
Electrical and Electronics Engineering

EE469 Embedded Systems

# Membrane Keypad Interface Project

*Author:*
Bengisu YÜCEL
Seyit Semih YİĞİTARSLAN
Student ID: 201726081
201726083

*Supervisors:*
Erdem AKAGÜNDÜZ

June 21, 2021

# Contents

**Abstract**

Within the scope of the embedded systems course, we use Texas Instrumentals EK-TM4C123GXL ARM® Cortex®-M4F Based MCU TM4C123G LaunchPad™ Evaluation Kit and Membran Keypad. We integrated the keypad with the EK-TM4C123GXL card. In the software part, studied on the Keil program in C language after that the UART protocol provides us to communication with Teraterm program.

# 1 Github Source Code

For the Membrane Keypad Interface project in Ek-TM4C123GXL microcontroller card source code is at the link below.

https://github.com/ssytrying/membraneKeypadInterface

# 2 Introduction

The EK-TM4C123GXL is a cortex M4 based microcontroller. The EK-TM4C123GXL includes programmable user buttons and an RGB LED for custom applications. For this device, we used a software development environment for our microcontroller card. The KEIL MDK is used for that. MDK includes the µVision IDE and debugger, Arm C/C++ compiler, and essential middleware components. So in the communication part with keypad and card, UART is a hardware communication protocol that uses asynchronous serial communication with configurable speed. Asynchronous means there is no clock signal to synchronize the output bits from the transmitting device going to the receiving end. Our device is a 4x4 Membrand Keypad for communication. The **fig.1** shows the our devices and connection. There is also shown a LCD but we did not use in our project.

# 3 Stages

## 3.1 Keypad Working Princible and Initialization Pins

The keypad consists of 8 pins as R5-R2-R3-R4, C4-C5-C6-C7, these are R1, R2, R3, R4and Column first to fourth, when each key is pressed, the pins connected to the Rowand Column are short-circuited respectively. For example, when the 1 key is pressed, R5 and C4 are short-circuited, considering this situation, the necessary code will be written on Keil. It works with scan algorithm, according to that, read and test the status of all column wires (check the value of C4, C5, C6, and C7) If column Cnis low, it is shorted to row R1 else column Cn is high, Cn is not shorted to any row wire and remains pulled high then repeat steps one and two, but with different row wire driven low (and others high). A keypress detects if a column line is detected in the low state, the key position is the intersection of that column and the row being driven low. We will connect PC4, PC5, PC6, PC7, PE5, PE2, PE3, andPE4 (8 pins) on the 4x4 Membrane Keypad shown in **fig.3**, On the software part, we write the $MatrixKeypadInit$ function

