Department of Electrical & Electronic Engineering

Brac University

Spring-2022

Course Code: EEE365

Course Title: Microprocessor and Interface Laboratory Project

Project Title: Simple calculator in LCD/7-segment displays capable of performing calculations (+, -, *, /, factorial, a^b) with reset option

Prepared by:

Name: Md. Jobaar Hossain ID: 19321018

Other Group members:

Sl.	***ID***	***Name***
1.	Md. Al Hasan Shaikat	19321001
2.	Md. Borhan Uddin Antor	19321025

Date of Submission: 25 th April, 2022

INDEX

SI	Content	Page No.
1	Introduction	01
2	Scopes and Objective	01
3	Apparatus and software	02
4	Project Specifications	02
5	Non-technical Constraints	03
6	Methodology	03-09
7	Design and alternatives	10-15
7.i	Design-01	10-11
7.ii	Design-02	12-13
7.iii	Design-03	14-15
7.iv	Justification of the best Design	15-16
8	Circuit Diagram	17-19
9	Results and Discussion	20-35
10	Applications	35
11	Future Scope	35-36
12	References	36
13	Appendices	36-58
14	Drive Link	58

I. Introduction

A microcontroller is a small chip of miniature computer that is built on a single metal-oxide-semiconductor (MOS) integrated circuit (IC) chip. A microcontroller is made up of one or more CPUs (processor cores), memory, and programmable input/output peripherals.

A tiny amount of RAM, as well as program memory in the form of ferroelectric RAM, EEPROM, NOR flash, or OTP ROM, is frequently provided on chip. We can use microcontrollers in almost every aspect of our life. By using a microcontroller we can do various types of work. We can calculate numbers, count time, make stop watches and so on. We can read, write, save, and delete a program from a microcontroller's flash memory, where the program instructs the chipset to do a certain task.

In our project, we have to make a simple calculator by using LCD/7 segment Displays capable of performing calculations with reset options.

In our project we use an ATmega32 microcontroller placed in the circuit inside that does all calculations smoothly, which would take a long time if done by hand. The primary function of a calculator is to do difficult calculations on huge numbers while saving a significant amount of time.

II. Scopes & objective

The main objective of our project is to design a simple calculator in LCD /7 segment displays which will able to perform mathematical operations such as

- Addition (+),
- subtraction (-),
- multiplication (×),
- division (÷),
- factorial (!) and
- power (a^b)

and it must be remembered that, we have to include a reset option. The calculator must also have numerical keys for inputting numbers. we may use the 6*4 keypads for that purpose.

III. Apparatus & software

Apparatus:

- ATmega32 Microcontroller
- 14 Segment Common Cathode Display(14 SEG-MPX8-CC-RED) *
- 7 Segment Cathode Display (7SEG-MPX8-CC-BLUE) *
- 16×2 Alphanumeric LCD display (LM016L)
- 4*4 Key pad Calculator
- 4*6 Kay pad Calculator
- 5V DC Power Source
- Connecting Wire

*In the apparatus we write the two respective names of the component, we also mentioned the proteus software code name of the component for convenience. We use star mark for this purpose

Software Tools:

- Proteus Simulation software (For Simulation)
- Code-Vision AVR (For code and HEX file purpose)
- Extreme Burner (For use run code in Microcontroller board)

IV. Project Specifications

In a simple calculator project, we have to design in a way where we can be able to do the expected calculation in our calculator. Our simple calculator will be capable of calculating the desired output if we may give him a correct input. In the proteus files we may use almost a big amount of number possibly 8 bits,

Mathematical symbols (+ , - , *, /, = ,^ ,!) and number keys from 0 to 9 in the keypad all designated to individual keys. The calculator can perform addition, subtraction, multiplication, division, power and factorial math. ON/C in the keypad for reset option. Here one thing must remember that the in the 6*4 Keypad. We may use the modulus key for the factorial purpose because we code them in such a manner and other uses on the power calculations.

In 4*4 keypad we use the other two buttons of the key, so that we may calculate the factorial and the power option. There are no keys for removing a single number or operator from the display. If a mistake is made while pushing keys, the ON/C button must be used to clear/reset the entire math and start over. It may take the positive and negative 8 bits value of the number. In our calculator, there is no use of a decimal point or any decimal value. After division, all values are shown in Quotient and Remainder form.

V. Non-Technical Constraints

There would be limitations if we calculate the over bit numbers like :5e-32. Because our expected simple calculator is used a limited bit to operate so that it may be impossible to operate in such a manner.

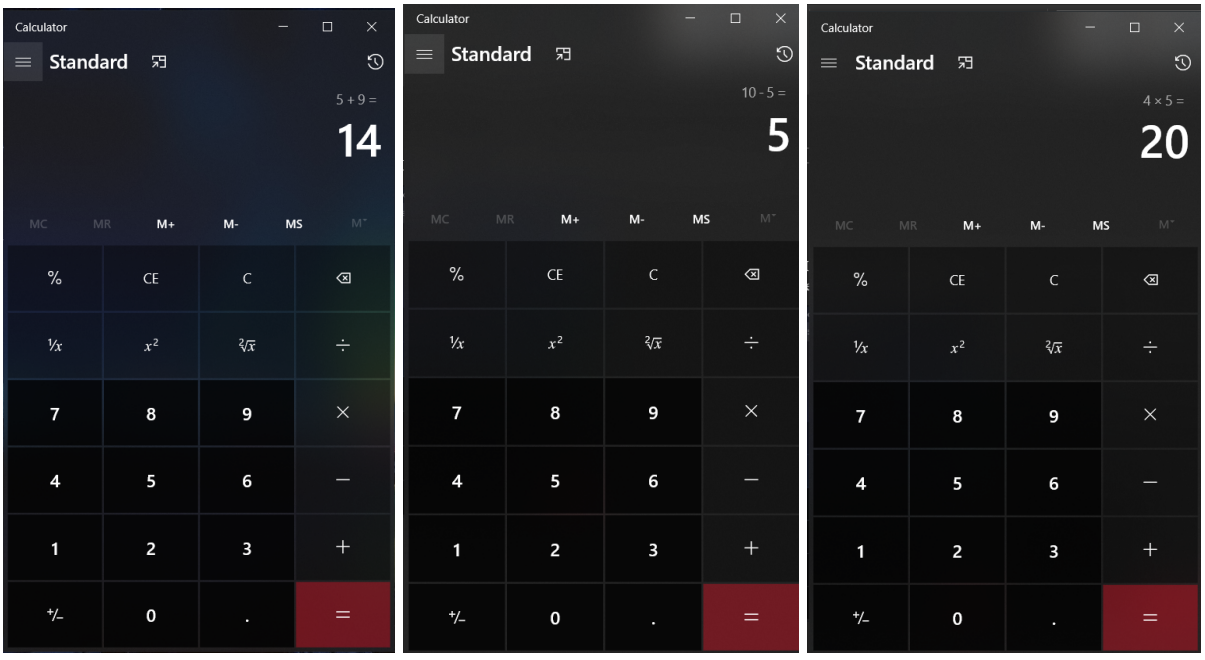
Sometimes the visibility process may take time, so we have to be patient during that time lap and also look after the operating temperature.Higher temperature may damage the component to work on.

We have to press the key in such a good manner. Do not press the keys in the keypad too hard, as they are a bit delicate and can break out or get stuck in. Sometimes pressing the keys rapidly makes the calculator not able to work properly. Keep a time gap of minimum 0.5s to 1s after each press for smooth working.

VI. Methodology

To make a simple calculator, we have to maintain proper methodology for these purposes. It must be remembered that a calculator receives numbers from a user (keypad) and displays them on the screen when the user chooses to perform mathematical operations on a number with another one. The calculator will display which operator the user presses. The calculator also displays the second number that will be used in conjunction with the first.

Here is the basic demonstration of a calculator. (Here, we use the windows calculator for demonstration purpose:



For implementation of this kind of simple calculator, we use atmega32, which means microcontroller.

For writing code and make output from the microcontroller, we follow/maintain some steps to obtain these:

Step-01: Display number and the sign of the keyboard

Step-02: Input value by pressing useful keyboard

Step-03: input correct operation and mathematical sign while calculating a number

Step-04: consciously check the button of factorial(!) and the power (^) while calculating a digits

Demonstration:

Step-01: Display number and the sign of the keyboard:

In our simple calculator project. Here we use two different keyboards to calculate the value of two different numbers. At these keyboards, those buttons each have individual functions to operate the different calculations purposes. There are some number buttons, operator buttons such as plus, minus, multiply, and an on/clear button to switch on or reset the calculator. Initially, we want to show the numbers or indicators in the display when certain specific buttons are hit. Let, we want to press 8 in the keyboard. Our code and others demonstration will show in to the display

In our calculator, we have to have a reset option that means whenever we press the ON/C button into our calculator, our program will notify us that the answer would be 0 which will clear after the respective calculations.
If the ON/C button is not touched, but any other key (2, 8, *, ! ,^) is pressed, the program will get a logic 0 from that specific button, and the program will recognize which button is pressed and show the assigned information of that button onto the display. Like in a matrix, the information of the key is tied to its location, and the computer will always identify the location first before displaying the right letters.

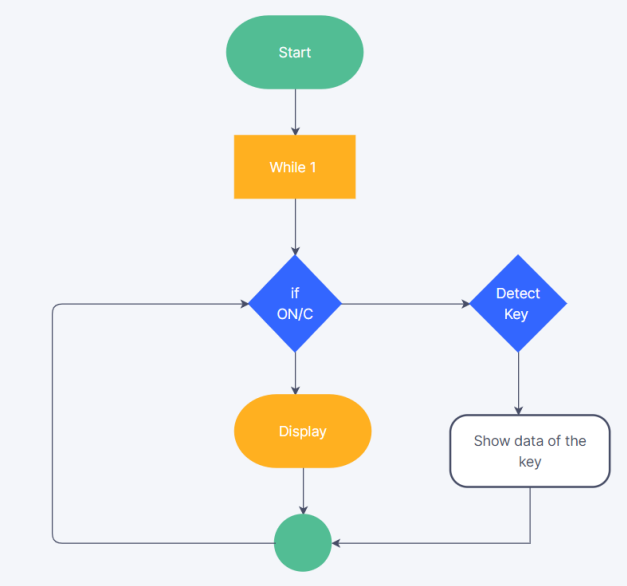


Figure: The Flowchart Representation of Our Step-01 Demonstration

Step-02: Input value by pressing useful keyboard:

In the last step, we are able to see how we press the keyboard and how the microcontroller actually displays in our press documentations. Basically, data was just simply shown in the display if that particular data key was pressed. When numbers or symbols are entered, the display displays them as "strings." Those characters are completely meaningless.

Suppose for if we input a big number let's say 366. It thinks that is just a string of "366". To input 366, press '3' key then '6' then '6'. Before that we define a variable name 'n' which is equals 0 that will store the value of 366.

So for **n**=0, pressing 3 makes **n = (0*10) +3.** This equation will be written in the program and delivered whenever the program detects that 3 is pressed. So now when **n**=3 and 1 is pressed, **n=(3*10)+6**, making **n** now equal to 36. Similarly, when 6 is pressed, **n=(36*10)+6**, which makes the value of **n= 366**. The equation for each individual number key is n = (n * 10) + num, where num is the number value assigned to that key. Finally, after setting the value of n equal to some number and pressing the ON/C button, the display will be cleared, as well as clearing and setting the value of n = 0.

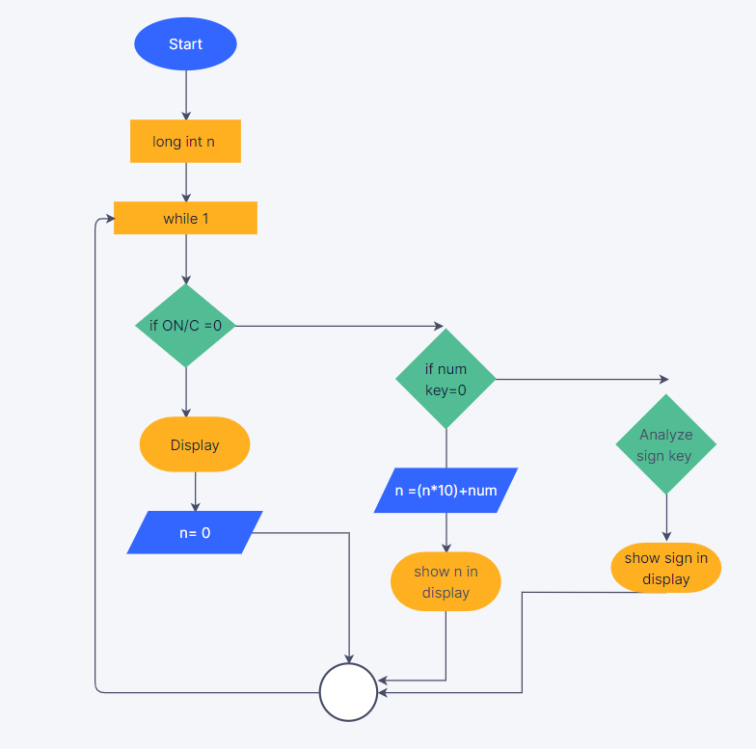


Figure:The Flowchart Representation of Our Step-02 Demonstration

Step-03: Input correct operation and mathematical sign while calculating a number

From the last two steps, we are demonstrating how we can use the basic things like press, Display on the screen and so on. For these, we have also notated our demonstration by creating Flowchart. For the first sign keys, there is no relevant function.

There are additional strings displayed in the display. We're going to fix it as well. Assume a user wants to perform a 7 + 2 computation. We must inform the calculator software of the meaning of the symbol '+,' which is plus. When the plus sign is used, the application must be configured in such a way that it performs the addition function. To calculate 7+2, first press button 7 and set the value of n to 7. When button + is pressed, we must preserve the value of n before doing the addition function, since we must also take the value of 2 and then conduct the addition

Now, we are not going to do addition when + is pushed because, after + is pushed, the calculator does not know yet with what other value the addition will be done as 2 is not pressed yet. We make the calculator do the addition after the equal sign is pressed. So whenever + is pushed, we set the value of a char **press** = 7, and when the calculator detects that = button is pushed, the program will first see if press is equal to 1 and then do the addition operation of 7 + 2 and show the value of the addition = 9.

For bigger calculations like 7+ 2+ 5, we need to add one more step inside the program for getting Appropriate value. Firstly, 7 is pressed and stored in **n** and then + is pressed so 7 is stored in **a**, **n** set to 0 and **press** is 2. When the next + is detected, now we want to put a condition that if the press is greater than 0, we perform the addition of 7 + 2 before = is pushed. So for + the program do **a = a + n**, which sets **a** = 9 as **a** =1 and **n** =4.

For the other sign the method is the same. Take the value of **n**, set **a** = **n** when the math sign is pressed and again take the value of **n**. Make **press** = any value other than 1 and 0. When the press is greater than 0 do the operation and show the value of the result when = is pressed.

In every If block in the flowchart, it is written like [=] key=0 or other expression for other keys because we are using an ATmega32 microcontroller inside the calculator to analyze which key is pressed in the calculator keypad. The microcontroller sees the keypad as a matrix. It sets all the PORTs connected to the columns of the keypad to logic 1 except one column which is set to logic 0 and the microcontroller shifts the logic 0 rapidly to the next column leaving the previous one logic high and continues to do that over and over.

