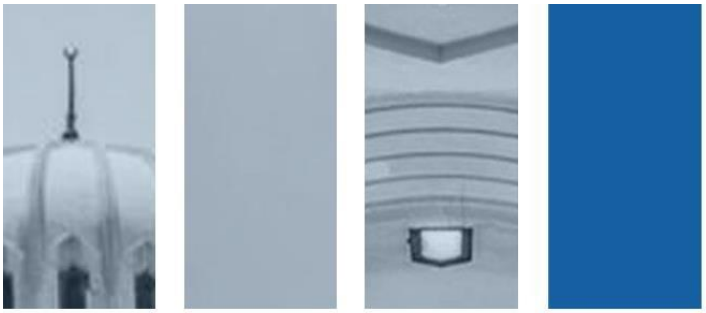

Program: CESS

Course Code: ***CSE 211***

Course Name: ***Introduction to Embedded Systems***

Submitted to:

Prof: **Sherif A. Hammad**

- **Ahmed Sameh** **19P5861**
- **Abdelrahman Ayman** **19P6458**
- **Osama Ayman** **19P1609**
- **Osama Lasheen** **19P6937**
- **Elsaeed Ahmed** **19P1087**

Contents

Table of Figures

1.0 Introduction

The goal of this project is to design a simple calculator with 2 extra features: a timer and a stopwatch. The calculator shall do the basic operations that are addition, subtraction, multiplication, and division. The timer and stopwatch features will be both handled separately by the hardware.

In this project we have 3 main modes

1.1 Calculator

In this mode we take inputs from the user and print that input on the LCD. We will take two numbers (each more than 1 digit) and a sign between them. We use the keypad numbers to get the numbers.

A button: +
B button: -
C button: /
D button: =
* button: x

1.2 Timer Mode

In this mode the user will set a time using the keypad, the timer will start counting down and as soon as it reaches the time zero it will trigger a buzzer. When we switch to this mode initially present 00:00 on the LCD then take the input from the user as minutes and seconds. The input is written as minutes and seconds then start the timer as soon as the user presses on the D button on the keypad.

1.3 Stopwatch Mode

In this mode the user will be using three buttons, one to start the stopwatch, one to pause the stopwatch, and one to reset the value back to 00:00.

When we switch to this mode initially present 00:00 on the LCD. Whenever the user presses on the start button start incrementing the stopwatch.