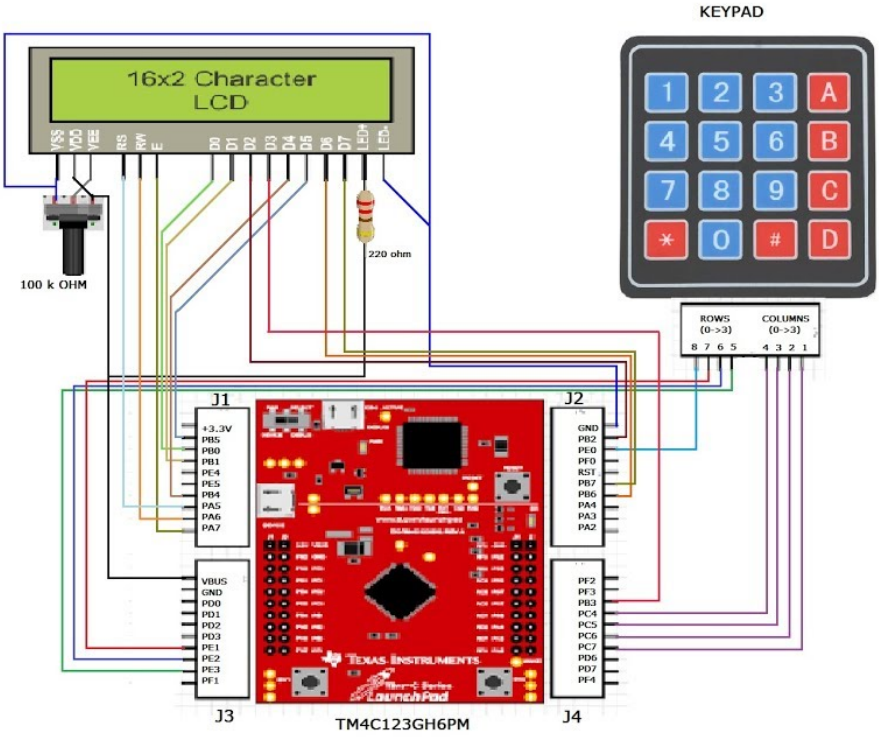


Figure 1: Texas Card and Keypad

```
void MatrixKeypad_Init(void){
  DisableInterrupts();
  // activate port C and port E
  SYSCTL_RCGCGPIO_R |= 0x14;
  GPIO_PORTC_DIR_R &= ~0xF0;
  GPIO_PORTC_AFSEL_R &= ~0xF0;
  GPIO_PORTC_DEN_R |= 0xF0;
  GPIO_PORTC_PCTL_R &= ~0xFFFF0000;
  GPIO_PORTC_AMSEL_R &= ~0xF0;
  GPIO_PORTC_IS_R &= ~0xF0;
  GPIO_PORTC_IBE_R &= ~0xF0;
  GPIO_PORTC_IEV_R |= 0xF0;
  GPIO_PORTC_ICR_R = 0xF0;
  GPIO_PORTC_IM_R |= 0xF0;

  NVIC_PRI0_R = (NVIC_PRI0_R&0xFF00FFFF)|0x00400000;
  GPIO_PORTE_DIR_R &= ~0x3C;
  GPIO_PORTE_AFSEL_R &= ~0x3C;
  GPIO_PORTE_DEN_R |= 0x3C;

  GPIO_PORTE_PCTL_R = (GPIO_PORTE_PCTL_R&0xFFF0FFFF)+0x00000000;
  GPIO_PORTE_AMSEL_R &= ~0x3C;
  GPIO_PORTE_IS_R &= ~0x3C;
  GPIO_PORTE_IBE_R &= ~0x3C;
  GPIO_PORTE_IEV_R |= 0x3C;
  GPIO_PORTE_ICR_R = 0x3C;
  GPIO_PORTE_IM_R |= 0x3C;

  NVIC_PRI1_R = (NVIC_PRI1_R&0xFFFFFF00)|0x00000040;
```