The rows of the keypad are also connected to other PORTs. So when a key is pushed, the microcontroller will know which key is pressed by row scanning. ATmega32 detects the key by detecting the columns and rows as a logic 0 will be input into it.The calculator will be working all the time as the program is in a non-stop while loop that is as long the power button is ON and there is power.

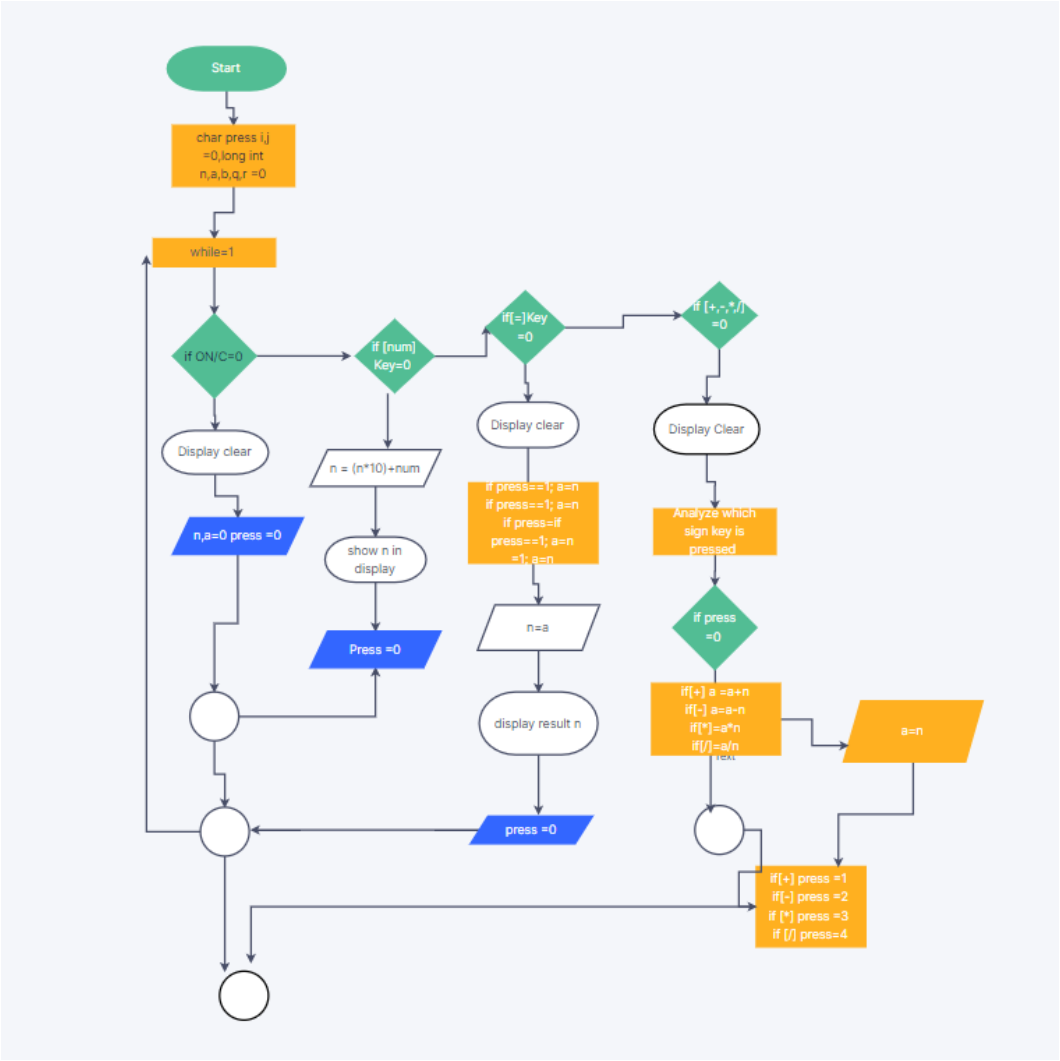


Figure:The Flowchart Representation of Our Step-03 Demonstration

NOTE: Flow-Chart Courtesy by: [Zenflowchart]

Step-04: Consciously check button of factorial(!) and the power(^) while calculating a digits:

In the last 3 steps, we demonstrated the way we can go to create a workable platform for our simple calculator. It was desired that our design must also have the ability to perform factorial and power operations.

Most simple calculators do not have these features except the scientific ones. Programming for these functions will be a bit different from the previous mathematical operators. When we perform a factorial on a number suppose 5, we write it like 5!. The result of 5! = 120 and we get this by doing (5 * 4 * 3 * 2 *1 = 120). We can write the C program for this by using a for **loop**. At first the user will insert a value by pressing his desired number. After that, when they press the "**!"** key, the calculator performs the factorial calculations.

Factorial code demonstration:

```
fact=1;
        for(i=a;i>0;i--)
        {
        fact*=i;
        }
```

Where " i " is a char.

So for **a= 5, i = a -1 = 5-1 =4** and 4 will be multiplied. Then i will be decremented to 3 and again multiplied to **a** which is now equal to 5*4. This will go on for **i > 0** or **i =1**. Thus we have the program for the factorial function. There is a special case for 0! which is equal to 1 and that cannot be calculated from this for loop. So we set a condition that if **a** equals 0, the answer is always 1. For factorial operation, we can make the result show immediately after the **!** is pressed because there are no more inputs so no need to press = key.

Power Code demonstration:

```
press=5
b=n;
n=a
for(j=1; j<b; j++){
n = n * a;
}
```

Where **j** is char, **b** is the power 3.
Basically in the loop, the value of **n = a = 5** will be multiplied by **a** which is also 5 for 3-1 =2 times. If ^ is pressed more than 1 times, the power operation will not work because **press** = 10 now and press is not equals to 5. So no results will be shown.

The divide operation is modified a little. Using previous expressions, if we divide 10 by 3 we will see the result as only 3. The remainder will not be shown. Two variable **q** and **r** are included that will calculate the quotient and remainder respectively by this expression:
q=a/n // to find the quotient.
r=a%n // to find the remainder
Then we can find both quotient and remainder and display them into the display.
So then 10/3 results:
Quotient: 3 Remainder: 1

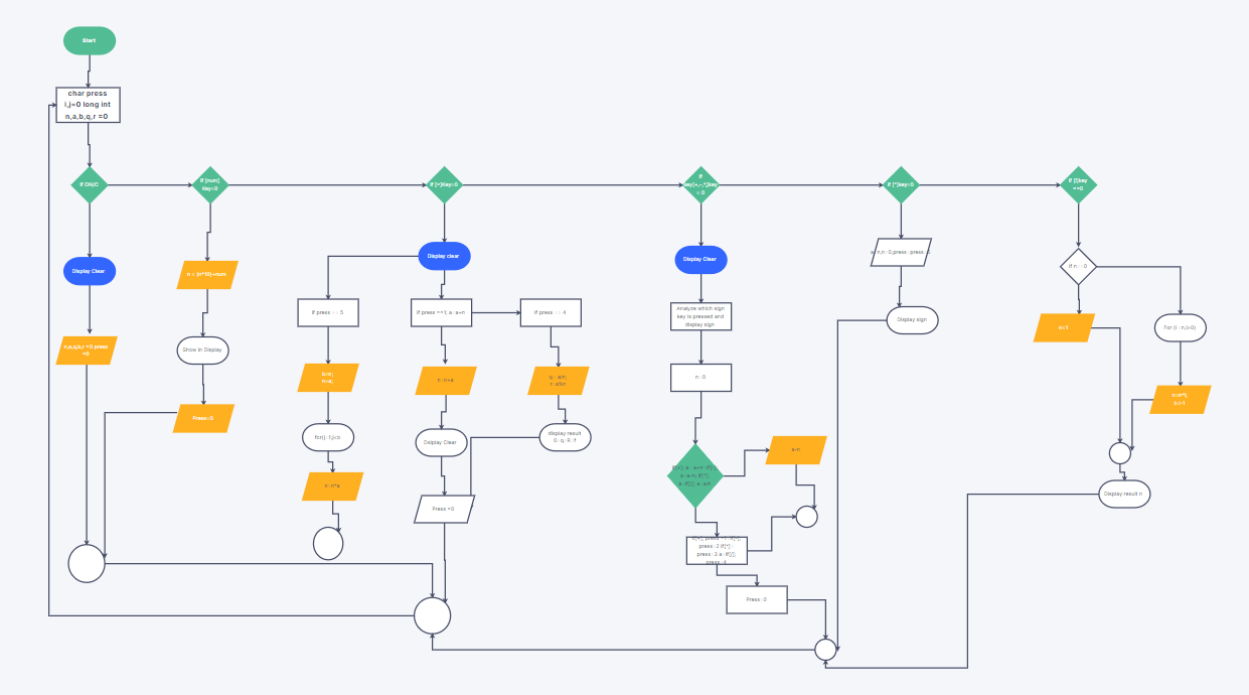


Figure:The Flowchart Representation of Our Step-04 Demonstration

NOTE: Flow-Chart Courtesy by: ZenFlowchart.com

VII. Design choices & alternatives

Design Approach-1:

Design using Alphanumeric LCD Display

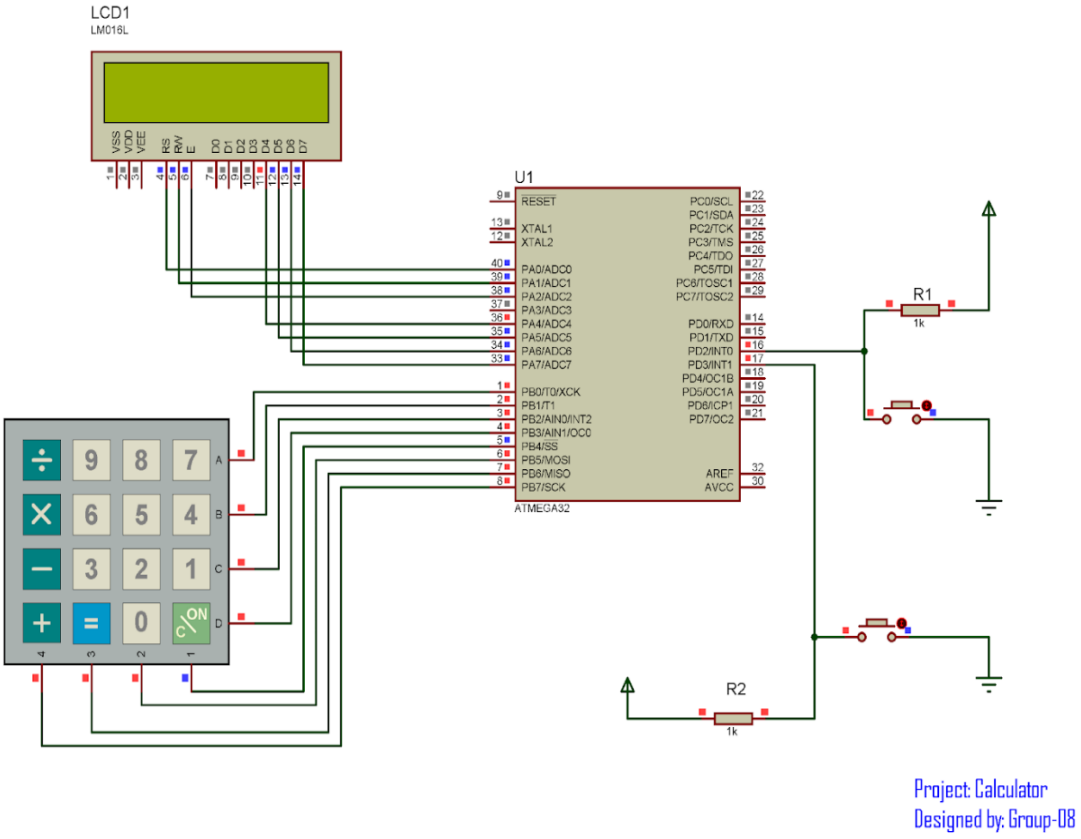


Figure:Simple Calculator Using Alphanumeric LCD Display

Description:

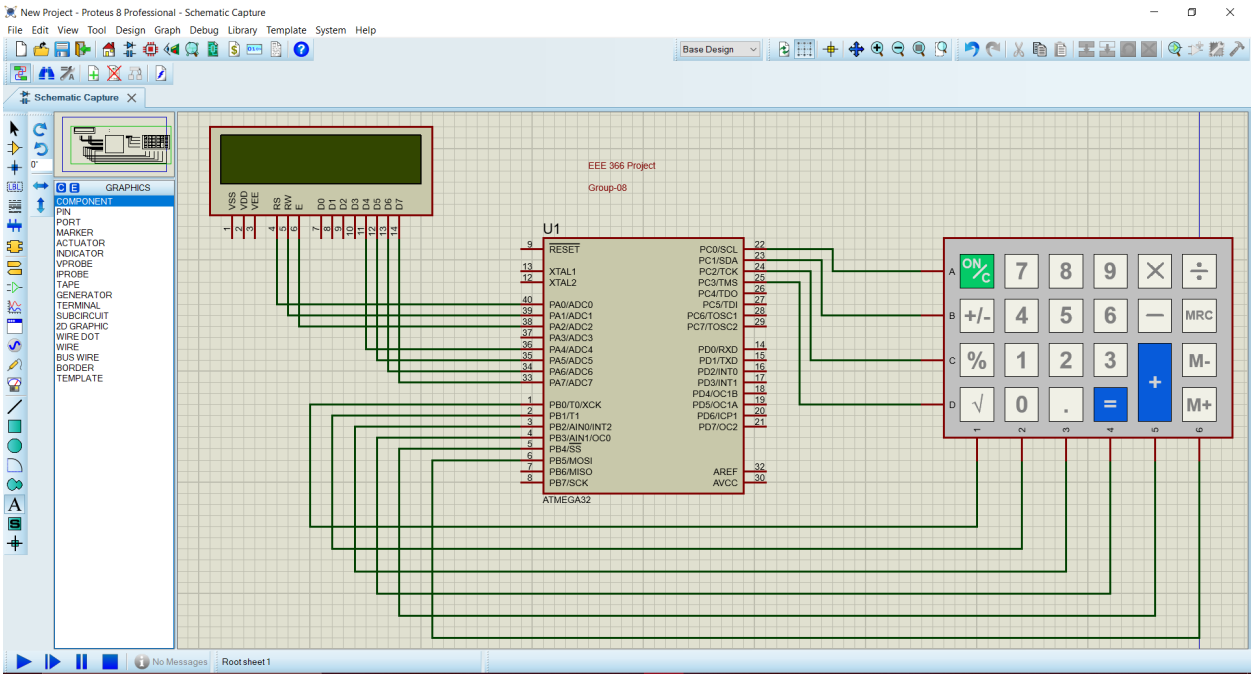
The C Code is developed and written using the logic of the flowchart. However some extra features were also added to the Calculator. For example, we used the 'Push Button' as a function of Interrupt and set it up with a resistor and power source.It is to be noted that we used the **Interrupt '0'** for showing the **factorial** operation. we used the 'Push Button' as a function of Interrupt and set it up with a resistor and power source.It is to be noted that we used the **Interrupt '1'** for showing the **power** operation.When the On/C button is pressed, 0 will appear in the top row of the display.

