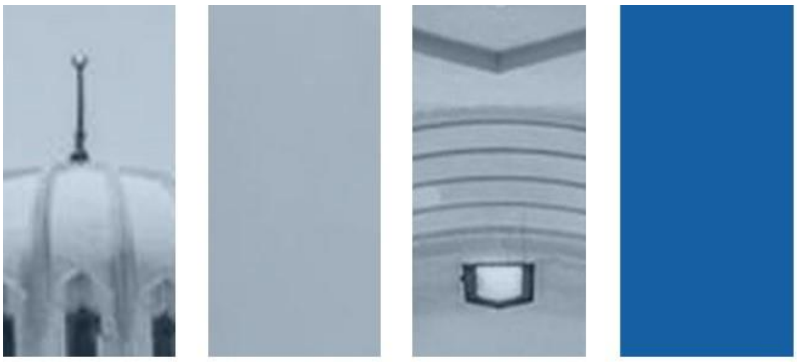

CSE211- Course Project

# Design of a Simple Calculator with Stopwatch & Timer Modes

***Introduction to Embedded systems***


***Submitted by:***

Ahmed Tarek Shafik 19P9286

Mohamed Hussein Mohamed 19P2570

Mostafa Mahmoud lofty Sakr 19P3024

Mohamed Ashraf Aboutaleb 19P7766

Mazen Ahmed Elaraby 19P9804

**Dr. Sherif A. Hammad**

**Ain Shams University**

**Faculty of engineering**



**Semester Fall 2022**

# Contents

**AIN SHAMS UNIVERSITY**
FACULTY OF ENGINEERING

# 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.

We have studied and implemented the basics of the GPIO interfacing by interfacing with the LCD, keypad and buzzer. Along with some experience with the GPTM and SYSTick modules. Programming the timers required us to use its interrupts to interface with the modules we mentioned earlier

Calculator supports the 4 basic mathematical operations (Addition, subtraction, multiplication and division) and notifies the user when he commits a syntax error.

The timer and stopwatch modules have the options to start, stop and reset. The timer fires an interrupt to trigger the buzzer after the specified time entered by the user has elapsed. Both the timer and stopwatch can run concurrently in the background without interfering with each other as each use a separate GPTM

Below is a Trial Run video for the project along with the source code:

https://drive.google.com/drive/folders/1YJXxPDZfV7lLwDPzOVdhpNmpVvj4iuj5?usp=share_link