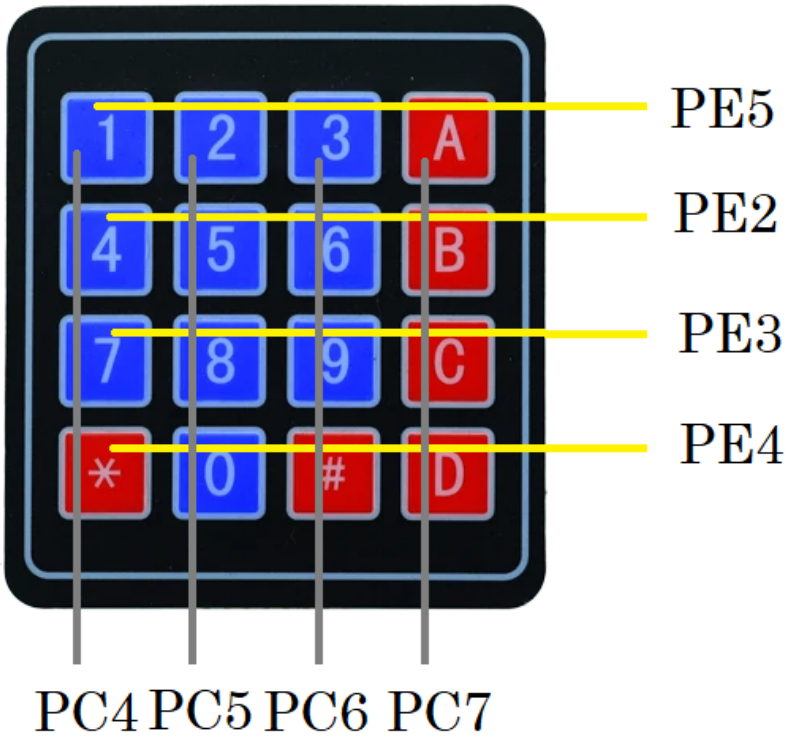


Figure 2: Keypad Rows and Columns

These codes are determined by the TM4C123 Microcontroller datasheet. First of all, we initialize the PortC and PortE with SYSCTL RCGCGPIO R. They are just like digital input buttons for that reason, we set the edge-sensitive and disabled the analog. To determine the edge level GPIO PORTC IS R, GPIO PORTC IBE R, GPIO PORTC IEV R definitions are important. Then we set the interrupt part. The initialization is setting according to the datasheet the link for the datasheet is in the references section.

## 3.2 Interrupts

Input devices allow the computer to gather information, and output devices can display information. Output devices also allow the computer to manipulate its environment. An interrupt is the automatic transfer of software execution in response to a hardware event that is asynchronous with the current software execution. This hardware event is called a trigger. The hardware event can either be a busy to ready transition in an external I/O device (like the UART input/output) After we enabled the interrupts, when one number or character is pressed, code gets into the handler functions. We have 2 Handler functions for both PortC and PortE in these functions the main idea is the same. Both of them for acknowledging the flags. For the PortC Handler function, the code is shown in **Algorithm 2**. When all steps have been done our project is completed. When we pressed sth. on the keypad, we acknowledge the interrupt.

```
void GPIOPortC_Handler(void){

        //unsigned long one = GPIO_PORTC_ICR_R ;
        //GPIO_PORTC_ICR_R = 0xF0;
  if(GPIO_PORTC_RIS_R&0x10){  // poll PC4
    GPIO_PORTC_ICR_R = 0x10;  // acknowledge flag4
                //UART_OutString("PC4\n");
  }
  else if(GPIO_PORTC_RIS_R&0x20){  // poll PC5
    GPIO_PORTC_ICR_R = 0x20;  // acknowledge flag4
                //UART_OutString("PC5\n");
  }
  else if(GPIO_PORTC_RIS_R&0x40){  // poll PC6
    GPIO_PORTC_ICR_R = 0x40;  // acknowledge flag4
                //UART_OutString("PC6\n");
  }
  else if(GPIO_PORTC_RIS_R&0x80){  // poll PC7
                GPIO_PORTC_ICR_R = 0x80;  // acknowledge flag4
  }
        else{
                volatile uint32_t errorCport = GPIO_PORTC_RIS_R;
        }
        GPIO_PORTC_ICR_R = 0xF0;
}
```

Algorithm 2: Handler Functions

In the handler function, we use the if statement. With and process we can get the necessary pin and GPIO PORTC RIS R is utilized in this register is set when an interrupt condition occurs on the corresponding GPIO pin.

```
void GPIOPortE_Handler(void){
  if(GPIO_PORTE_RIS_R&0x20){  // poll PE5

                if(GPIO_PORTC_RIS_R&0x10 && keyElements[12] == 0){
                        UART_OutString("1");
                        joker(keyElements,12);
                }
                else if(GPIO_PORTC_RIS_R&0x20 && keyElements[13] == 0){
                        UART_OutString("2");
                        joker(keyElements,13);
                }
                else if(GPIO_PORTC_RIS_R&0x40 && keyElements[14] == 0){
                        UART_OutString("3");
                        joker(keyElements,14);
                }
                else if(GPIO_PORTC_RIS_R&0x80 && keyElements[15] == 0){
                        UART_OutString("A");
                        joker(keyElements,15);
                }
    GPIO_PORTE_ICR_R = 0x20;  // acknowledge flag4
                //UART_OutString("PE5\n");
```

Algorithm 3: PortE Handler

After determined the PortC Handler, one must individually be setting the keys. The and operation is needed. In other sections, the Joker function and UART OutStrings will be explained.