At first we declared header file #include <mega32.h> ,#include <delay.h>,#include <alcd.h>,#include <stdio.h>,#include <stdlib.h>.we defined DDRB as keypad_ddr, PORTB as keypad_port and PINB as input_data. We used arithmetic type specifiers unsigned *char*, *int*, *float*. Then, we used the num_get standard facet for parsing sequences of characters to obtain numerical values. Its member get is called by standard input streams to obtain formatted numerical values with the extraction operator .In this code we have used a total of 7 flags as operators. We have used flag 1,flag 2,flag 3,flag 4,flag 5,flag 6 & flag 7 as an operator of division, multiplication, minus, plus, equal, factorial & power. In the while loop condition we gave the condition flag=5 which is operator of equal. Then under this condition we used the if condition to operate the rest of the operation.As we have used a 4:4 keypad , there is no extra key to operate factorial and power. To solve this problem we have used interrupt 0 as factorial and interrupt 1 as power.

To display digits in the lcd display, we first of sent '0' only in column no 0. [c=0] (i.e. PB.4). Now if the row lines [PB.0 to PB.3] are read one by one, and if any '0' is read (say) from row 'r' [r=0 to 3], it will identify that the switch [in between column no. '0' and the specific row 'r'] is pressed. If all rows are read as '1', it will correspond to the 'no' key pressed in that column 0. Then send '0' to the other three columns one after another [column no. c=1 to 3] and similarly the row lines are read, if this process is continuously repeated and if any key is pressed, it will be detected in this process of scanning.

Design Approach-2:

Design using 4:6 keypad



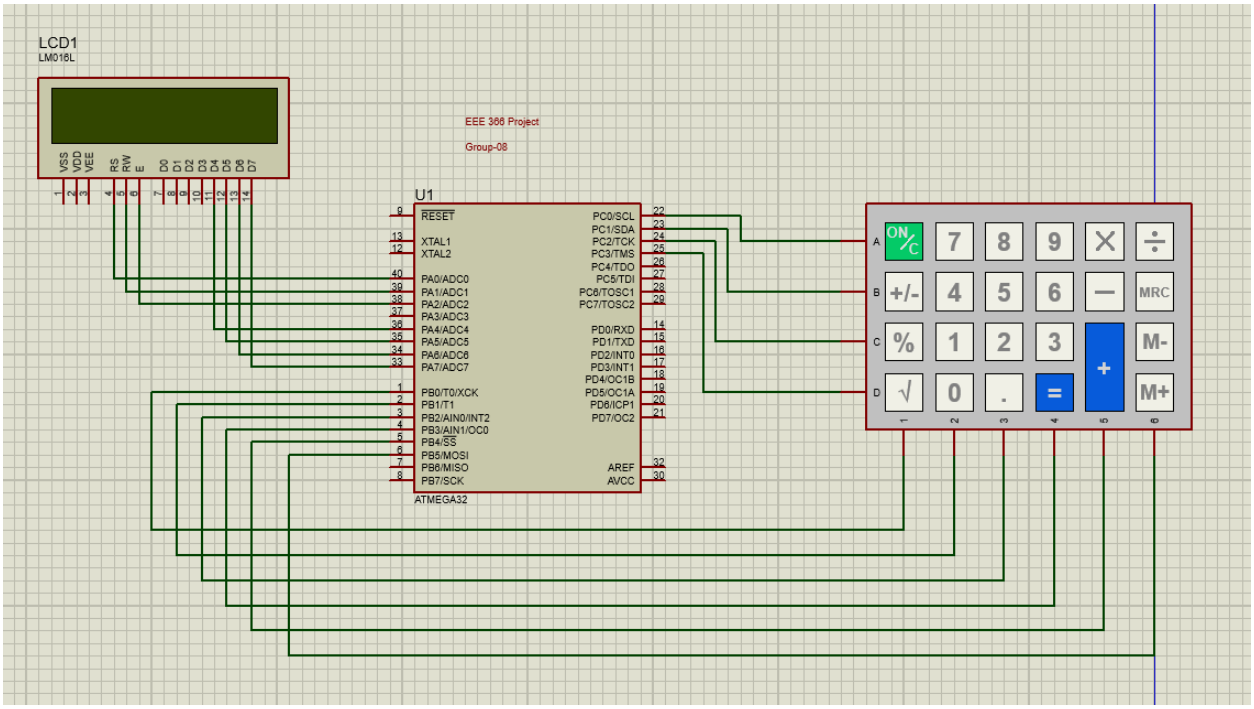


Figure: Simple Calculator Using 4:6 keypad

Description:

PORTA interfaces a 16*2 Alphanumeric LCD Display with an Atmega32. To send and receive data, the PORTB and PORTC pins are linked to the calculator keypad. When a number button is pressed, the LCD will display that number; if a sign button is pressed, the LCD will display the sign; and after a calculation, the LCD will display the result as a number.

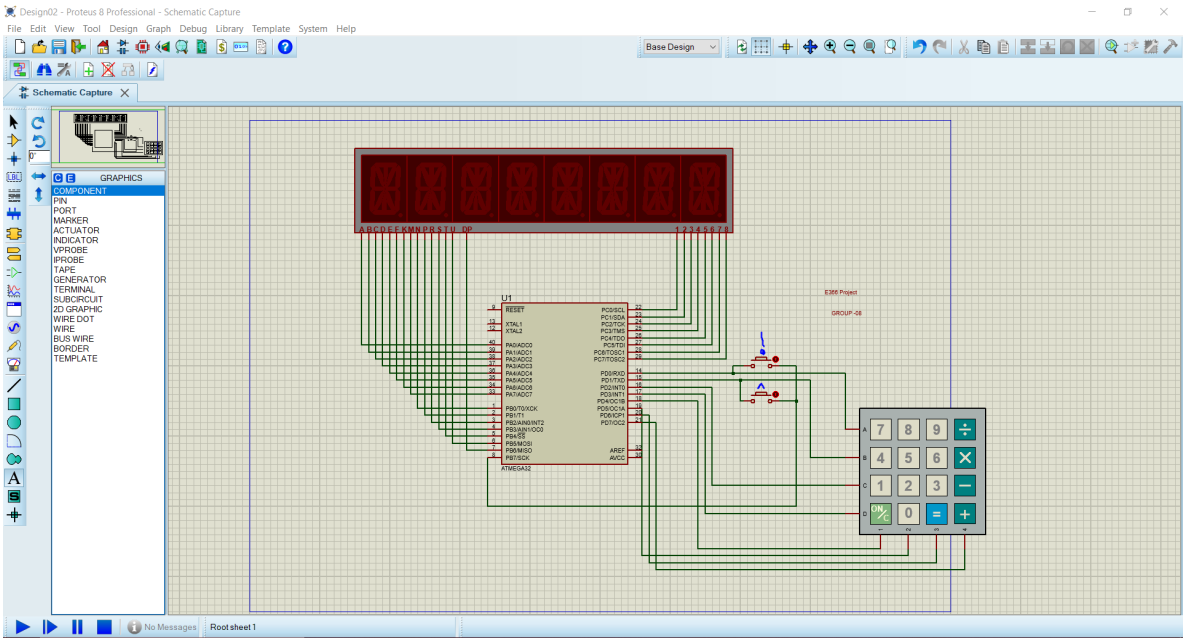
The 16*2 Alphanumeric LCD Display has 2 rows and 16 columns meaning it can show a maximum of 16 letters, numbers or symbols at any of the two rows. Each column can display one character. Each row can display sixteen characters.

It must be remembered that LCD (Liquid Crystal Display) displays are unrelated to 7 or 14 segment displays, and unlike them, LCD uses fluorescent lamps as backlight at the back of the display rather than LEDs. They all, however, fulfill the same function of showing data. As data, an LCD display may display any combination of numbers, characters, and symbols.

The C Code is developed and written using the logic of the flowchart. However some extra features were also added to the Calculator. For example there can be a key [+/-] that can make any input number negative. When the On/C button is pressed, 0 will appear in the top row of the display. The value of any number cannot be greater than 2000000000 and smaller than -2000000000 in any input or calculation. If the value exceeds this range, the Calculator will show MATH ERROR. Also if any number is divided by 0 the display will show MATH ERROR.

Design Approach-3:

Design using Fourteen Segment Display



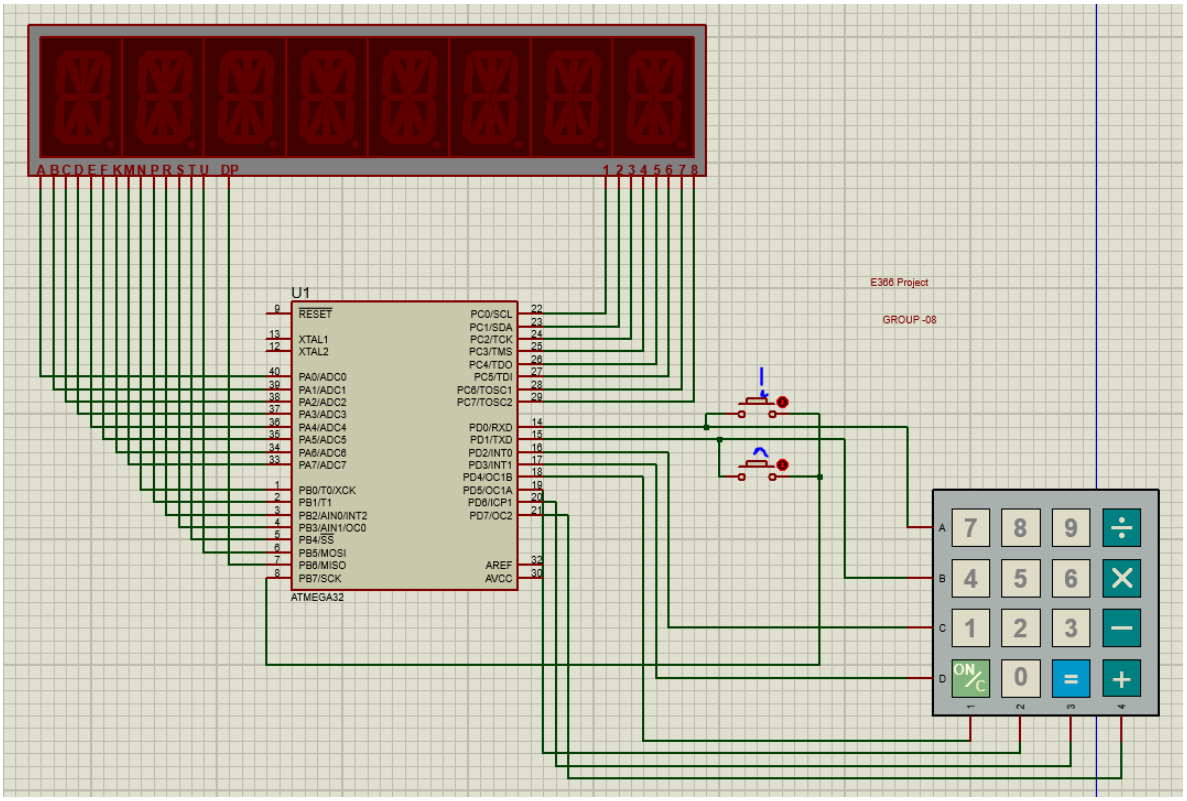


Figure: Simple Calculator Using Fourteen Segment Display

Description:

LEDs are used in each segment of a common cathode fourteen segment display, much like in a seven segment display.

However, as the name implies, there are fourteen segments that may display not just numbers, but also characters and symbols. There are fourteen data pins linked to the segments, which allow each to light separately and show numbers, alphabets, and signs. There is also a pin for DP to light up a circular section to indicate a dot or decimal. Of course, the display features control pins that allow us to transfer data to a specific location on the display.

All of the PORTA, PORTC, and PORTB pins, with the exception of PIN7 of PORTB, are linked to the fourteen segment display. PORTC delivers data for position selection. Calculator Keypad communicates with the microcontroller by connecting to all PORTD pins and PIN7 of PORTB. We must improve the design by employing a tiny Calculator Keypad and attaching two more buttons for the power () and factorial (!) functions.

When a numerical button on the calculator keypad is hit, that number is shown; when the second number is touched, the first number shifts left to make room for the following number. This procedure will continue even for operators. The display will additionally indicate which arithmetic operation is being done by displaying "+" for addition, "" for multiplication, and so on. When the equal button is clicked, the display will show "=" and the numerical result of the final number after computation.

Justification of the best Approach:

We have approached the project with three alternate designs that are, by using 4*4 keypad with display and an atmega32, 6*4 keypad with display and an atmega3 and by 7 segment display. However, we will choose only the best one from them as a subsystem for our final circuit design.

In our perspective ,the best approach or best design will be design-01. We already demonstrate all the proteus output from the design-01 and we will also show the exact same hardware demonstration in our lab. Design-02 and Design-03 we also code and simulate but due to the increase of page number of hardcopy we are not going to show this graphical/proteus calculation

but already mention the google drive where we find the proteus and exactly the same code of the code vision AVR.

Initially, at our design-01, we spontaneously do any kind of numerical operations with the especially, we can do the higher bit of number. In this project we use the keypad as well as the button in our simulation files.we are already out of the picture of results how these work.

secondly, the design-02 Display of alphanumeric characters on an LCD screen. The limitations of LED segmented displays are not applicable to alphanumeric LCD. Because an LCD panel can display all numbers, alphabets, and signs, there is no chance of user misunderstanding while conveying information. In addition, LCD panels may display 16 characters or numbers in one of the two rows, for a total of four rows.times the number of 7 and 14 segment displays, using just 7 I/O pins of the Data will be received by a microcontroller. As a result, we have more free ports to employ for larger interfaces.If necessary, upgrade keypads or other devices. Another benefit of LCD displays is that they may be customized.are more portable than the other types of displays

Thirdly, the design-03 Seven segment display with the potential to show up to eight digit numbers. If the result of a math computation exceeds a number with more than 8 digits, the result cannot be shown appropriately. We are also unable to add another display since Alternate Design 1 has insufficient open I/O ports, and Alternate Design 2 has none to bring and attach another monitor. The majority of the ports are already in use to link the microcontroller to the first display. To use one 8-digit fourteen segment display, we must reserve 23 of the Atmega32's 32 I/O pins.

Comparatively, we told that design-01 is best among three because, we have constraints of 7 segment display, we use a specific bit of number in our design-03 like “23+234=” here we will not able do the calculation it would be better if we do a 1 bit simple calculation. but at design 1 we can overflow our numbers. We can do “236+456=962” these types of calculations. our second design keypad 6*4 is not available in the market even though we code these in a way where some keyboard like % use as factorial which is not that much useful, by these we can say our 1st design will be best among the threes.

