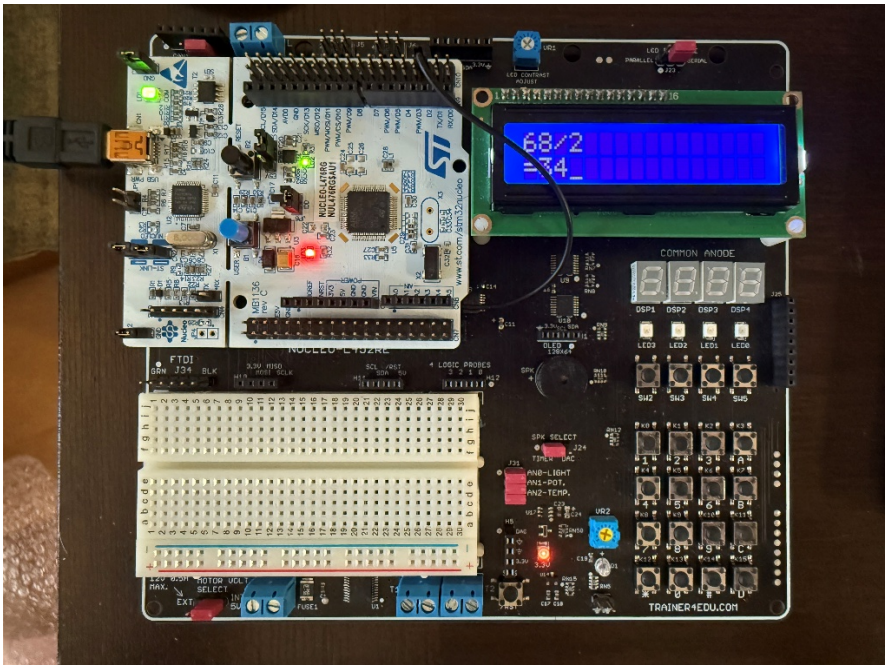
Each aspect of the project was tested individually. They keypad and LCD were verified to work based on the example code given in class. Most of the testing came down to ensuring that numbers were stored and displayed correctly, and that the arithmetic was correct. For operations that were outside of the scope of the calculator, we implemented an error message. We tested various sequences of numbers and operands that would yield results as well as sequences that would produce errors.