2.0 Circuit Topology

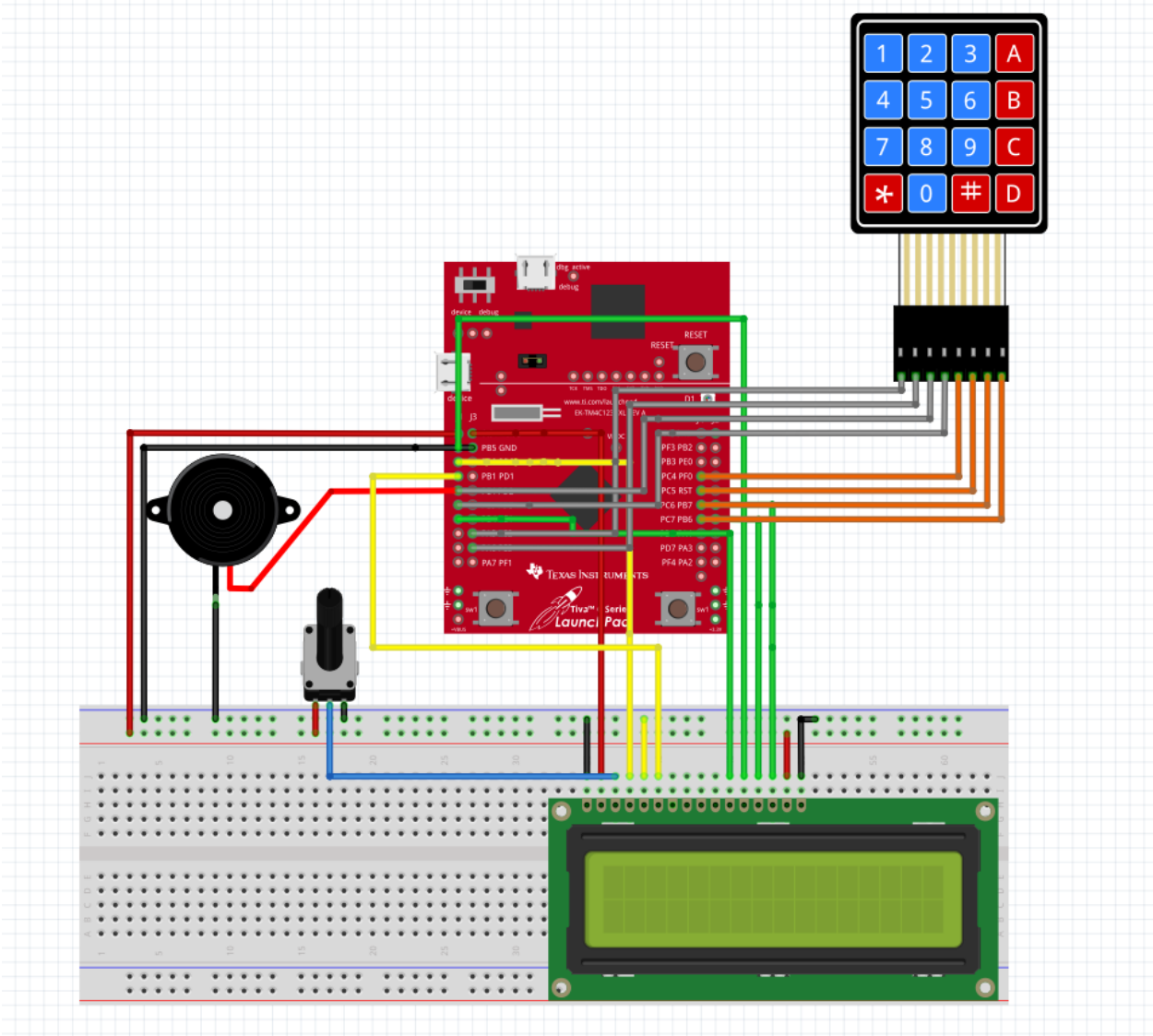

Figure 1 Circuit Topology

3.0 Flow Chart

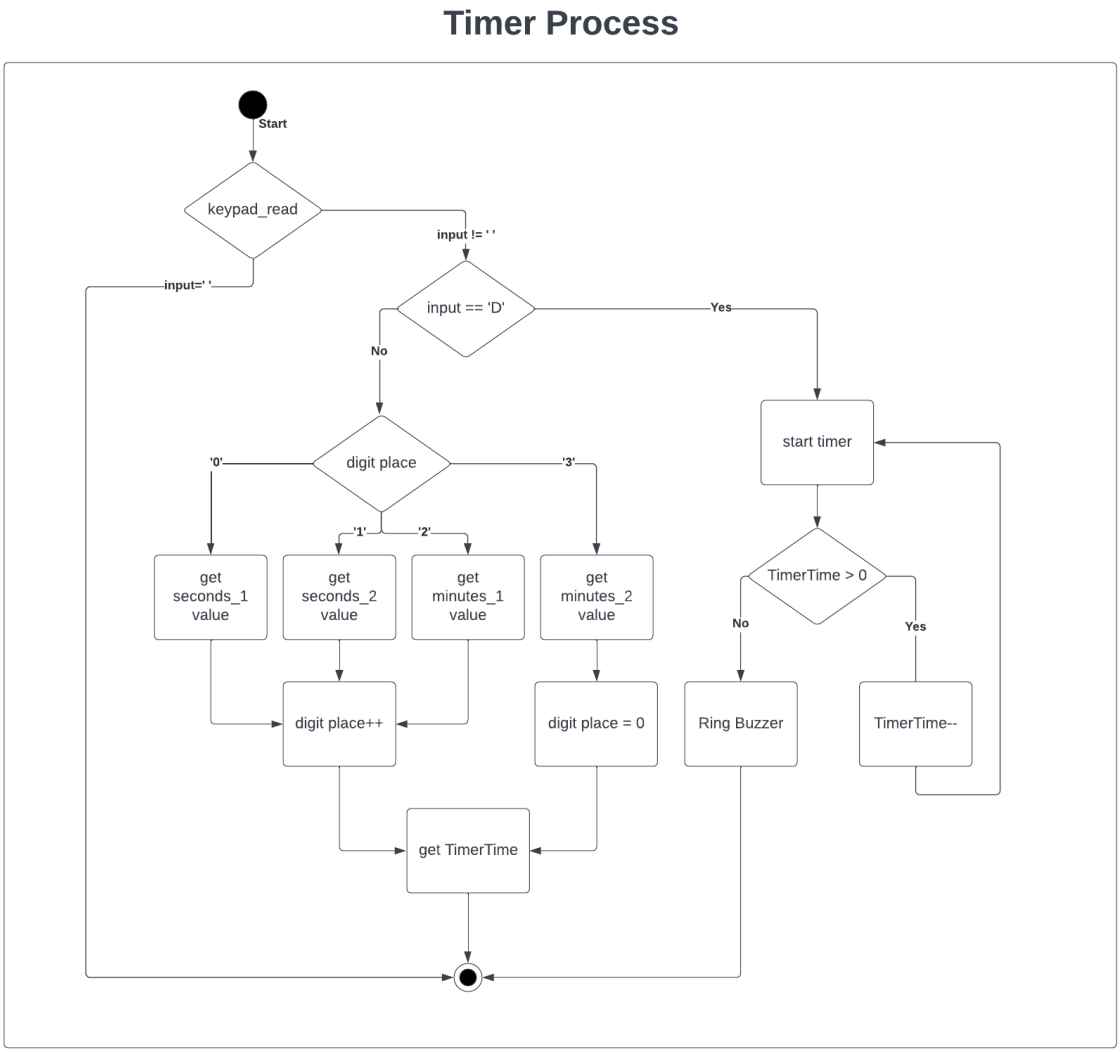

Figure 2 Timer Process

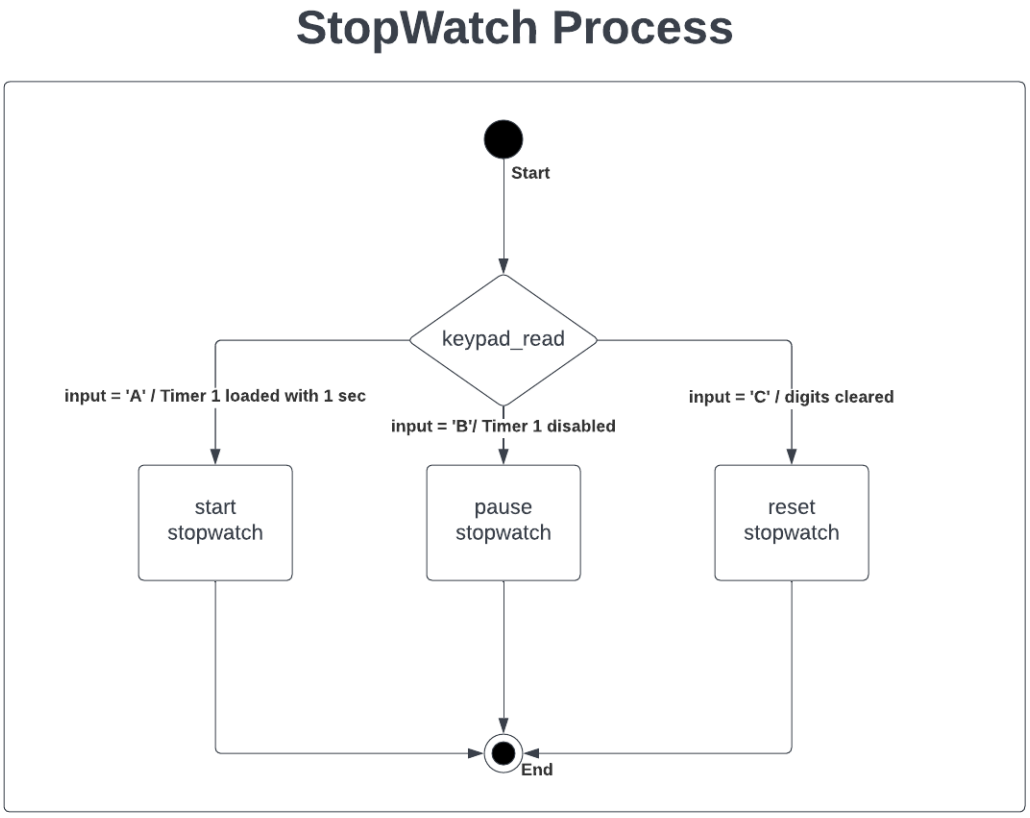

Figure 3 Stopwatch Process

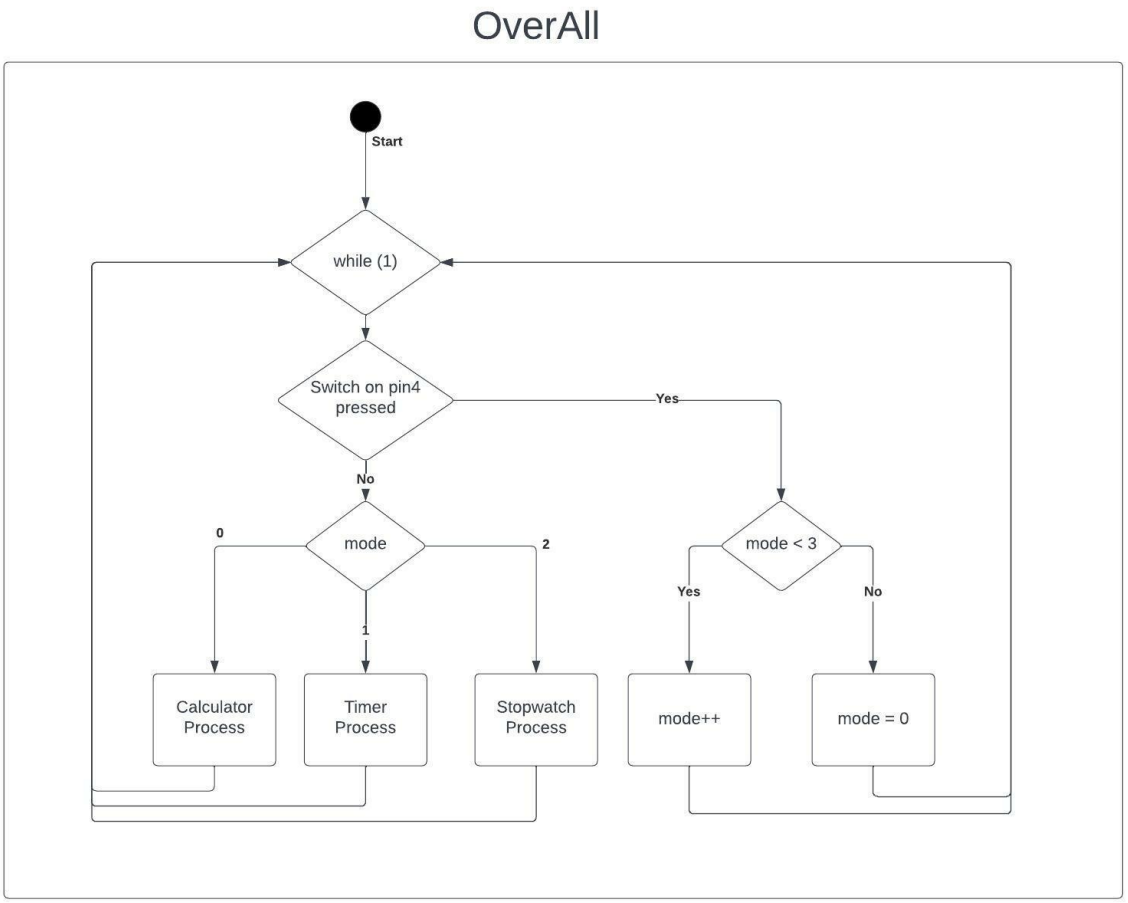

Figure 4 Overall Flowchart

4.0 Function Description

```
// Change Input/Output Modes {0 : Stopwatch, 1 : Calculator, 2 : Timer}
void ChangeMode(){
  while(GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4) == 0);
  if(mode == 2){mode=0;} // reseting the mode to the first one (0)
  else{mode++;} // incrementing the current mode to the next one
  minutes_1 = 0;
  minutes_2 = 0;
  seconds_1 = 0;
  seconds_2 = 0;
  x=0;y=0;z=0;isOpEntered = false;opCode = 'A';
  TimerDisable(TIMER1_BASE, TIMER_BOTH); // disabling Timer 1 (pausing the counting of the stopwatch)
  TimerIntClear(TIMER1_BASE, (TIMER_TIMA_TIMEOUT | TIMER_TIMB_TIMEOUT));
  TimerIntClear(TIMER0_BASE, (TIMER_TIMA_TIMEOUT | TIMER_TIMB_TIMEOUT)); // clearing the interrupt of Timer 0
  TimerTime = 0;
  digit_place= '0';
  GPIOIntClear(GPIO_PORTF_BASE, GPIO_INT_PIN_4);
}
```