VIII. Circuit Diagram:

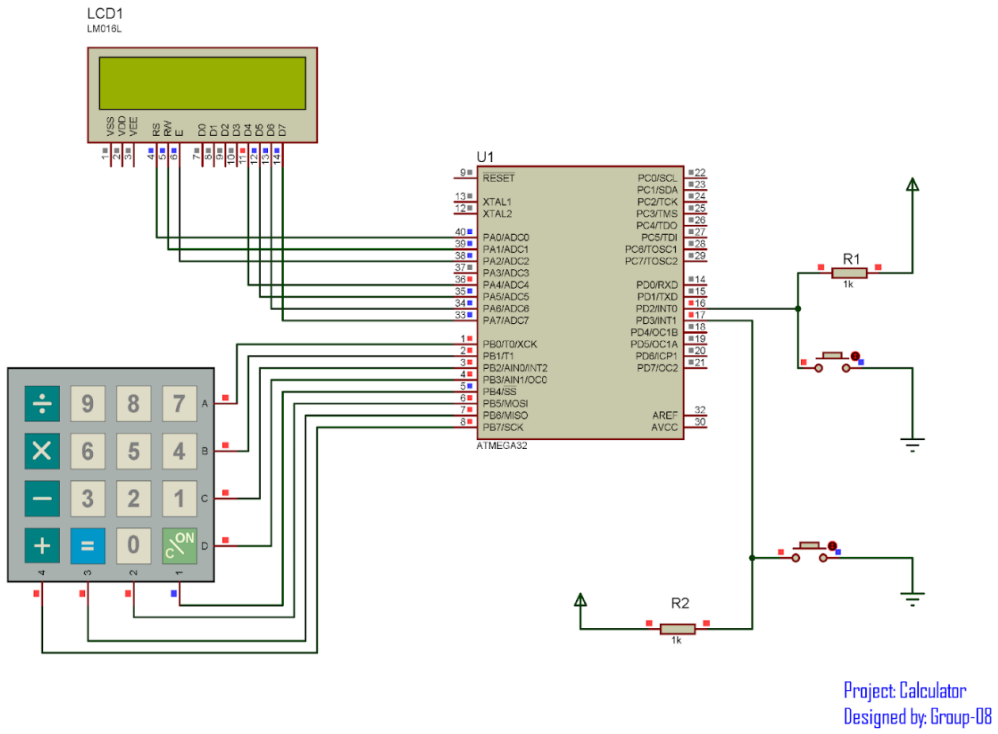
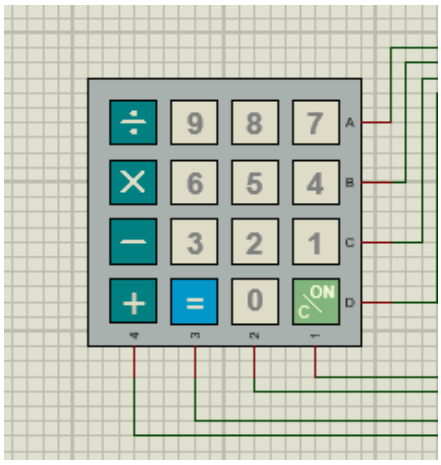
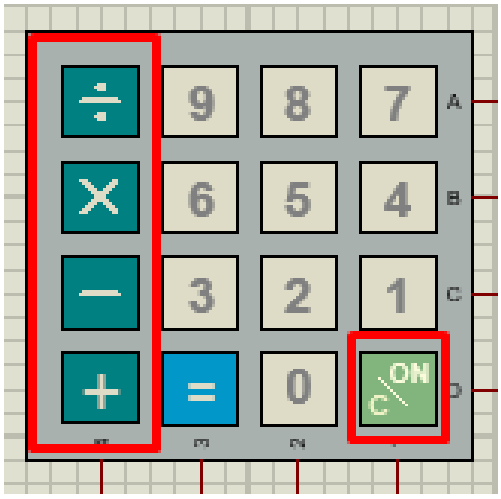


Figure: Simple Calculator Using Alphanumeric LCD Display

In this circuit diagram, which will show the hardware in our lab. We use the design-01 for the presentation purpose because we already mentioned the justification of our lab. Here we use an Atmega32, keyboard and alphanumeric display, here we also use the 2 buttons along with a 1 kilo ohms resistor.

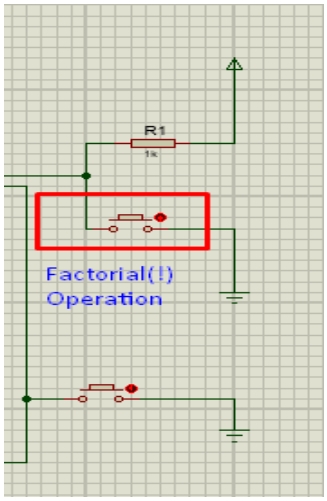


In our design, exactly the keypad shown, if you are seen in the keypad.Here, we use the 4*4 Keypad in our keypad, we try to create exactly the same manner of port.For our best design, we have used this keypad as it is affordable and it will be found in the market. In the case of using this keypad, we need to have some modification in our design to exhibit the factorial(!) and power(a^b) operation in our calculator.

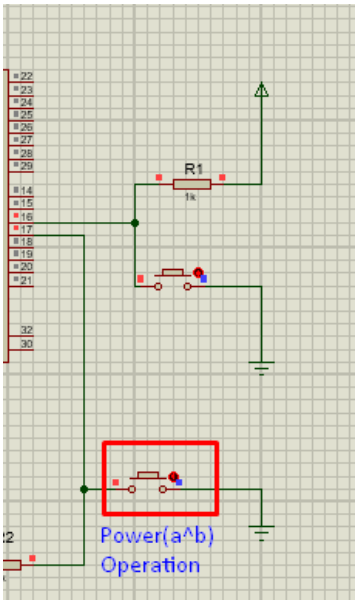


In terms of using this keypad in our design no-1, we had to do some modifications. However, the indicated column in this picture with red color containing the mathematical operators served their self purposes in our design. In addition, the ON/C button worked as the reset button in our design manual. The rest of the buttons are as it is and have their own individual functions

For this if we press the "+" keypad. the addition symbol will be visible in the display, similarly this if we press the "+" keypad. The additional symbol will be visible in the display. this if we press the "-" keypad. the subtraction symbol will visible in the display this if we press the "*" keypad. the multiple symbols will be visible in the display. In our question, It must say there will be a reset option. We use the "ON/C" button for the rest matter.



Here, for exhibiting the operation of Factorial ,we had no way but to modify our design in such a way so that the factorial works properly and fulfill the goals.In that case, we used the 'Push Button' as a function of Interrupt and set it up with a resistor and power source.It is to be noted that we used the Interrupt '0' for showing the factorial operation .Through this modification, we could ensure the availability of every buttons of the keypad have their own functions.



Here, for exhibiting the operation of Power ,we had no way but to modify our design in such a way so that the Power works properly and fulfill the goals.In that case, we used the 'Push Button' as a function of

Interrupt and set it up with a resistor and power source.It is to be noted that we used the Interrupt '1' for showing the power operation .Through this modification, we could ensure the availability of every buttons of the keypad have their own functions.

IX: Results and Discussion:

Let us test our calculator performance and see whether it is doing the calculations correctly. basic calculation (+,-,*,/) we use the windows calculator and for the factorial we use the online calculator for these purposes.

Addition(+):

Software Simulations:

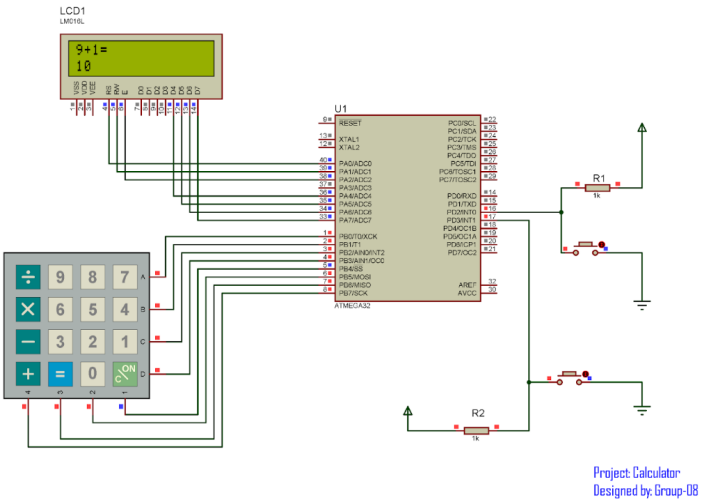


Figure: Exhibiting Addition Operation [9+1=10]

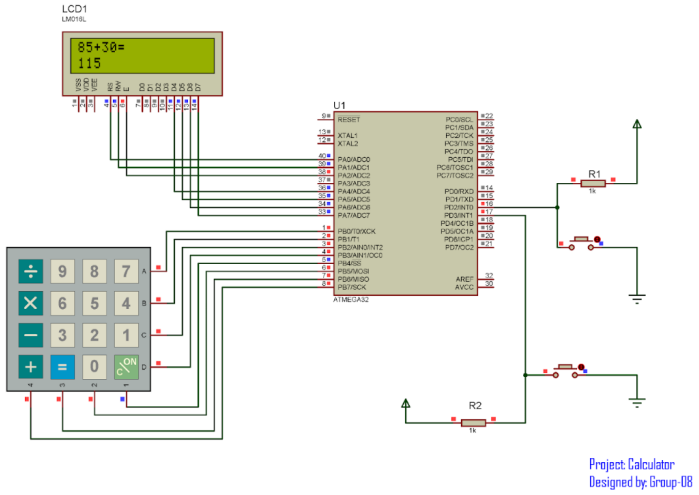
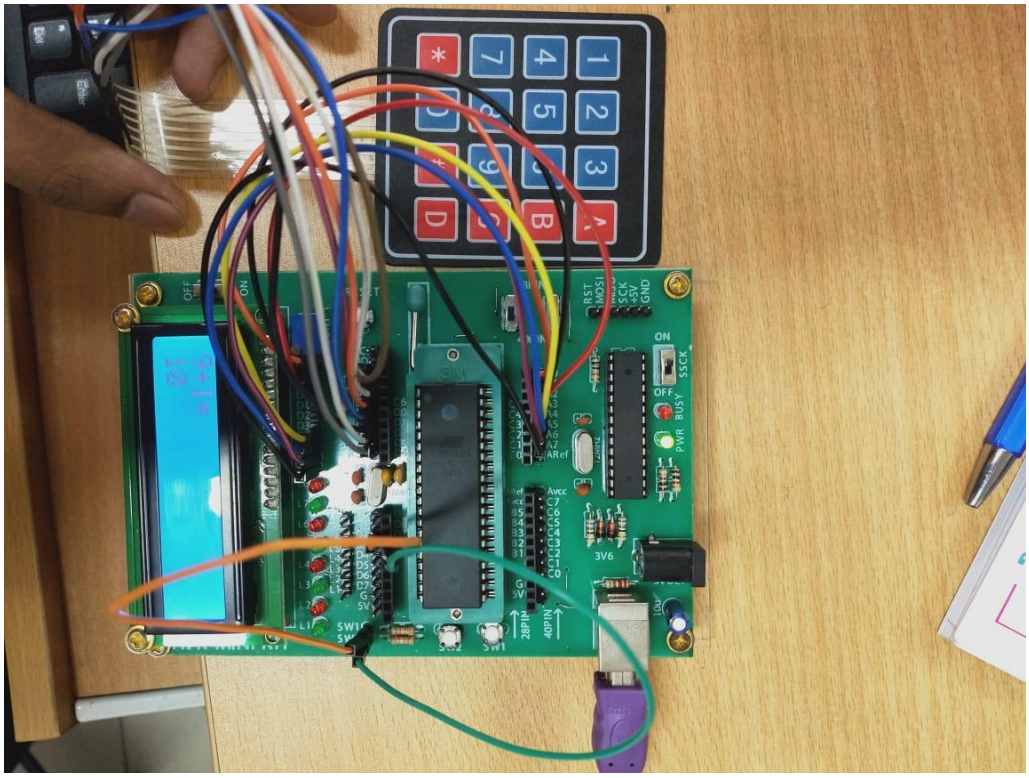
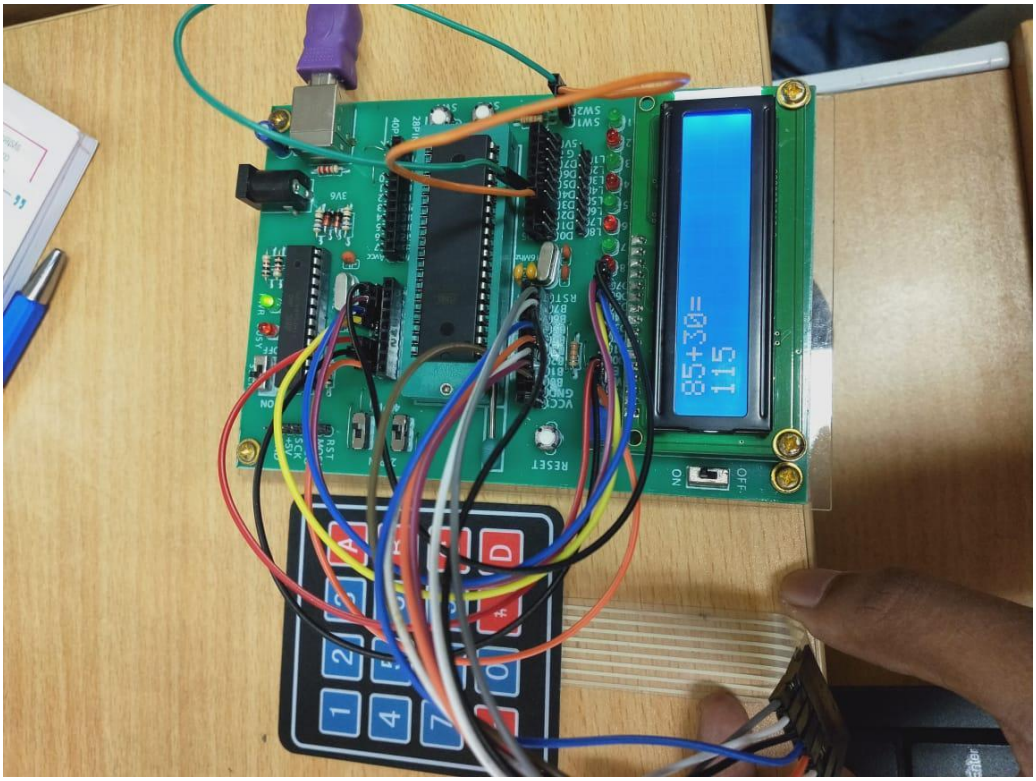


Figure:Exhibiting Addition Operation [85+30=115]

Hardware Simulations:





Discussion:

It is observed that we successfully showed the addition operation in our design calculator. We command our code in such a way that when we press the '+' button in the keypad then the flag 4 will be raised and the addition operation will occur.We modified our code in such a way so that we get not only single digit addition result but also multiple digits.