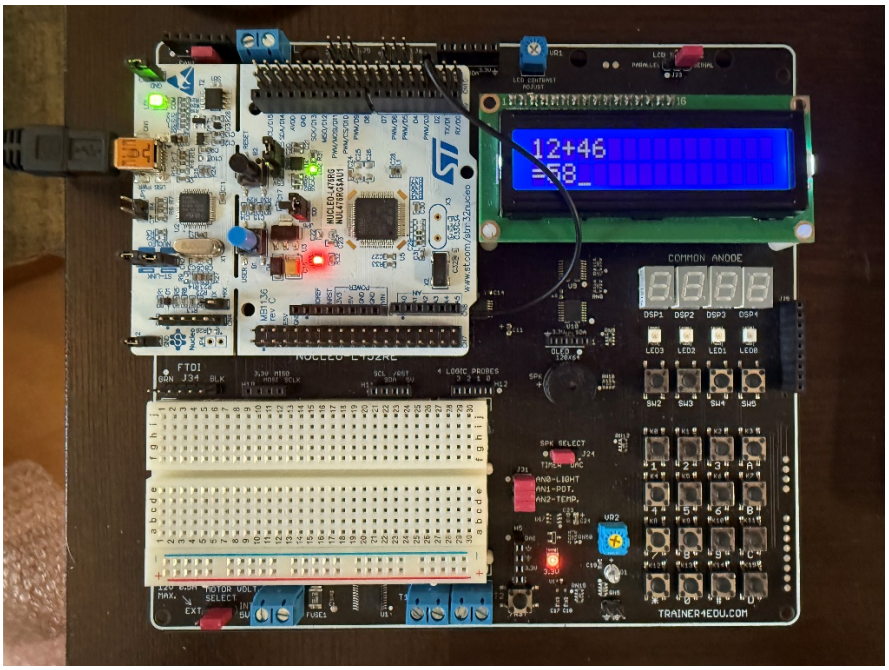
Prompt Screen



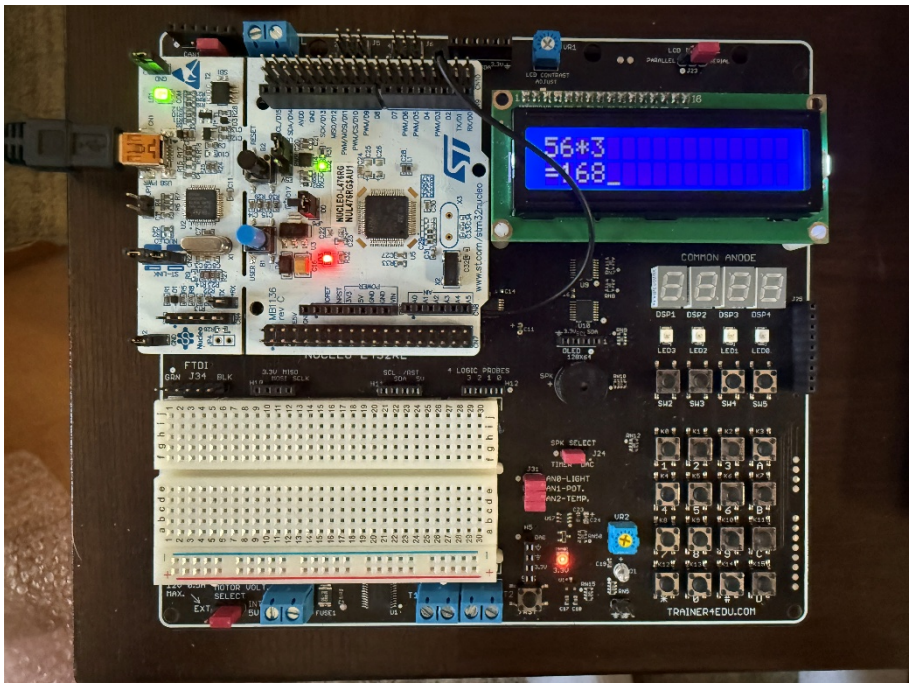
Subtraction



Division



Addition



Multiplication

1. Measures we put in place considering public health, safety, and welfare included making the device as simple as possible, making it easy for users to understand how to use the device and ensuring that all components of the device are secured to the motherboard tightly and free of any damage, reducing the risk of shorting and potentially harming the user. Some things we could have implemented to the design to increase safety, health and welfare are using an antimicrobial keypad to reduce the spread of germs in case of multiple users, and creating an enclosure for the device so that components could not be easily tampered with and potentially cause harm.
2. Considering global; cultural, environmental, and economic factors, we used universally recognized numbers and operands, so that the device can be used globally. The STM32 and LCD are low power components and cost effective. The device uses only a few, cost effective components, making it more affordable. Some things we could have improved on in those areas are adding different languages to the displayed text so that it can be understood outside of the English language, further optimizing the code and device to be simpler and with cheaper parts where possible to reduce cost of production.

Description of Teamwork Experience

1. To ensure fairness and equal contribution, we split the project into two distinct parts. One of us focused on detecting the keypad input and storing the values in the corresponding arrays, while the other worked on the calculation logic and display output. For the report section, we each handled different parts to share the workload evenly. To stay on track, we communicated our tasks and progress via email, sharing updates along with our main.c file.
2. We met regularly to discuss our vision for the project and the features we wanted to implement. We outlined the framework of the code and assigned sections to each team member. Communication was crucial; we kept each other informed about any changes, ensuring both of us fully understood the code. By breaking the project into manageable tasks, the workload became less daunting, making it feasible to meet our deadline.