According to the MatrixKeypad Init as we can see some initialization includes the interrupt process. For an interrupt to occur, these five conditions must be simultaneously true but can occur in any order.

1. device arm,
2. NVIC enable,
3. global enable,
4. interrupt priority level must be higher than current level executing,
5. hardware event trigger.

## 3.3 UART Communication

Here communicate with UART(Universal Asynchronous Receiver/Transmitter). This serial port allows the microcontroller to communicate with devices such as other computers, printers, input sensors, and LCDs. Serial transmission involves sending one bit at a time, such that the data is spread out over time. The total number of bits transmitted per second is called the baud rate. The reciprocal of the baud rate is

the bit time, which is the time to send one bit. The Algorithm 3 code shown as with the UART OutString("A") for all elements we must do that operation. For string, we get the output with OutString by the way we can get hexadecimal, Character, and decimal value with UART OutDec for example,

```
int res1 = (k*10)/l;
                int temp1 = res1/10;
                UART_OutString(" %_% Result: ");
                UART_OutUDec(k);
                UART_OutString(" / ");
                UART_OutUDec(l);
                UART_OutString(" = ");
                int temp2=k/l;
                UART_OutUDec(temp2);
                UART_OutChar('.');
```

Algorithm 4: UART Communication Code

In this part we use Teraterm to see the output then we calibrate it. First, the

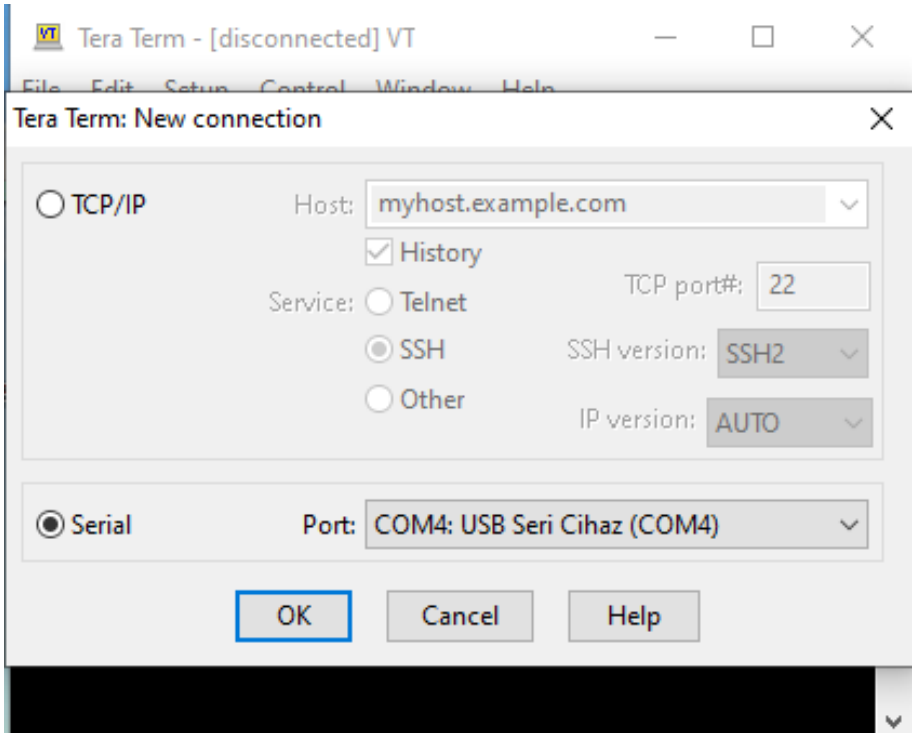


Figure 3: Uart Connection

TeraTerm program is opened and the section with USB input to which Texas Instrument EK-TM4C123GXL is connected is selected from the Serial section. After this process is completed and you press the "New Setting" button, again on the TeraTerm screen, as seen in the figure, the Speed 115200 is selected by first clicking