Subtraction(-):

Software Simulations:

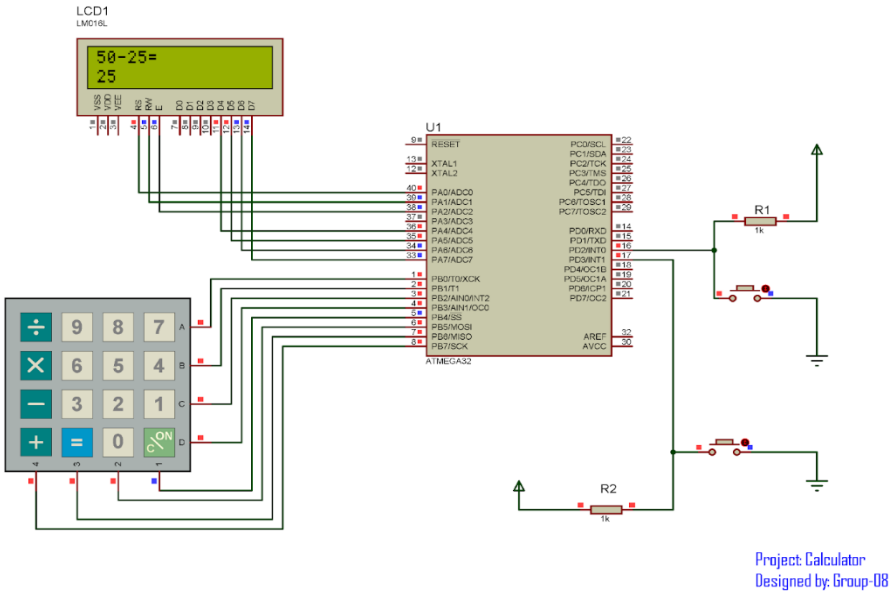


Figure:Exhibiting Subtraction Operation [50-30=25]

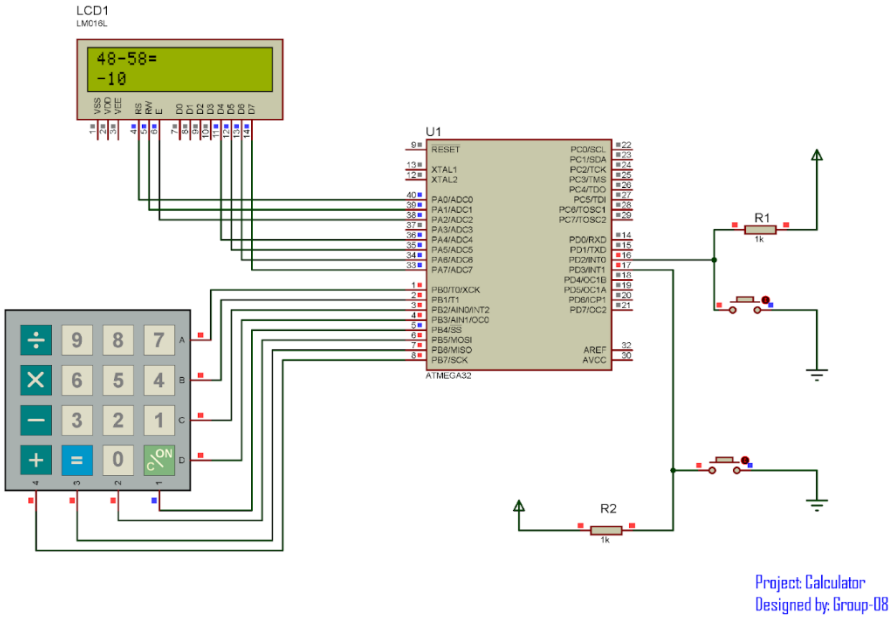
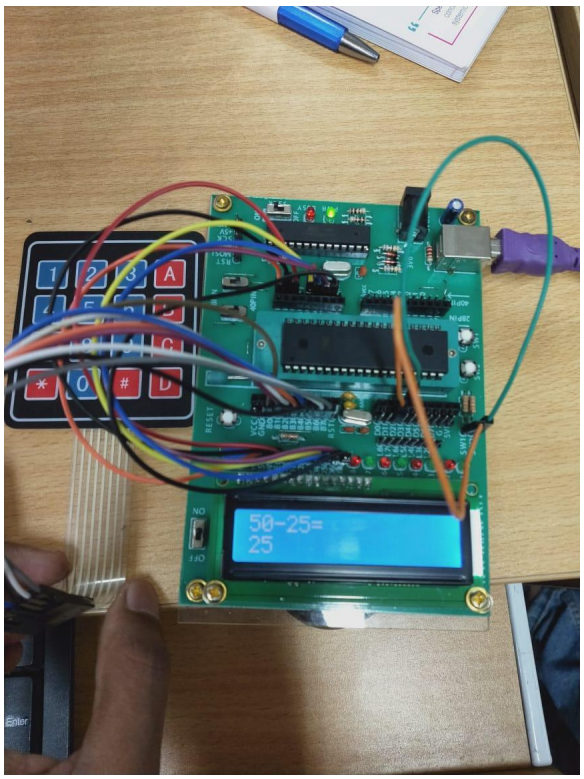
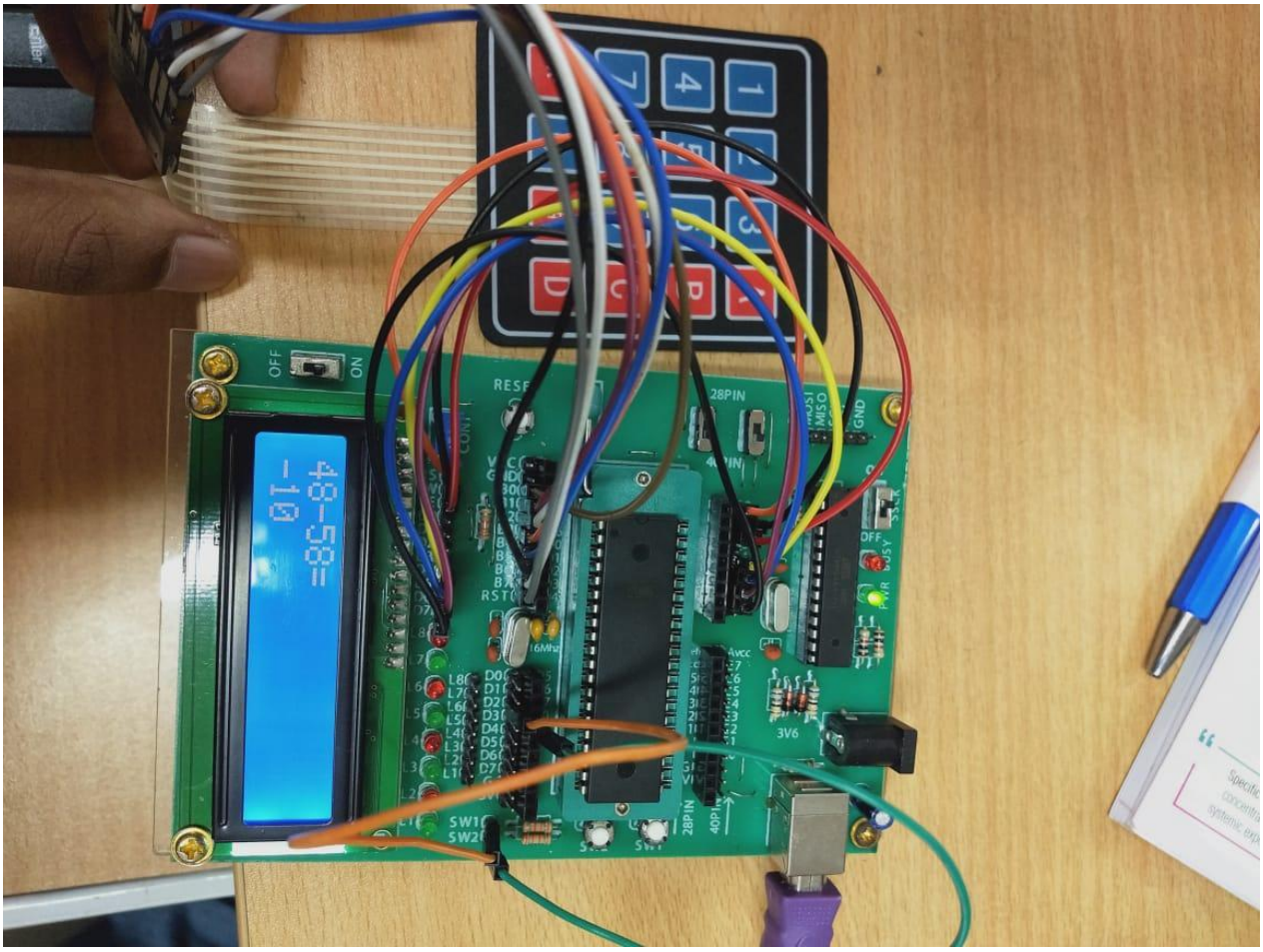


Figure:Exhibiting Subtraction Operation [48-58=10]

Hardware Simulations:





Discussion:

It is observed that we successfully showed the subtraction operation in our design calculator. We command our code in such a way that when we press the '-' button in the keypad then the flag 3 will be raised and the subtraction operation will occur.We modified our code in such a way so that we get not only single digit subtraction results but also multiple digits.

Multiplication(*):

Software Simulations:

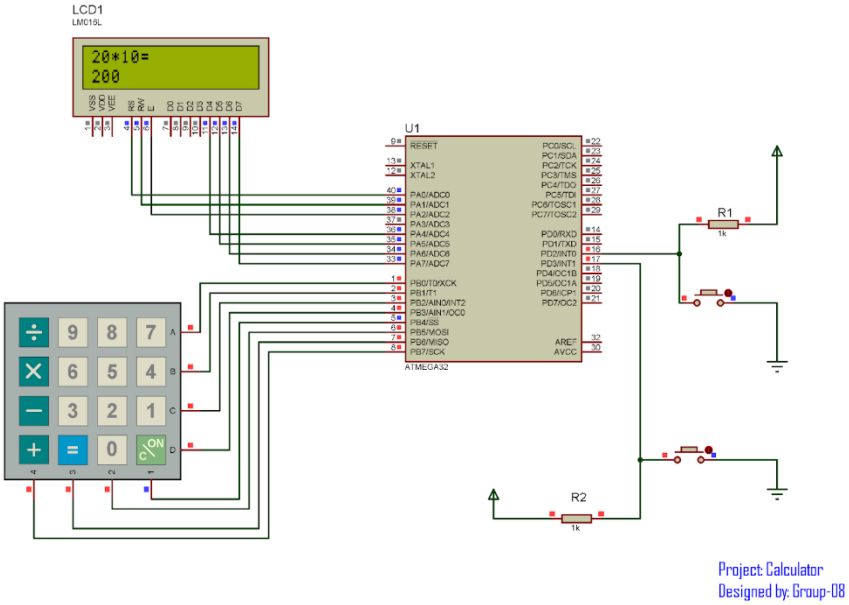


Figure:Exhibiting Multiplication Operation [20*10=200]

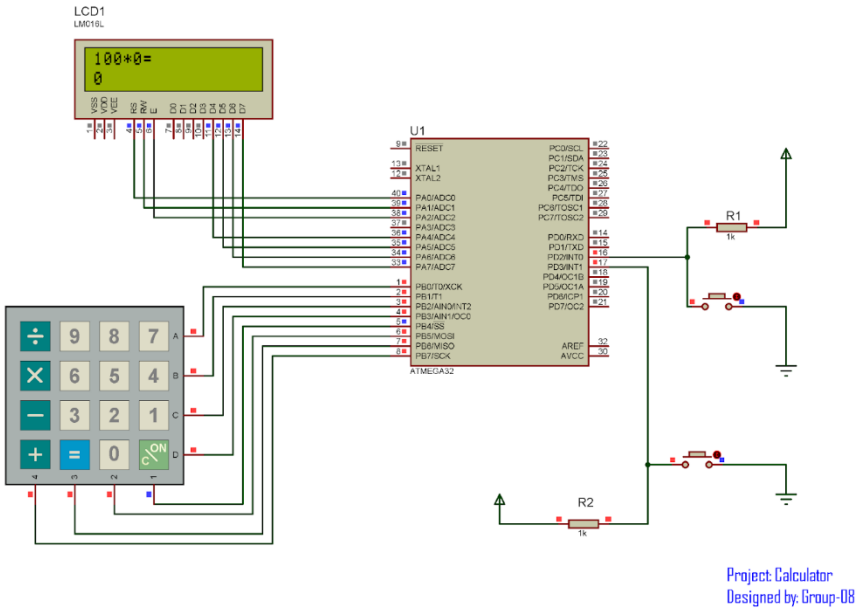
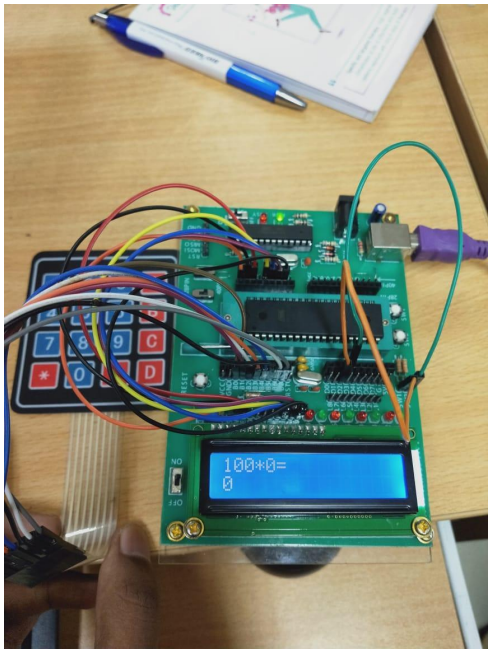
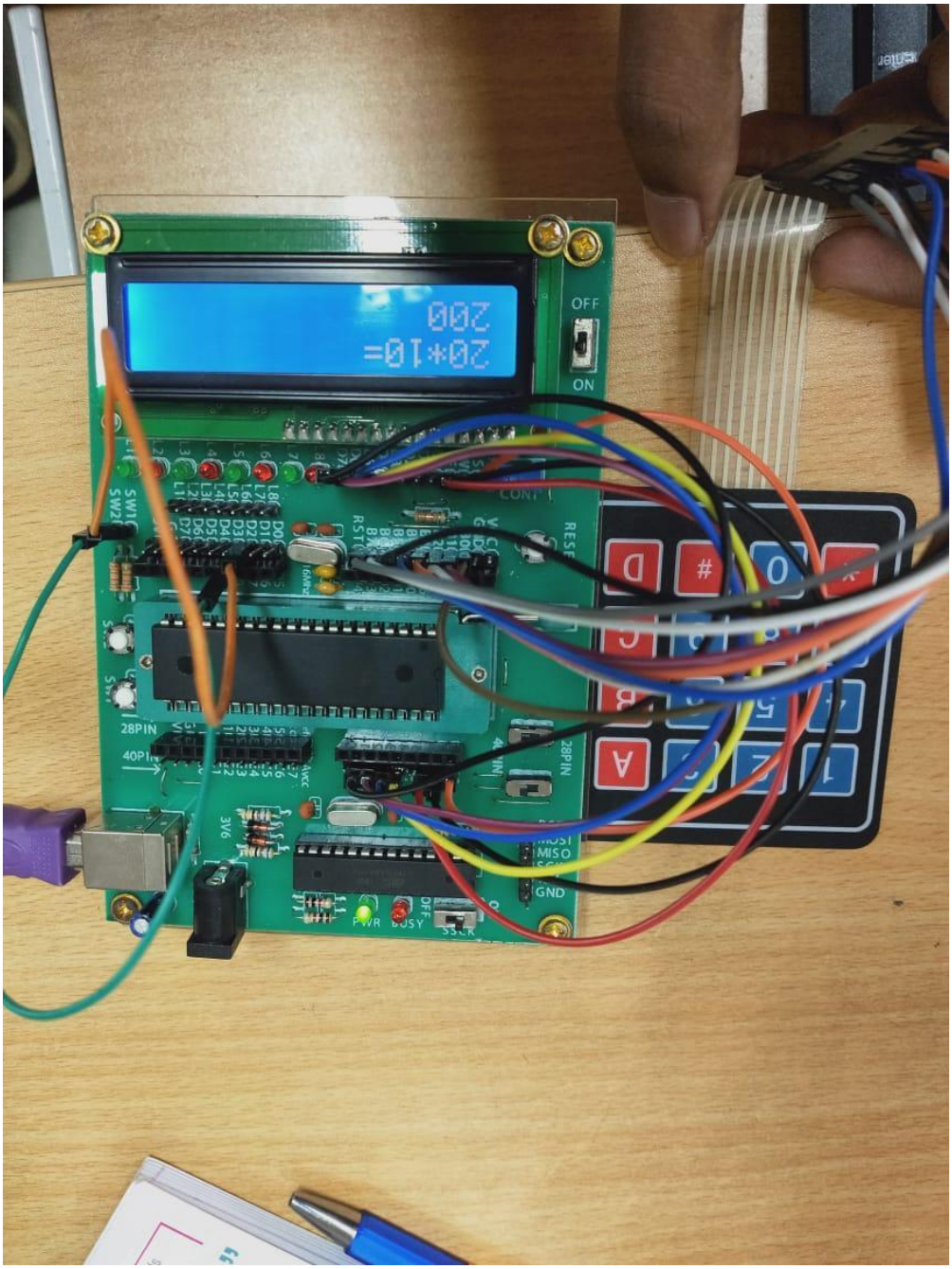


Figure:Exhibiting Multiplication Operation [100*0=0]

Hardware Simulations:





Discussion:

It is observed that we successfully showed the multiplication operation in our design calculator. We command our code in such a way that when we press the '*' button in the keypad then the flag 2 will be raised and the multiplication operation will occur.We modified our code in such a way so that we get not only single digit multiplication results but also multiple digits.