Figure 5 Screenshot 1

This function checks the switch (pin 4) if it's pressed and changes the mode according to mode diagram.

```
// Stopwatch Input
void GetStopWatchInput(){
  char input = keypad_read();

  if(input == ' '){return;}
  if (input == 'A'){
    TimerLoadSet(TIMER1_BASE, TIMER_BOTH, 16000000 - 1); // loading 1 second into Timer 1
    TimerEnable(TIMER1_BASE, TIMER_BOTH); // enabling Timer 1 to start counting
    printFlag = true;
  }
  else if (input == 'B'){
    TimerDisable(TIMER1_BASE, TIMER_BOTH); // disabling Timer 1 (pausing the counting of the stopwatch)
    printFlag = true;
  }
  else if (input == 'C'){
    // clearing the digits (reseting the stopwatch)
    minutes_1 = 0;
    minutes_2 = 0;
    seconds_1 = 0;
    seconds_2 = 0;

    TimerDisable(TIMER1_BASE, TIMER_BOTH); // disabling Timer 1 (pausing the counting of the stopwatch)
    printFlag = true;
  }
}
```

Figure 6 Screenshot 2

This function call keypad_read to get input. If input is 'A' the stopwatch is enabled and starts counting time measured in minutes:seconds (MM:SS) format.

If input is 'B', the stopwatch is disabled.

Finally, when input is 'C' the stopwatch is reset

```
102  // Stopwatch Timeout Function
103  void toggle_StopWatch(){
104    if(seconds_1 <= 9){
105      seconds_1 ++;
106      if ( seconds_1 == 10){
107        seconds_1 = 0;
108        seconds_2 ++;
109        if ( seconds_2 == 6){
110          minutes_1 ++ ;
111          seconds_2 = 0 ;
112          if ( minutes_1 == 10){
113            minutes_1 = 0;
114            minutes_2 ++;
115            if ( minutes_2 == 6){
116              minutes_2 = 0 ;
117            }
118          }
119        }
120      }
121    }
122    printFlag = true;
123  }
```

Figure 7 Screenshot 3

This function adjusts the minutes and seconds variables for display. So, when seconds reaches 59, the next second will trigger the minute variable to be incremented by 1 and the seconds variables will be reset to 00.

```
142
143  char getOp() {
144    if(opCode == 'A') return '+';
145    else if(opCode == 'B') return '-';
146    else if(opCode == 'C') return '/';
147    else if(opCode == '*') return 'x';
148  }
```

Figure 8 Screenshot 4

As there are four operations that our calculator can perform, we have to identify the operation code that the user provide as input. These code mapping is shown in the figure above.

```
136  // Calculator Input
137  void GetCalculatorInput(){
138    char input = keypad_read();
139    if(ChangeMode(input)){return;}
140    if(input == ' '){return;}
141    if (input == 'A' || input == 'B' || input == 'C' || input == '*'){
142      opCode = input;
143      sprintf(calculator, "%d%c", x, getOp());
144      isOpEntered = true;
145      printFlag = true;
146    }
147    else if(input == 'D'){
148      if (opCode == 'A'){
149        z = x + y;
150      }
151      else if (opCode == 'B'){
152        z = x - y;
153      }
154      else if (opCode == 'C'){
155        if(y != 0)
156          z = (float)x / y;
157      }
158      else if(opCode == '*'){
159        z = x * y;
160      }
161      sprintf(calculator, "= %.3f", z);
162      x=0;y=0;z=0;isOpEntered = false;opCode = 'A';
163      printFlag = true;
164    }
```

Figure 9 Screenshot 5

```
165     else{
166       if(!isOpEntered){
167         x *= 10;
168         x += (input - '0');
169         sprintf(calculator, "%d", x);
170         printFlag = true;
171       }
172       else{
173         y *= 10;
174         y += (input - '0');
175         sprintf(calculator, "%d%c%d", x, getOp(), y);
176         printFlag = true;
177       }
178     }
179   }
180
```