# Circuit topology

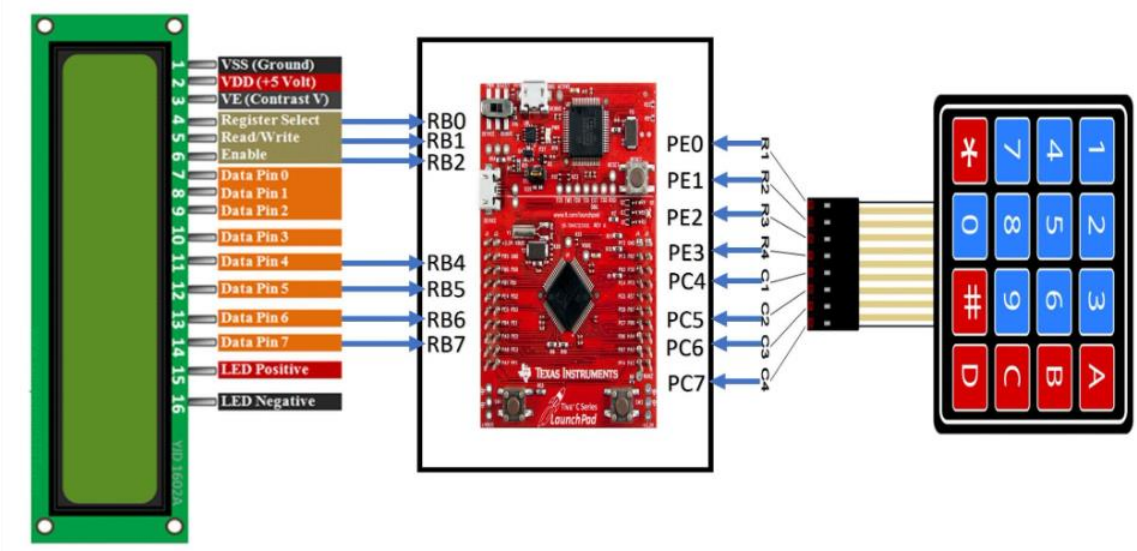


*Figure 2-1 Circuit Schematic*

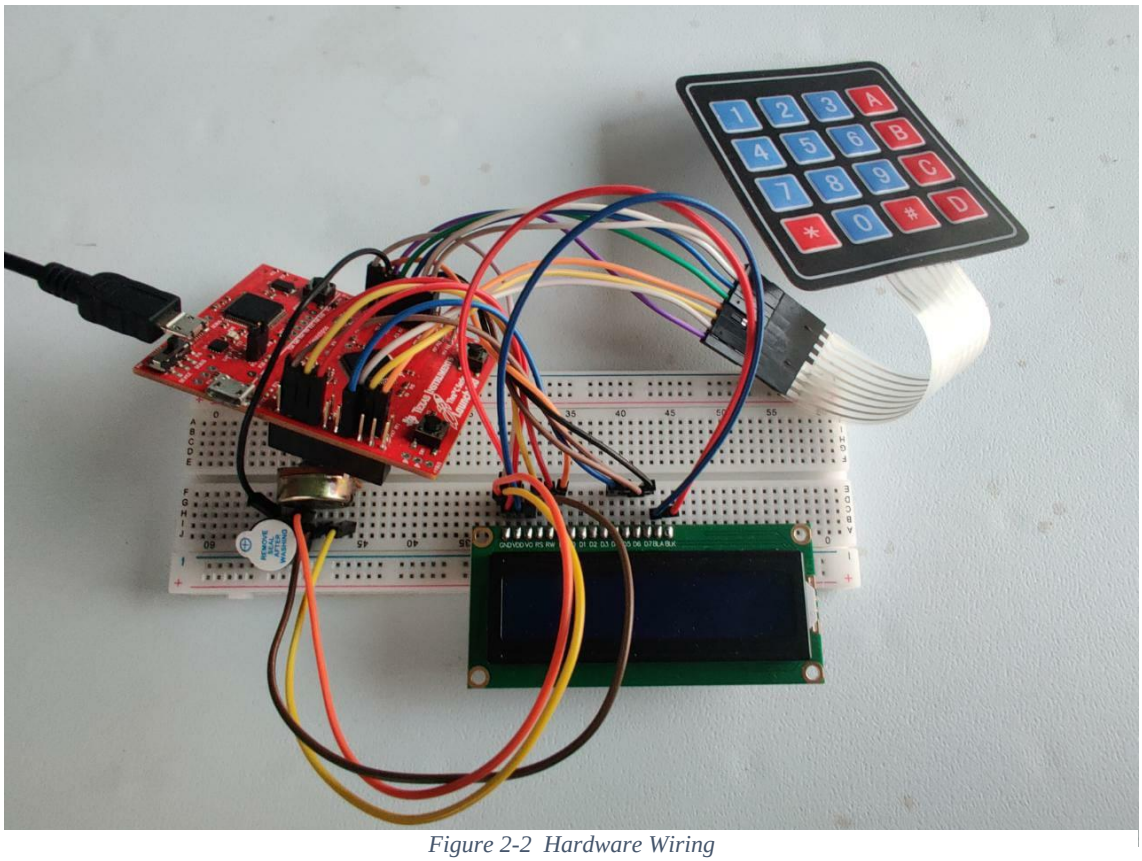

*Figure 2-2 Hardware Wiring*

# Flowcharts

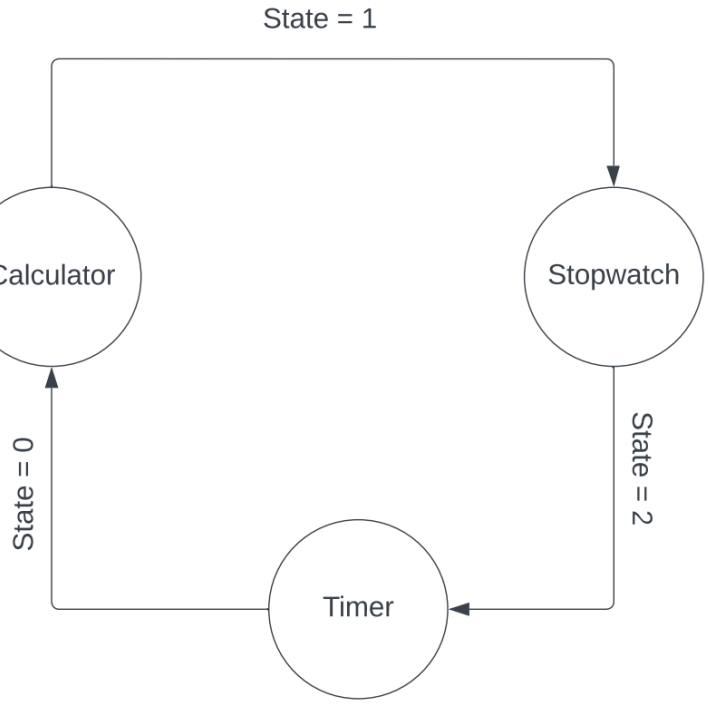


*Figure 3-1 Main FSM to switch modes*

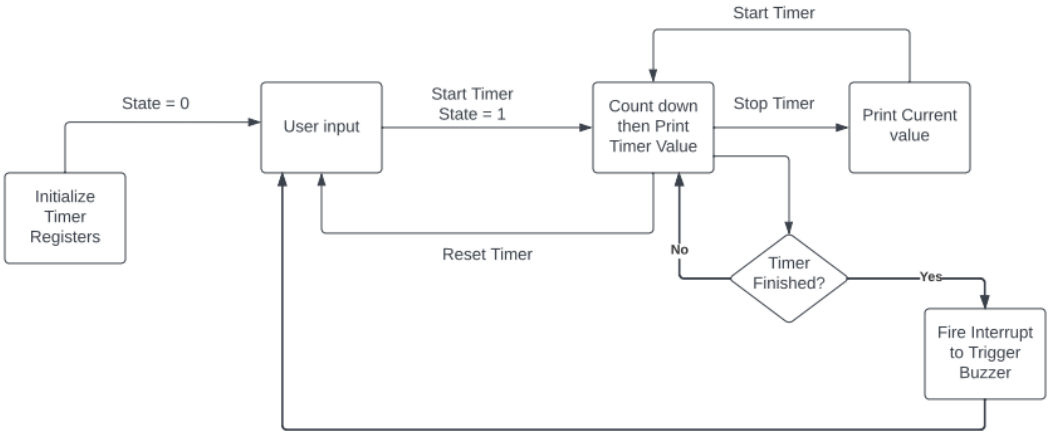


*Figure 3-2 Timer Flowchart*

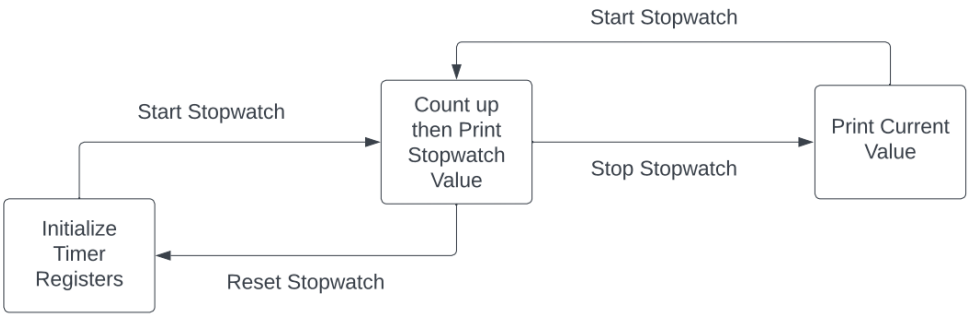


*Figure 3-3 Stopwatch Flowchart*

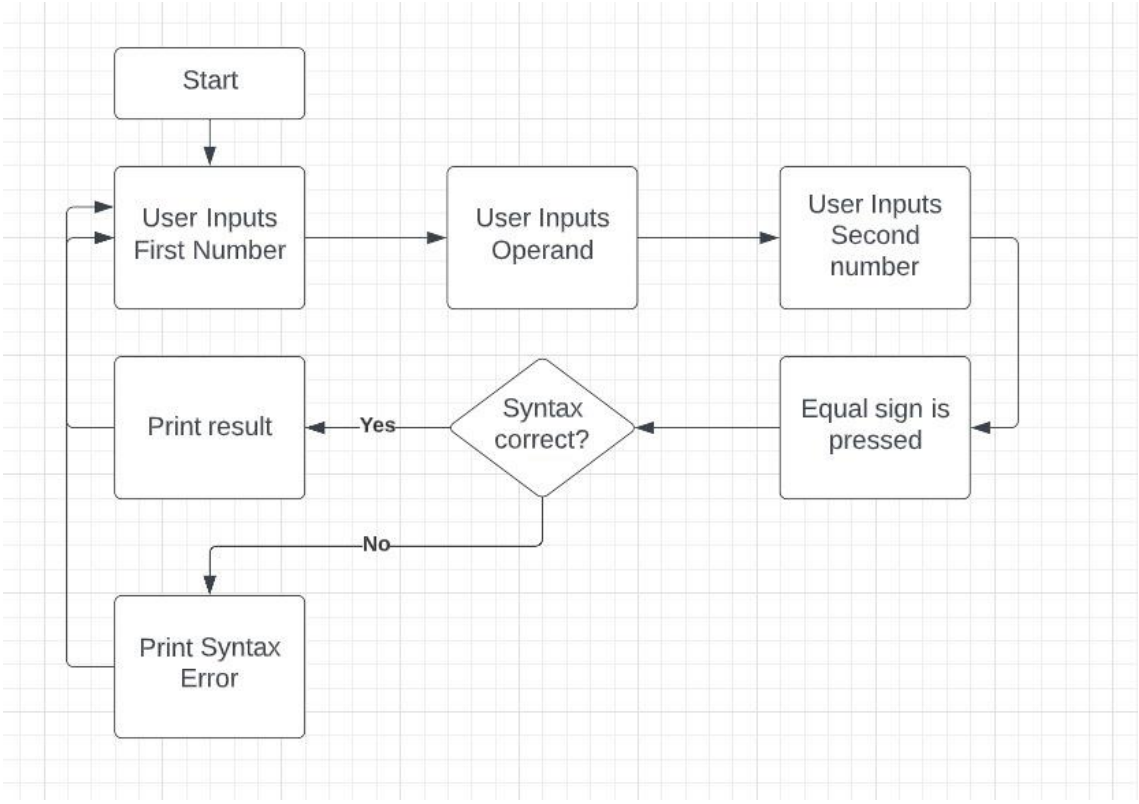


*Figure 3-4 Calculator Flowchart*

**AIN SHAMS UNIVERSITY**
FACULTY OF ENGINEERING

# Code Snippets

## Main Function

```
int main()
{
  __asm("CPSIE I"); //change processor status interrupt enable

  LCD_init();
  KeyPad_Init();
  stopwatch_init();
  timer_init();

  uint8 state = 0;