Division(/):

Software Simulations:

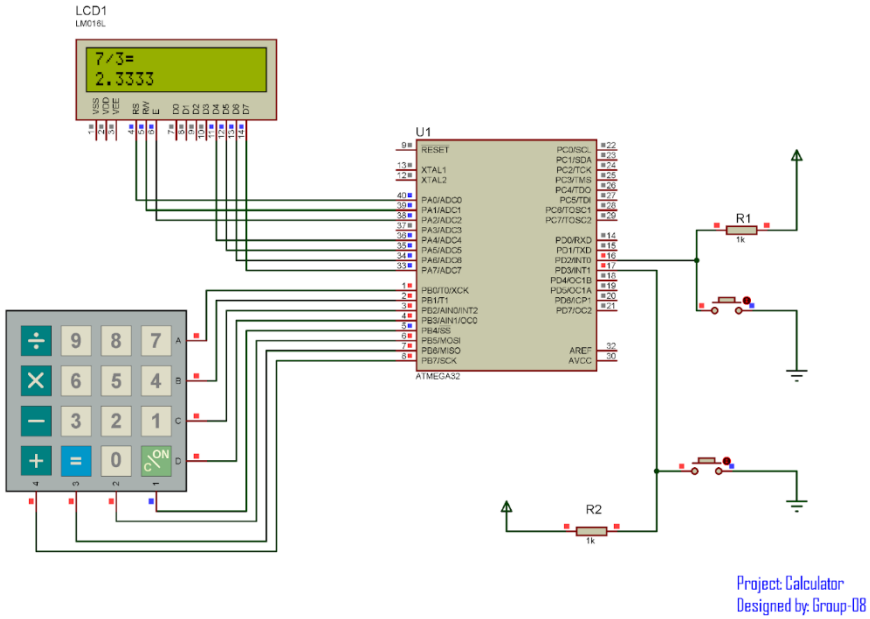


Figure:Exhibiting Division Operation [7/3=2.3333]

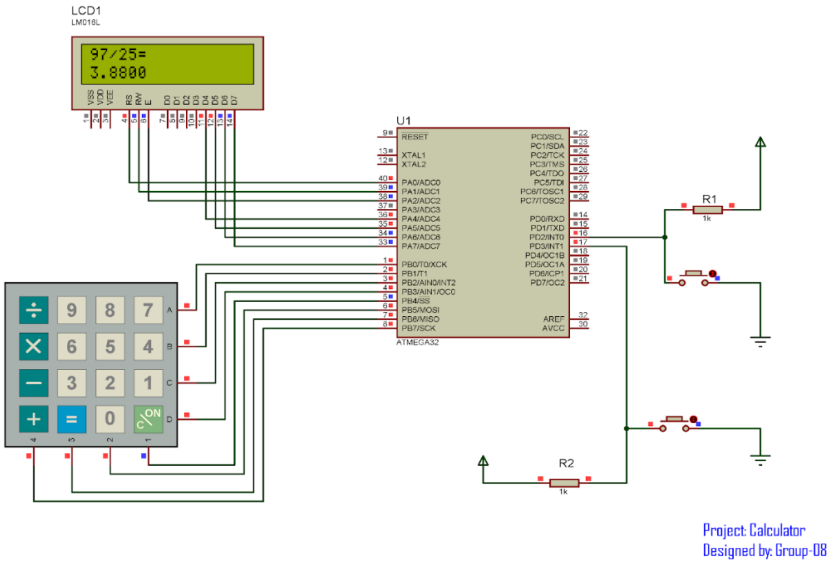
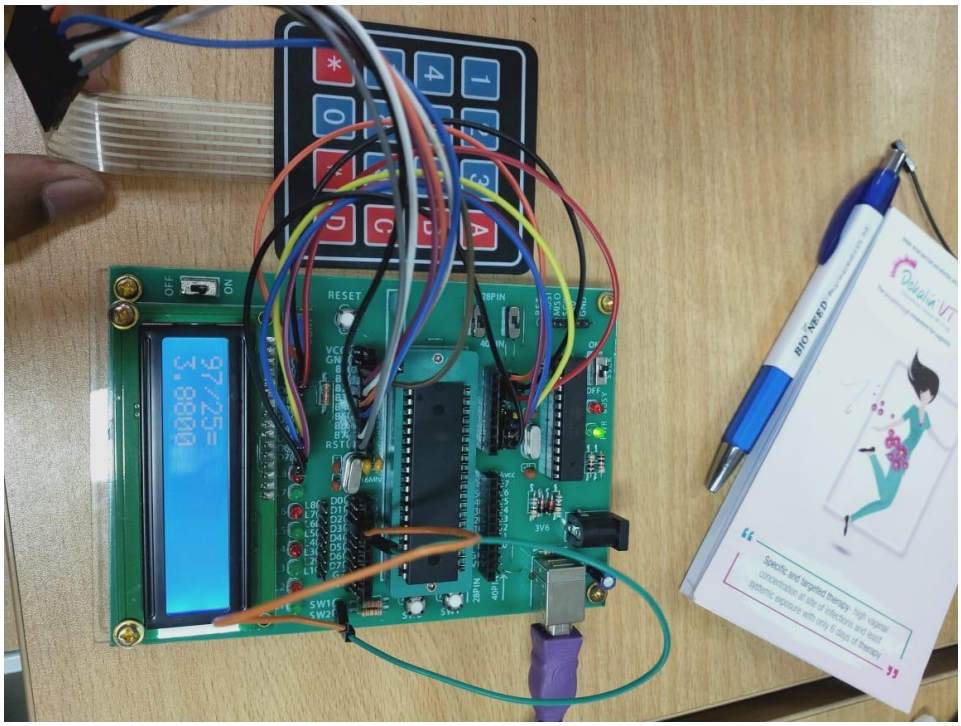
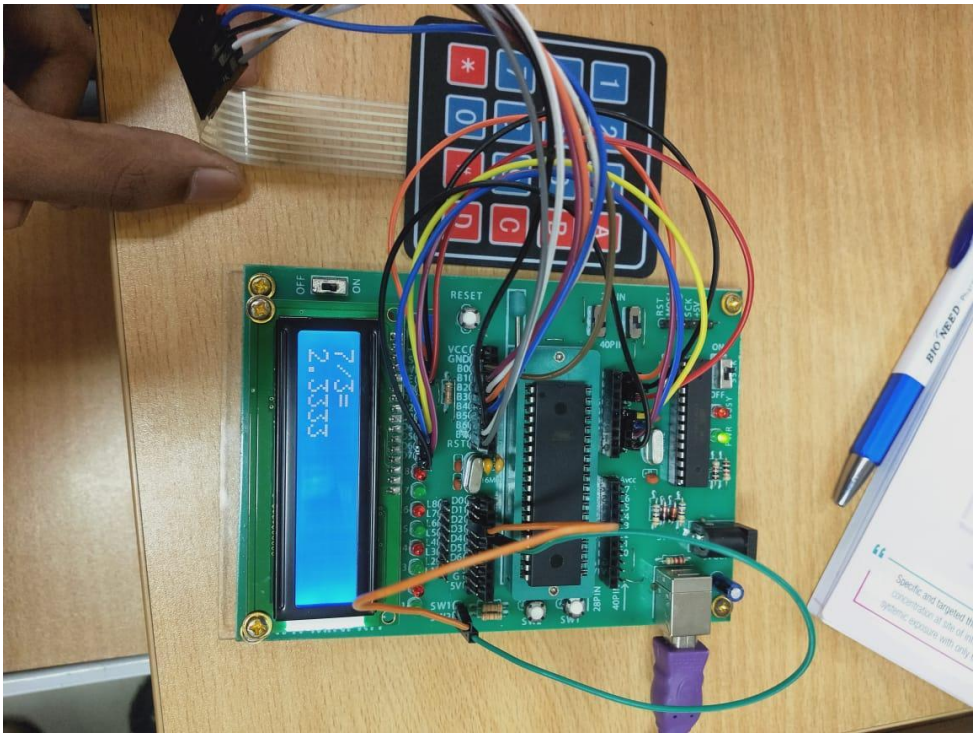


Figure:Exhibiting Division Operation [97/25=3.8800]

Hardware Simulations:





Discussion:

It is observed that we successfully showed the division operation in our design calculator. We command our code in such a way that when we press the '/' button in the keypad then the flag 1 will be raised and the division operation will occur.We modified our code in such a way so that we get not only single digit division results but also multiple digits.

Factorial(!):

Software Simulations:

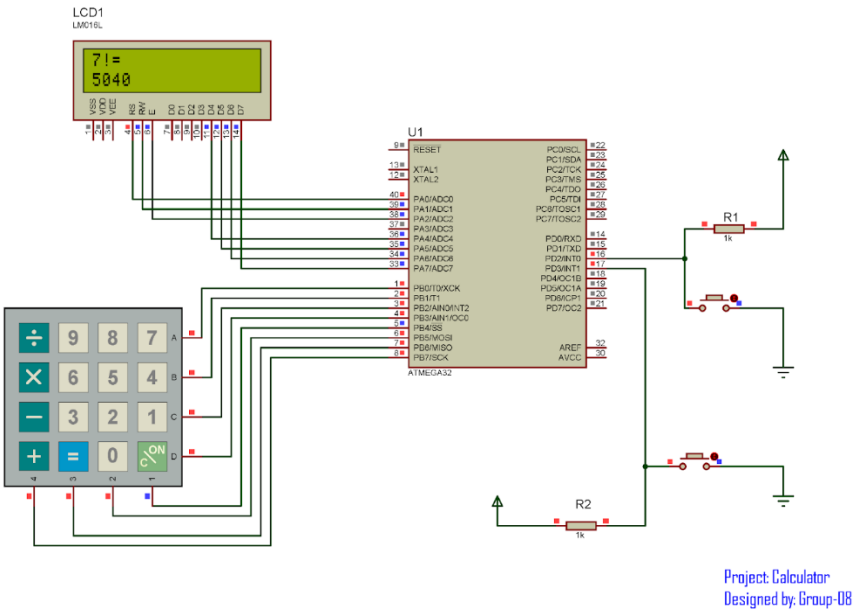


Figure:Exhibiting Factorial Operation [7!=5040]