Figure 10 Screenshot 6

This function is responsible for getting the calculator input and performing the operation. It can be seen that when user enter ‘D’ the expression provided is calculated according to the operation code given.

```
185  // Seconds to MM:SS format
186  int get_seconds(){
187    return (seconds_1 + (seconds_2 * 10) + (minutes_1 * 60) + (minutes_2 *600));
188  }
189
190  // MM:SS format to Seconds
191  void seconds_to_format(int seconds){
192    int min = seconds/60;
193    int sec = (seconds - (min*60));
194    if(min < 10){
195      minutes_1 = min;
196      minutes_2 = 0;
197    }
198    else{
199      minutes_1 = min%10;
200      minutes_2 = (min/10)%10;
201    }
202    if(sec < 10){
203      seconds_1 = sec;
204      seconds_2 = 0;
205    }
206    else{
207      seconds_1 = sec%10;
208      seconds_2 = (sec/10)%10;
209    }
210  }
211
```

Figure 11 Screenshot 7

This function adjust the time format to be in MM:SS for displaying on the LCD.

```
212  // Start Timer One shot with 1 seconds
213  void Timerdelay() {
214    TimerLoadSet(TIMER0_BASE, TIMER_BOTH, 16000000 - 1);
215    TimerEnable(TIMER0_BASE, TIMER_BOTH);
216  }
217
218  // Timer Input
219  void GetTimerInput(){
220    char temp = keypad_read();
221    if(ChangeMode(temp)){return;}
222    if(temp != ' '){
223      if(temp == 'D'){
224          TimerTime = get_seconds();
225          Timerdelay();
226          printFlag = true;
227          return;
228      }
229      if(digit_place == '0'){digit_place = '1'; seconds_1 = (temp - '0');printFlag = true;}
230      else if(digit_place == '1'){digit_place = '2'; seconds_2 = (temp - '0');printFlag = true;}
231      else if(digit_place == '2'){digit_place = '3'; minutes_1 = (temp - '0');printFlag = true;}
232      else if(digit_place == '3'){digit_place = '0'; minutes_2 = (temp - '0');printFlag = true;}
233    }
234  }
```

Figure 12 Screenshot 8

Here, we are using one shot timer that triggers every 1 seconds.

```
// Timer One shot Timeout Function
void toggle_Buzzer(){
  if(TimerTime > 0){
    TimerIntClear(TIMER0_BASE, (TIMER_TIMA_TIMEOUT | TIMER_TIMB_TIMEOUT));
    TimerTime--;
    seconds_to_format(TimerTime);
    Timerdelay();
  }
  else{
    Toggle_Bit(GPIO_PORTE_DATA_R, 1); // turning on the buzzer
    delay(250);
    Toggle_Bit(GPIO_PORTE_DATA_R, 1); // turning off the buzzer
    delay(250);
    Toggle_Bit(GPIO_PORTE_DATA_R, 1); // turning oon the buzzer
    delay(250);
    Toggle_Bit(GPIO_PORTE_DATA_R, 1); // turning off the buzzer
    delay(250);
    Toggle_Bit(GPIO_PORTE_DATA_R, 1); // turning on the buzzer
    delay(250);
    Toggle_Bit(GPIO_PORTE_DATA_R, 1); // turning off the buzzer
    TimerIntClear(TIMER0_BASE, (TIMER_TIMA_TIMEOUT | TIMER_TIMB_TIMEOUT)); // clearing the interrupt of Timer 0
    TimerTime = 0;
    minutes_1 = 0;
    minutes_2 = 0;
    seconds_1 = 0;
    seconds_2 = 0;
    digit_place= '0';
  }
  printFlag = true;
}
```

Figure 13 Screenshot 9

When the timer elapses, it is cleared and reset using this function.

```
int main() {
  //PORT F Setup
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
  while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOF));

  //Enabling Timer 0 & 1 Interrupts
  timerInit();
  stopwatchInit();
  __asm("CPSIE I");

  //Red Led Setup
  //GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);

  //Button Setup
  GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_4);
  GPIOIntRegister(GPIO_PORTF_BASE, ChangeMode);
  GPIOIntTypeSet(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_FALLING_EDGE);
  GPIOIntEnable(GPIO_PORTF_BASE, GPIO_INT_PIN_4);
  Set_Bit(GPIO_PORTF_PUR_R, 4);

  //KeyPad Setup
  keypad_init();

  //LCD Setup
  LCD4bits_Init();     //Initialization of LCD
  LCD4bits_Cmd(0x01);         //Clear the display
  LCD4bits_Cmd(0x80);           //Force the cursor to beginning of 1st line
  delayMs(500);     //delay 500 ms for LCD (MCU is faster than LCD)

  //Welcome
  welcome();
```