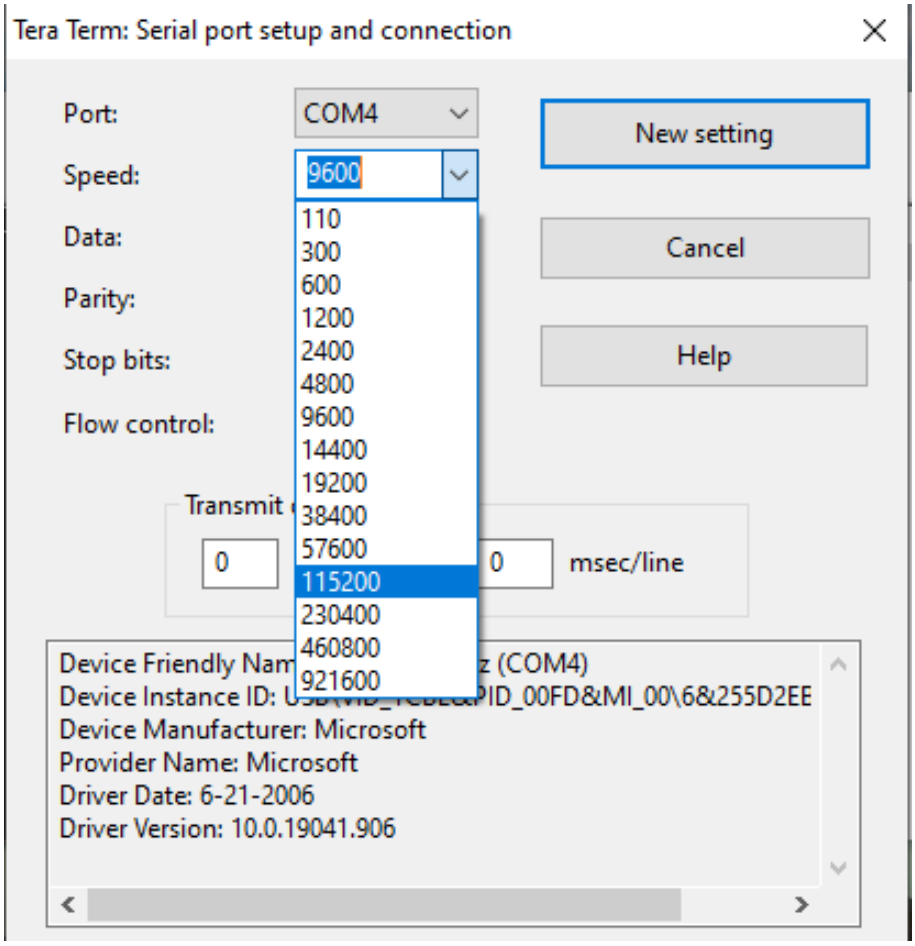


Figure 4: Uart Setup

the Setup and then the Serial port buttons. This step is very important to get the correct output from TeraTerm. Output from Keil is now properly visible in TeraTerm.

## 3.4 Extra Operations

In addition to that project, we want to design a basic calculator with a 4x4 keypad. So what we add all of these operations are 4 basic calculation processes. The joker function is a joker. We defined here the array length as sixteen (4x4=16). We gave "A" key is provide an addition operation, "B" key provides a division, "C" key for subtraction "*" is for multiplication and "D" key means equivalent. The keys are located in that array. First of all, as seen in the code, an array and an integer value are taken in func1. While the KLM value obtained shows which elements of the array were pressed to the keypad by the user, this algorithm enters the loop for multiplication, addition, and division operations. If no addition, multiplication, or division operation is performed after pressing the D key, the error message is printed on the screen, otherwise necessary operations are performed. In the code, some variables are assigned for division, multiplication, and addition, respectively, and enter the if condition in this way.