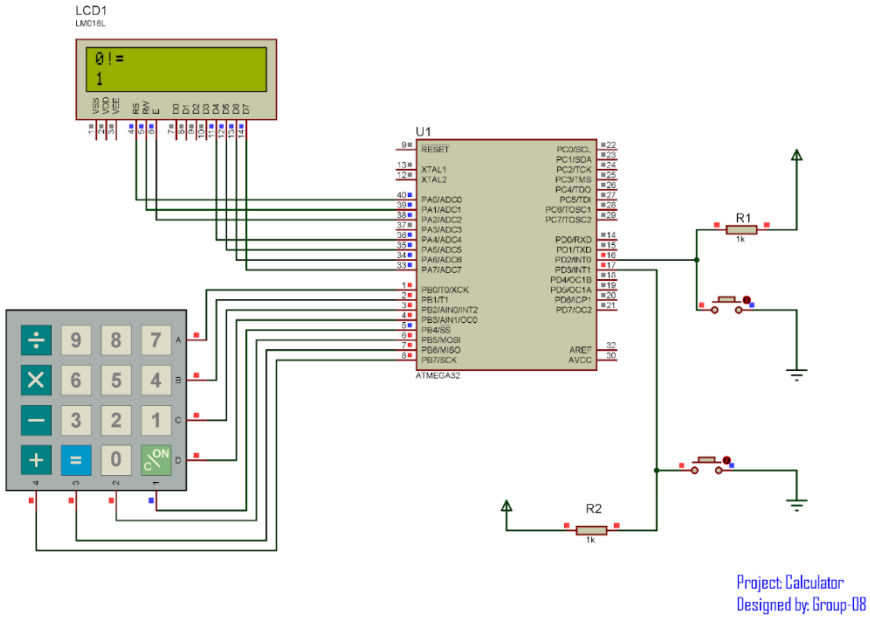
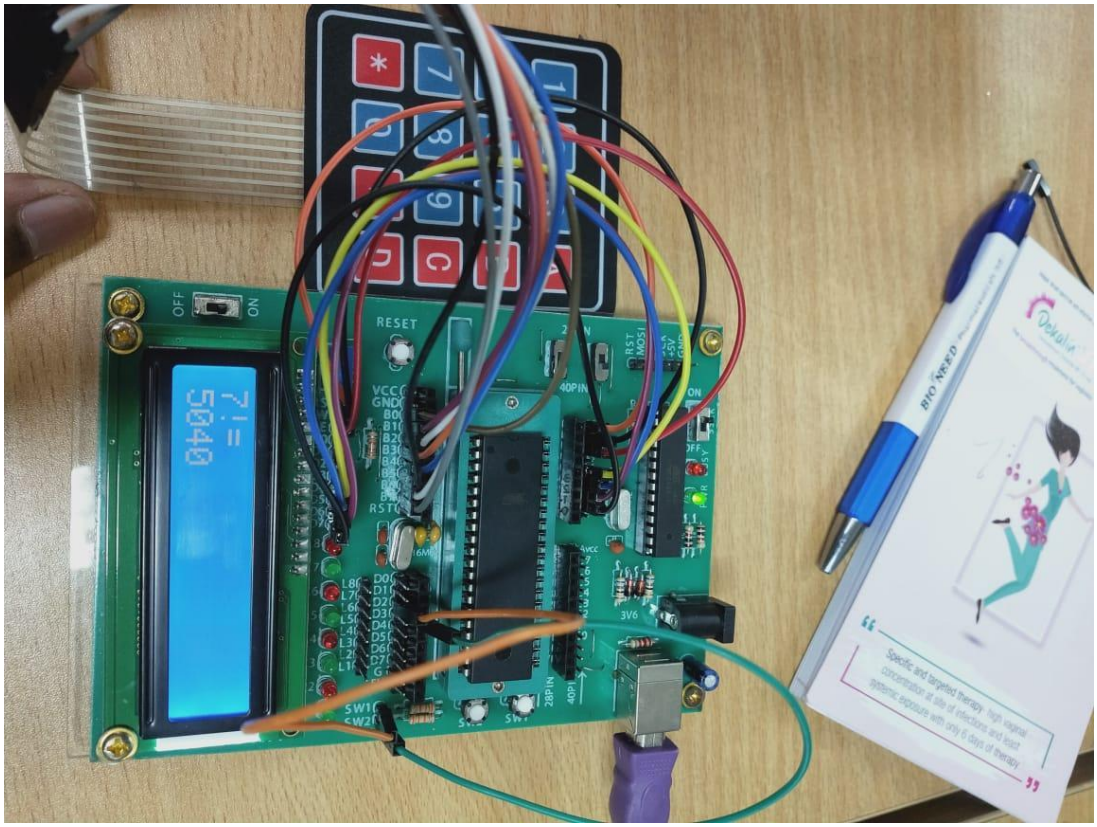
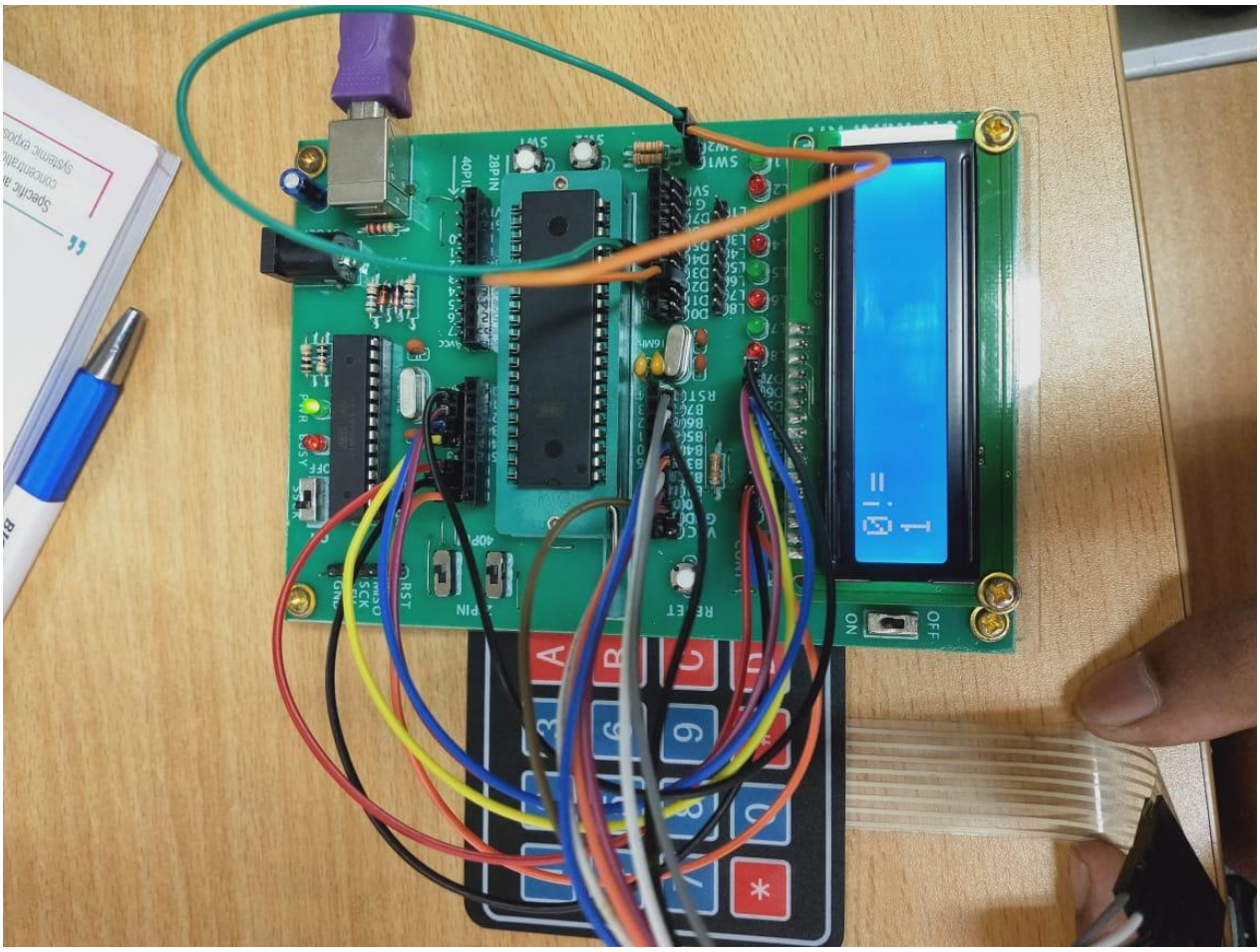


Figure:Exhibiting Factorial Operation [0!=1]

Hardware Simulations:





Discussion:

It is observed that we successfully showed the factorial operation in our design calculator. We command our code in such a way that when we press the '!' button in the keypad then the flag 5 will be raised and the factorial operation will occur.We modified our code in such a way so that we get not only single digit factorial results but also multiple digits.

Power(a^b):

Software Simulations:

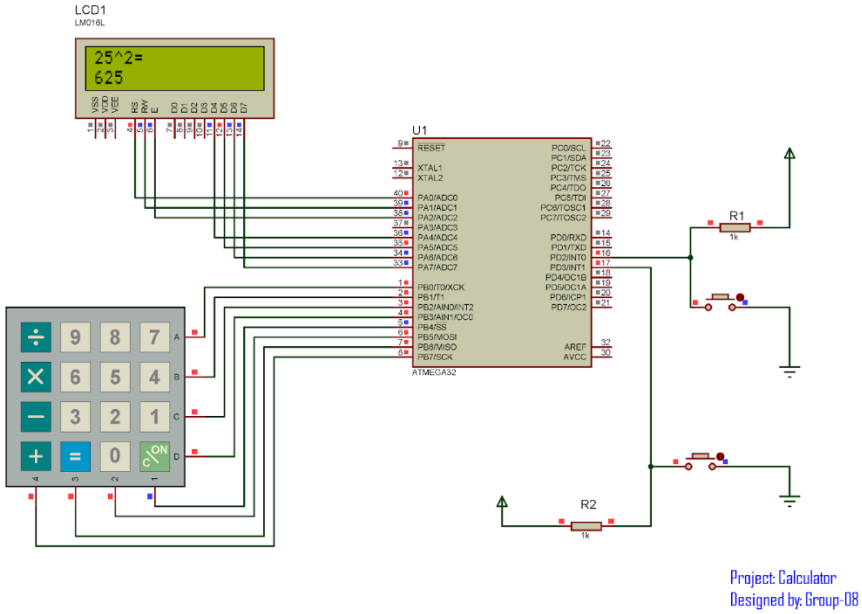


Figure:Exhibiting Power Operation [25^2=625]

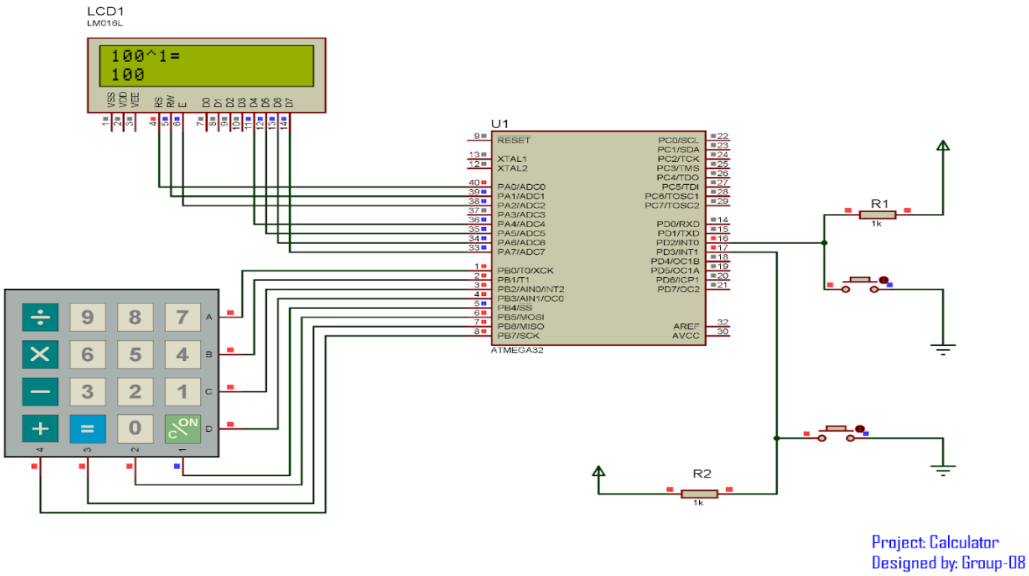
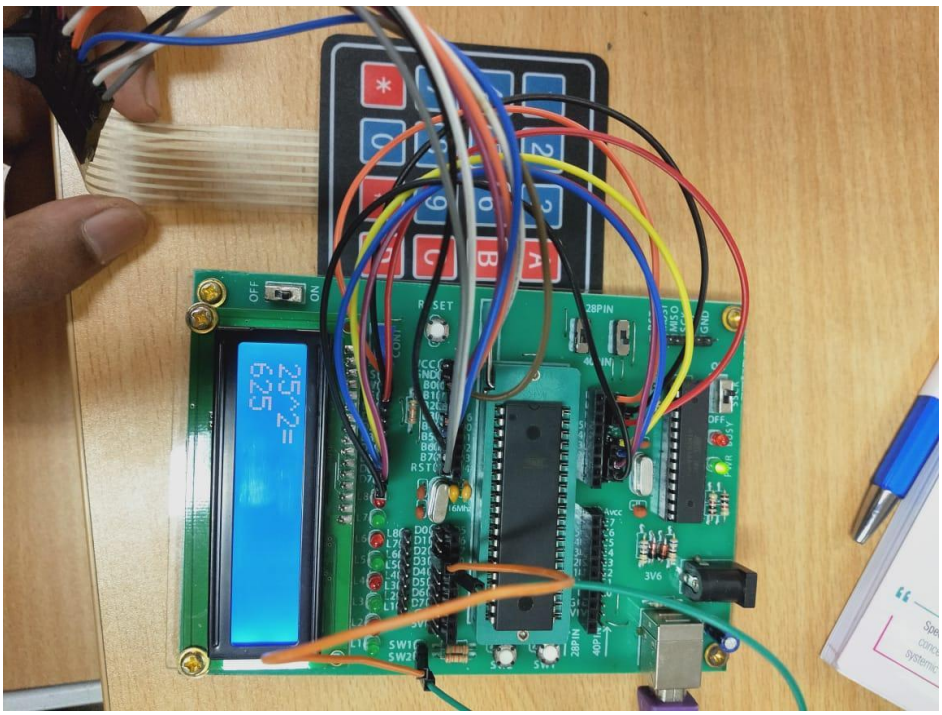
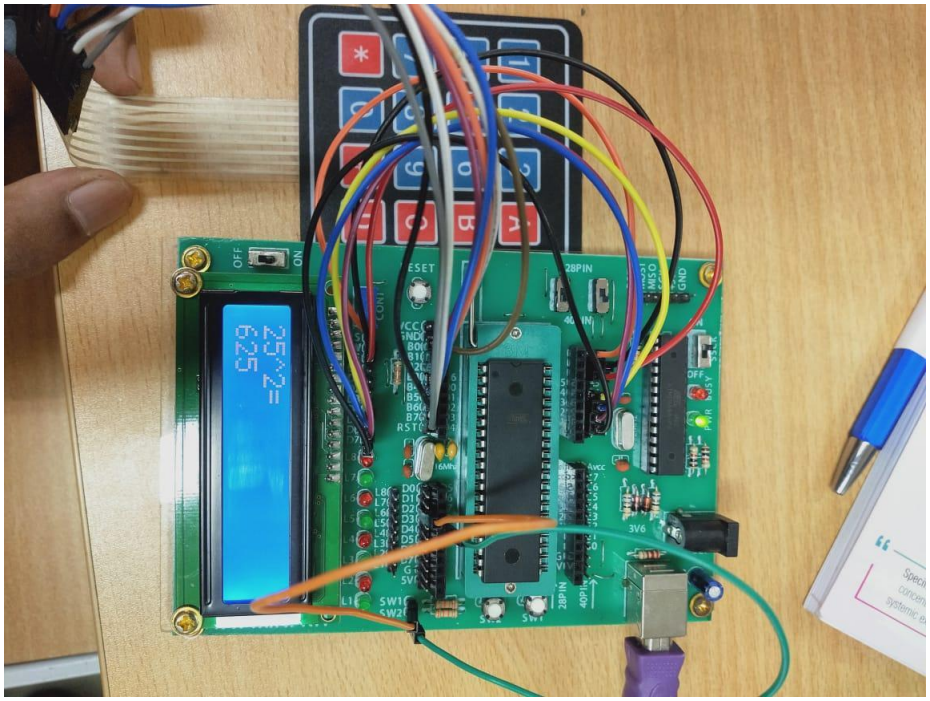


Figure:Exhibiting Power Operation [100^1=100]

Hardware Simulations:





Discussion:

It is observed that we successfully showed the power operation in our design calculator. We command our code in such a way that when we press the '^' button in the keypad then the flag 6 will be raised and the power operation will occur.We modified our code in such a way so that we get not only single digit power results but also multiple digits.

The results from our calculator for the same calculations were matched with the hardware simulation and our calculator has shown exactly the same result values as the other calculator. To conclude, we have successfully develop and design a simple calculator that can correctly do calculations using mathematical operators of + - * / ^ and !. also having a RESET option that clears the screen and all values.

X. Application

Here we have implemented a "Simple Calculator Project". This system is a necessity in the students' day to day life mainly and also for the common people for their simple calculations. So the Possible stakeholders for our project can be the students, general peoples or from calculator production enterprises such as Casio, Calc Pal, Sharp etc. Another option is that we could start a new company, dedicated to selling this product. Outside investors would also be willing to invest in such a company as the growth prospect of this system is high.

With the modernisation of scientific society in the upcoming days, it is essential that such a tool exists in everyone's hands. We want this project to be implemented everywhere as this is an advanced system that modernizes existing calculators.This project is solely for educational purposes, gaining knowledge and getting experience at dealing with obstacles while designing a system from scratch.

XI. Future Scope

Scientific (e.g. Casio fx-991EX) and simple calculator (e.g. Casio SI 300sc) are already available in the market and will be there in the future for consumers which are capable of doing all the mathematical calculations and more that our simple calculator does. These calculators have 8051 microcontrollers or other specially designed microprocessors inside, designed and programmed especially for computation tasks by their company, which are manufactured cost effectively. They requires less operating power (1.5 – 2V).If we want to build a simple calculator with Atmega32, it will become too bulky and not be cost effective nor have feasible design because Atmega32 microcontrollers are made for varieties of bigger tasks by consuming more power (5V minimum). They are not as portable as 8051 or other microcontrollers that can squeeze inside a calculator. So stepping into the Calculator market will be immensely competitive unless our product is more useful, capable and able to perform more functions by being more efficient, which can be achieved if we replace Atmega32 with a microcontroller that has the sole purpose of computations.