Figure 14 Screenshot 10

```
void timerInit() {
  //Timer 0 Setup
  SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
  while(!SysCtlPeripheralReady(SYSCTL_PERIPH_TIMER0));
  TimerConfigure(TIMER0_BASE, TIMER_CFG_ONE_SHOT);

  //Enabling Timer 0 Interrupts
  TimerIntEnable(TIMER0_BASE, (TIMER_TIMA_TIMEOUT | TIMER_TIMB_TIMEOUT));
  TimerIntRegister(TIMER0_BASE, TIMER_BOTH, toggle_Buzzer);
}
```

Figure 15 Screenshot 11

```
void stopwatchInit() {
  //Timer 1 Setup
  SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
  while(!SysCtlPeripheralReady(SYSCTL_PERIPH_TIMER1));
  TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

  //Enabling Timer 1 Interrupts
  TimerIntEnable(TIMER1_BASE, (TIMER_TIMA_TIMEOUT | TIMER_TIMB_TIMEOUT));
  TimerIntRegister(TIMER1_BASE, TIMER_BOTH, toggle_StopWatch);
}
```

Figure 16 Screenshot 12

```
void stopwatchInit() {
  //Timer 1 Setup
  SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
  while(!SysCtlPeripheralReady(SYSCTL_PERIPH_TIMER1));
  TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

  //Enabling Timer 1 Interrupts
  TimerIntEnable(TIMER1_BASE, (TIMER_TIMA_TIMEOUT | TIMER_TIMB_TIMEOUT));
  TimerIntRegister(TIMER1_BASE, TIMER_BOTH, toggle_StopWatch);
}
```

Figure 17 Screenshot 13

```
void keypad_init() {

  DIO_Init(PORTE, 1, OUT);
  DIO_Init(PORTE, 2, OUT);
  DIO_Init(PORTE, 3, OUT);
  DIO_Init(PORTE, 4, OUT);
  DIO_Init(PORTE, 5, OUT);

  DIO_Init(PORTC, 4, IN);
  DIO_Init(PORTC, 5, IN);
  DIO_Init(PORTC, 6, IN);
  DIO_Init(PORTC, 7, IN);

}
```

Figure 18 Screenshot 14

Initially, we initialize port F, Timer 0 by feeding them with system clock. Then, we enable interrupts for Timer 0 in order to count each 1 second. After that, we set up Red led and LCD and print the welcoming screen.

```
325   while(1){
326     if(mode != previousMode) {
327       printFlag = true;
328       previousMode = mode;
329     }
330     if(mode == 0){
331       GetCalculatorInput();
332       if(printFlag){
333           if(calculator[0])LCD_Print("Calculator:", calculator);
334           else LCD_Print("Calculator:", ".....................");
335           printFlag=false;
336       }
337     }
338     else if(mode == 1){
339       GetTimerInput();
340       if(printFlag){sprintf(timer, "%d%d:%d%d", minutes_2, minutes_1, seconds_2, seconds_1);LCD_Print("Timer:", timer);printFlag=false;}
341     }
342     else if(mode == 2){
343       GetStopWatchInput();
344
345       if(printFlag){sprintf(stopwatch, "%d%d:%d%d", minutes_2, minutes_1, seconds_2, seconds_1);LCD_Print("StopWatch:", stopwatch);printFlag=false;}
346     }
347   }
348 }
```

Figure 19 Screenshot 15

This is our main part. We keep on checking the mode. If mode is 0, then it is calculator. We get input and checks if it needs to be printed on LCD.

If mode is 1, it is timer. We get input from timer variables and print it.

Finally, if mode is 2, it is stopwatch. We get input from stopwatch variables and print it.

Video File:

https://engasuedu-my.sharepoint.com/personal/19p6458_eng_asu_edu_eg/_layouts/15/stream.aspx?id=%2Fpersonal%2F19p6458%5Feng%5Fasu%5Fedu%5Feg%2FDocuments%2FIMG%5F2984%2EMOV&ga=1

Source Code:

https://drive.google.com/file/d/1VpOemSFNfb4K4CRp4O_XGo3GFOqC5eTy/view?usp=share_link