1. Addition
2. Division
3. Multiplication
4. Subtraction
5. Equivalent

### 3.4.1 Addition

In this operation we add two values by this code. We said that all keys have their special place in an array based on this we do the addition process in for loop. it gets the value that before pressed "A" and "D" then add them. KLM value is 15 means in the array "A" key. Then, if the user entered the "A" value to make an addition operation from the keypad, jump this condition. Similarly in the other operations.

```
void joker(int *array, int klm){
        int i= 0;
        int equal;
        for (i = 0; i< 16; i++){
                if(klm == 15){
                        if(array[i] == 1){
                                addition = numArray[i];
                        }
                        numArray[15] = 1;                              /
                }
```

Algorithm 5: Addition Keypad Definition

```
if(numArray[15] == 1){
                                        equal = addition + numArray[i];
                                        UART_OutString("(¯L¯) Result: ");
                                        UART_OutUDec(addition); UART_OutString(" + ");
                                        UART_OutUDec(numArray[i]);
                                        UART_OutString(" = ");
                                        UART_OutUDec(equal);
                                        numArray[15] = 0;
```

Algorithm 6: Addition Process

The addition operation is completed, makes 0 to "+"'s value to imply not do again addition operation

### 3.4.2 Division

If user entered the "B" value to make division operation from a keypad, jump this condition. We defined in here for four statement those are; undefined result, infinity result, normal and with float division. It gets the value that pressed before the "B" key and before the "D" key.

```
else if(klm == 3){
                        if(array[i] == 1){
                                division = numArray[i];
                        }
                        numArray[3] = 1;
                }
```

Algorithm 7: Division Keypad Definition

The division value makes equals to numArray's wanted value to take the user's number input.

```
void calculateDivide(int k,int l){
        if(l == 0){
                if(k == 0){
                        UART_OutString("Result is undefined! ");
                }else{
                        UART_OutString("Result is infinity! ");
                }
        }
        else{
                int res1 = (k*10)/l;
                int temp1 = res1/10;
                UART_OutString(" %_% Result: ");
                UART_OutUDec(k);
                UART_OutString(" / ");
                UART_OutUDec(l);
                UART_OutString(" = ");
                int temp2=k/l;
                UART_OutUDec(temp2);
                UART_OutChar('.');
                int lastResult = res1-temp1*10;
                UART_OutUDec(lastResult);
        }
}
```

Algorithm 8: Division Operation

### 3.4.3 Multiplication

When klm value is 8, then it means pressed "*", we get the multiplication process.

```
else if(klm == 8){
                        if(array[i] == 1){
                                multiplication = numArray[i];
                        }
                        numArray[8] = 1;
                }
```

Algorithm 9: Multiplication Keypad Definition

```
else if(klm == 8){
                        if(array[i] == 1){
                                multiplication = numArray[i];
                        }
                        numArray[8] = 1;
                }
```

Algorithm 10: Multiplication Process

In the numArray, the place where the multiplication process is 1 and it is understood that the last multiplication is done.

### 3.4.4 Subtraction

The two conditions occur in that process, if the first value is smaller than the second value then the output value must be negative, that's why we write the code for two separate situations. In the **Algorithm 12** shows that.

```
else if(klm == 7){
                        if(array[i] == 1){
                                substraction = numArray[i];
                        }
                        numArray[7] = 1;
                }
}
```

Algorithm 11: Subtraction Array Definition

```
else if(numArray[7] == 1){
                if( substraction >= numArray[i]){
                        equal = substraction - numArray[i];
                        UART_OutString(":-> Result: ");
                        UART_OutUDec(substraction); UART_OutString(" - ");
                        UART_OutUDec(numArray[i]);
                        UART_OutString(" = ");
                        UART_OutUDec(equal);
                }else{
                        equal = numArray[i] - substraction;
                        UART_OutString(" :-< Result: ");
                        UART_OutUDec(substraction); UART_OutString(" - ");
                        UART_OutUDec(numArray[i]);
                        UART_OutString(" = -");
                        UART_OutUDec(equal);
                }
                numArray[7] = 0;
        }
```

Algorithm 12: Subtraction Process

### 3.4.5 Equivalent and No Operation

If we pressed "D" one after another, we show the No operation message on teraterm.

```
else{
                                        UART_OutString("No operation");
                                }
                        }
                }
                else if(i != klm){
                        array[i] = 0;
                }else{
                        array[klm] = 1;
                }
        }
        OutCRLF();
        bekle();
}
```

When D is pressed the equal result is shown in Teraterm.