In the future, it is unlikely there will be any need for calculators because all smartphones or any mobile phones now have simple calculators programmed inside them. However scientific calculators are still in use by high school, college and university students where sometimes smartphones are not allowed during

examinations. Some calculators became cheaper and have more features than previous versions like plotting and programmable abilities. Even smartphones now have the ability to download a scientific calculator application that can work exactly like a physical scientific calculator. So in an age of advancing smartphone technologies, the need for simple calculators are slowly getting dominated, and turning into a lost technology.

We tried to implement the project to the best of our abilities and looking back at it, we consider the project as a success. We were limited by the use of ATmega32 as the microcontroller in the system. Using modern microcontrollers such as a NodeMCU or Raspberry Pi in the project would allow us to implement even more advanced features such as AI assisted home security and facial recognition systems. For now, these are out of the scope of the project and we left it as future improvements.

XII. Reference

1.https://www.circuitstoday.com/matrix-keypad-interfacing-proteus

2.https://www.youtube.com/watch?v=wroeqfLr_js

3.https://atmega32-avr.com/simple-calculator-using-avr-microcontroller/

4.https://electronics.stackexchange.com/questions/555415/making-a-simple-calculator-with-atmega32

5.https://en.wikipedia.org/wiki/C_data_types?fbclid=IwAR20ODkV1-78ul31KyFd1JgKBiJNhGA-tPzV8Houta9KETVsoQFbRlQluzM#:~:text=long%20int,2%2C147%2C483%2C647%2C%20%2B2%2C147%2C483%2C647%5D%20range

6.https://www.geeksforgeeks.org/use-of-flag-in-programming/

7.https://www.definitions.net/definition/interrupt+flag

8.https://web.ics.purdue.edu/~jricha14/Timer_Stuff/TIFR.htm

9.https://www.youtube.com/watch?v=dbjiVOsGUFA&ab_channel=ElectronicsDeveloper

10. https://www.youtube.com/watch?v=wroeqfLr_js

11. https://www.youtube.com/watch?v=dbjiVOsGUFA

12. https://www.youtube.com/watch?v=kFkvFm6YzbM

13. https://www.youtube.com/watch?v=aRqiwwKb5jY

14. https://www.zenflowchart.com/

XIII. Appendices

A. Source Code:

Design Approach-01

Code:

```
#include <mega32.h>
#include <delay.h>
#include <alcd.h>
#include <stdio.h>
#include <stdlib.h>
#define keypad_ddr DDRB
#define keypad_port PORTB
#define input_data PINB
void keypad (void);
int numGet (void);
int a=0, b=0,key=20, flag=0,flag1=0,num,p, fact;
unsigned char c[16], i;
interrupt [EXT_INT0] void ext_int0_isr(void)
{
  lcd_putchar('!');
  flag=7;
   delay_ms(300);
    }
 interrupt [EXT_INT1] void ext_int1_isr(void)
{
 lcd_putchar('^');
 flag=6;
  delay_ms(300);
   }
void main (void)
{
 While (1) {
    keypad_ddr=0xF0;
```

```
    lcd_init(16);
     a=numGet();
     delay_ms(50);
     flag1=flag;
     b=numGet();
     delay_ms(50);
    GICR|=(1<<INT1) | (1<<INT0) | (0<<INT2);
    MCUCR=(1<<ISC11) | (0<<ISC10) | (1<<ISC01) | (0<<ISC00);
    MCUCSR=(0<<ISC2);
    GIFR=(1<<INTF1) | (1<<INTF0) | (0<<INTF2);
#asm("sei")
    while(flag==5)
    {
     lcd_gotoxy(0,1);
     if(flag1==4)
     sprintf(c,"%d",a+b);
     else if (flag1==3)
      sprintf(c,"%d",a-b);
      else if (flag1==2)
      sprintf(c,"%d",a*b);
      else if (flag1==1)
      ftoa((float)a/b,4,c);

      else if (flag1==6){
         p=a;
          for(i=1;i<b;i++)
          {
            p*=a;
          }
          sprintf(c,"%d",p); }
```

```
        else if (flag1==7){
                    sprintf(c,"%d",fact);}
      lcd_puts(c);
      keypad_port=0b11101111;
      if(input_data.3==0) {lcd_clear(); flag = 0;
      }
  }
  }
      void keypad(void)
{
key=20;
keypad_port=0xFF;

keypad_port=0b11101111;
delay_ms(2);
if(input_data.0==0) key=7;
if(input_data.1==0) key=4;
if(input_data.2==0) key=1;
if(input_data.3==0) {lcd_clear(); flag = 0;}
keypad_port=0xFF;
keypad_port=0b11011111;
delay_ms(2);
if(input_data.0==0) key=8;
if(input_data.1==0) key=5;
if(input_data.2==0) key=2;
if(input_data.3==0) key=0;
keypad_port=0xFF;
keypad_port=0b10111111;
delay_ms(2);
if(input_data.0==0) key=9;
if(input_data.1==0) key=6;
if(input_data.2==0) key=3;
```

```
if(input_data.3==0) { lcd_putchar('='); flag=5; delay_ms(300); }
keypad_port=0xFF;
keypad_port=0b01111111;
delay_ms(2);
if(input_data.0==0) { lcd_putchar('/'); flag=1; delay_ms(300); }
if(input_data.1==0) { lcd_putchar('*'); flag=2; delay_ms(300); }
if(input_data.2==0) { lcd_putchar('-'); flag=3; delay_ms(300); }
if(input_data.3==0) { lcd_putchar('+'); flag=4; delay_ms(300); }
keypad_port=0xFF;}
int numGet(void)
{
   flag=0;
   num=0;
   while(flag<1){
      keypad();
      if(key!=20){
         sprintf(c,"%d",key);
         lcd_puts(c);
         delay_ms(300);
         num+=key;
         num*=10;
      }} return num/10;}
```

Design Approach-02

Code:

```
/* Project: to design a Calculator Code */
// EEE366 Project
//Group-08

#include <mega32.h>
#include <alcd.h>
#include <delay.h>
char x,I,j,press = 0;
long int n,a,b,q,r;
void main(void)
   {
   DDRB=0xFF;
   DDRC=0xF0;
   lcd_init(16);
   while(1)
      {
      PORTC=0b00001111;
      PORTB=0b00111110;
      lcd_gotoxy(x,0);
   if(PINC.0==0)
      {
      lcd_clear();
      x=0;
      lcd_putsf("0");
      n=0; a=0; b=0; q=0; r=0; press=0;
      }
```

```
if(PINC.1==0){
    lcd_putsf("-");
    press=6;
    delay_ms(500);
    x++;
    }
if(PINC.2==0)
    {
    lcd_putsf("!");
    delay_ms(500);
    if(n==0){
    n=1;}
else{
for(i=n-1;i>0;i--)
    {n*=I;}
}
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);
}
if(PINC.3==0)
    {
    a=n;
    n=0;
    press=press+5;
    lcd_putsf("^");
    delay_ms(500);
```

```
    x++; }
 PORTB=0b00111101;
 lcd_gotoxy(x,0);
if(PINC.0==0)
    {
    lcd_putsf("7");
    n=(n*10)+7;
    delay_ms(500);
    x++;
    }
 if(PINC.1==0)
    {
    lcd_putsf("4");
    n=(n*10)+4;
    delay_ms(500);
    x++;
    }
 if(PINC.2==0)
    {
    lcd_putsf("1");
    n=(n*10)+1;
    delay_ms(500);
    x++;
    }
 if(PINC.3==0)
    {
    lcd_putsf("0");
```

```
    n=(n*10)+0;
    delay_ms(500);
    x++; }
 PORTB=0b00111011;
 lcd_gotoxy(x,0);
if(PINC.0==0)
    {
    lcd_putsf("8");
    n=(n*10)+8;
    delay_ms(500);
    x++; }
 if(PINC.1==0)
    {
    lcd_putsf("5");
    n=(n*10)+5;
    delay_ms(500);
    x++;}
 if(PINC.2==0)
    {
    lcd_putsf("2");
    n=(n*10)+2;
    delay_ms(500);
    x++; }
 PORTB=0b00110111;
 lcd_gotoxy(x,0);
 if(PINC.0==0)
    {
```

```
    lcd_putsf("9");
    n=(n*10)+9;
    delay_ms(500);
    x++; }
if(PINC.1==0)
    {
    lcd_putsf("6");
    n=(n*10)+6;
    delay_ms(500);
    x++; }
 if(PINC.2==0)
    {
    lcd_putsf("3");
    n=(n*10)+3;
    delay_ms(500);
    x++; }
   if(PINC.3==0)
    {
    if(n<-1999999999 || n>1999999999){
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("MATH ERROR");}
else{
if(n>=0 || n<=0)
    {
    lcd_clear();
    lcd_gotoxy(0,1);
```

```
    lcd_printf("=%ld",n); }
if(press==1)
    {
    a+=n;  n=a;

    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);
    }
if(press==2)
    {
    a-=n;
    n=a;
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);   }
if(press==3)
    {
    a*=n;
    n=a;
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);  }
 if(press==4)
    {
    if(n==0){
    lcd_clear();
```

```
    lcd_gotoxy(0,1);
    lcd_printf("MATH ERROR"); }
    else{
    q=a/n;
    r=a%n;
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=Q:%ld R:%ld",q,r);
    n=q;
    } }
    if(press==5){
    if(n==0){
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=1");
    }
    else{
    b=n;
    n=a;
for(j=1;j<b;j++){
n*=a;
}
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);
    }
}
```

```
  if(press==6){
     n=n*-1;
     lcd_clear();
     lcd_gotoxy(0,1);
     lcd_printf("=%ld",n);}}
     x=0; press=0;
     delay_ms(500); }
  PORTB=0b00101111;
  lcd_gotoxy(x,0);
  if(PINC.0==0){
     x=0;
     if(press>0){
     a*=n;
     }
  else{
 a=n;}
  press=3;
  n=0;
  lcd_clear();
  lcd_putsf("*");
  delay_ms(500);
  x++; }
if(PINC.1==0){
  x=0;
  if(press>0){
  a-=n;}
  else{
```

```
a=n;  }
press=2;
n=0;
lcd_clear();
lcd_putsf("-");
delay_ms(500);
x++; }
if(PINC.3==0){
x=0;
if(press>0){
a+=n; }
else{
a=n;}
press=1;
n=0;
lcd_clear();
lcd_putsf("+");
delay_ms(500);
x++;}
PORTB=0b00011111;
lcd_gotoxy(x,0);
if(PINC.0==0){
x=0;
if(press>0){
a/=n }
else{
a=n;}
```

```
press=4;
n=0;
lcd_clear();
lcd_putsf("/");
delay_ms(500);
x++;} }}
```

Design Approach-03

Code:

```
/* Project: to design a Calculator Code */
// EEE366 Project
//Group-08

#include <mega32.h>
#include <alcd.h>
#include <delay.h>
char x,i,j,press = 0;
long int n,a,b,q,r;
void main(void)
  {
  DDRB=0xFF;
  DDRC=0xF0;
  lcd_init(16);
  while(1)
    {
    PORTC=0b00001111;
    PORTB=0b00111110;
    lcd_gotoxy(x,0);
  if(PINC.0==0)
```

```
    {
    lcd_clear();
    x=0;
    lcd_putsf("0");
    n=0; a=0; b=0; q=0; r=0; press=0;
    }
if(PINC.1==0){
    lcd_putsf("-");
    press=6;
    delay_ms(500);
    x++;
    }
if(PINC.2==0)
    {
    lcd_putsf("!");
    delay_ms(500);
    if(n==0){
    n=1;}
else{
for(i=n-1;i>0;i--)
    {n*=i;}
}
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);
}
if(PINC.3==0)
    {
    a=n;
```

```
    n=0;
    press=press+5;
    lcd_putsf("^");
    delay_ms(500);
    x++; }
 PORTB=0b00111101;
 lcd_gotoxy(x,0);
if(PINC.0==0)
    {
    lcd_putsf("7");
    n=(n*10)+7;
    delay_ms(500);
    x++;
    }
 if(PINC.1==0)
    {
    lcd_putsf("4");
    n=(n*10)+4;
    delay_ms(500);
    x++;
    }
 if(PINC.2==0)
    {
    lcd_putsf("1");
    n=(n*10)+1;
    delay_ms(500);
    x++;
    }
 if(PINC.3==0)
```

```
    {
    lcd_putsf("0");
    n=(n*10)+0;
    delay_ms(500);
    x++; }
 PORTB=0b00111011;
 lcd_gotoxy(x,0);
if(PINC.0==0)
    {
    lcd_putsf("8");
    n=(n*10)+8;
    delay_ms(500);
    x++; }
 if(PINC.1==0)
    {
    lcd_putsf("5");
    n=(n*10)+5;
    delay_ms(500);
    x++;}
 if(PINC.2==0)
    {
    lcd_putsf("2");
    n=(n*10)+2;
    delay_ms(500);
    x++; }
 PORTB=0b00110111;
 lcd_gotoxy(x,0);
 if(PINC.0==0)
    {
```

```
    lcd_putsf("9");
    n=(n*10)+9;
    delay_ms(500);
    x++; }
if(PINC.1==0)
    {
    lcd_putsf("6");
    n=(n*10)+6;
    delay_ms(500);
    x++; }
 if(PINC.2==0)
    {
    lcd_putsf("3");
    n=(n*10)+3;
    delay_ms(500);
    x++; }
   if(PINC.3==0)
    {
    if(n<-1999999999 || n>1999999999){
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("MATH ERROR");}
else{
if(n>=0 || n<=0)
    {
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n); }
if(press==1)
```

```
    {
    a+=n;  n=a;

    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);
    }
if(press==2)
    {
    a-=n;
    n=a;
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);   }
if(press==3)
    {
    a*=n;
    n=a;
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);  }
 if(press==4)
    {
    if(n==0){
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("MATH ERROR"); }
    else{
    q=a/n;
```

```
    r=a%n;
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=Q:%ld R:%ld",q,r);
    n=q;
    } }
    if(press==5){
    if(n==0){
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=1");
    }
    else{
    b=n;
    n=a;
for(j=1;j<b;j++){
n*=a;
}
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);
    }
}
if(press==6){
    n=n*-1;
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_printf("=%ld",n);}}
    x=0; press=0;
```

```
      delay_ms(500); }
   PORTB=0b00101111;
   lcd_gotoxy(x,0);
   if(PINC.0==0){
      x=0;
      if(press>0){
      a*=n;
      }
   else{
 a=n;}
   press=3;
   n=0;
   lcd_clear();
   lcd_putsf("*");
   delay_ms(500);
   x++; }
if(PINC.1==0){
   x=0;
   if(press>0){
   a-=n;}
   else{
   a=n;  }
   press=2;
   n=0;
   lcd_clear();
   lcd_putsf("-");
   delay_ms(500);
   x++; }
   if(PINC.3==0){
```

```
x=0;
if(press>0){
a+=n; }
else{
a=n;}
press=1;
n=0;
lcd_clear();
lcd_putsf("+");
delay_ms(500);
x++;}
PORTB=0b00011111;
lcd_gotoxy(x,0);
if(PINC.0==0){
x=0;
if(press>0){
a/=n }
else{
a=n;}
press=4;
n=0;
lcd_clear();
lcd_putsf("/");
delay_ms(500);
x++;} }}
```

B. Drive Link:

https://drive.google.com/drive/folders/19sMriTKCXeomBg5a8JWnrSgv7sKtDJzo?usp=sharing