Algorithm 13: Equivalent

# 4 Outputs

This section shows the teraterm main screen. Each number and characters show line by line. All the UARTOut command shows the output when the statement is correct. The **fig.6** and **fig.5** are example outputs.

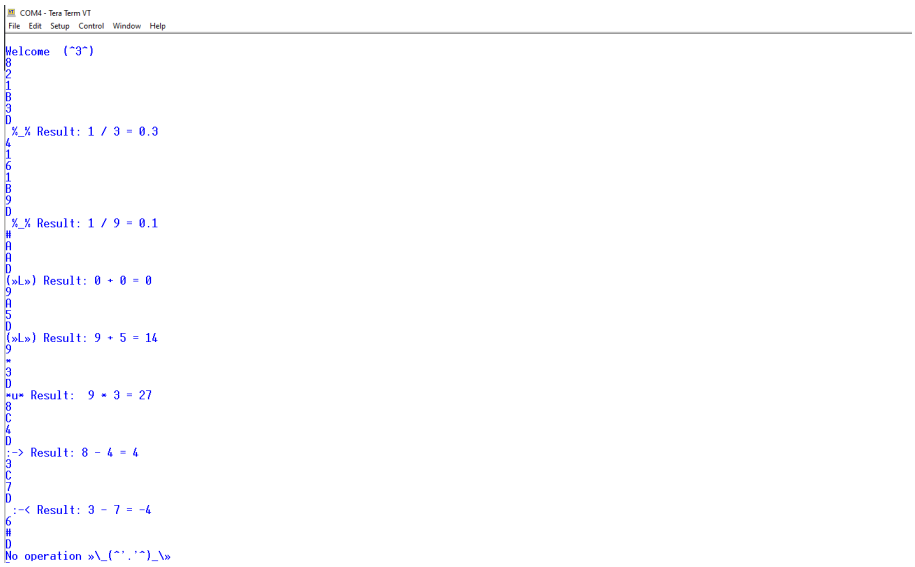


Figure 5: Tera Term Output Values

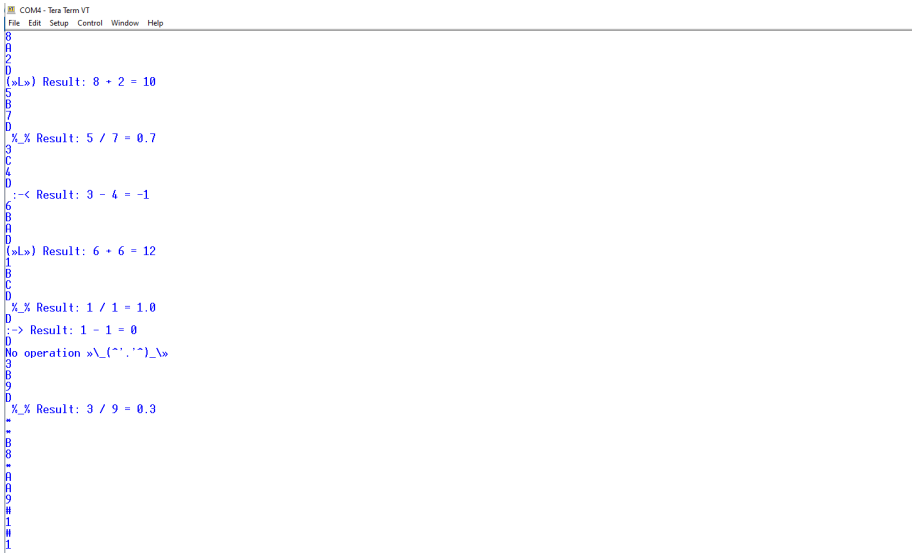


Figure 6: Tera Term Output Values-2

# A References

https://users.ece.utexas.edu/~valvano/Volume1/E-Book/C12_Interrupts.htm
https://ee315.cankaya.edu.tr/uploads/files/tm4c123gh6pm.pdf
https://ee469.cankaya.edu.tr/course.php?page=index
https://users.ece.utexas.edu/~valvano/Volume1/E-Book/
http://users.ece.utexas.edu/~valvano/arm/