  while(1)
  {
    switch(state)
    {
      case(0):
        {
          Calculator();
          state = (state + 1) % 3;
          break;
        }
      case(1):
        {
          timer();
          state = (state + 1) % 3;
          break;
        }
      case(2):
        {
          stopwatch();
          state = (state + 1) % 3;
          break;
        }
    }
  }
  return 0;
}
```

# Keypad Functions

```
uint8 KEY[4][4]={

   {'1','2','3','A'},
   {'4','5','6','B'},
   {'7','8','9','C'},
   {'*','0','#','D'}
};
void KeyPad_Init(void){

  PORT_Configuration PTRE={UNLOCKED,0x0F,0x0F,0x00,0x0F};
  DIO_init(PORTE,&PTRE);

  GPIO_PORTE_ODR_R =0x0F; //open drain

  PORT_Configuration PTRC={UNLOCKED,0xF0,0x00,0xF0,0xF0};
  DIO_init(PORTC,&PTRC);
}

uint8 KeyPad_Read (int x){
//0 non-blocking, 1 blocking
while(1)
  {
    delayz(350); //debounce protection
    DIO_writePort(PORTE, 0x0);
    if (DIO_readPort(PORTC) != 0xF0)
       {
         DIO_writePort(PORTE, 0xF);
         for(int i=0; i<4; i++)
         {
           DIO_writePin(PORTE,i,0);
           for(int j=0; j<4; j++)
           {
             if (DIO_readPin(PORTC,j+4)==0){
               return KEY[i][j];
             }
           }
           DIO_writePin(PORTE,i,1);
         }
       }
    if(x == 0)
      return 0;
  }
  }
```

# LCD Functions

```
void LCD_init(void)
{
  //initializing port
  SYSCTL_RCGCGPIO_R |= 0x00000002U; //enabling clock for port B
  while((SYSCTL_PRGPIO_R & 0x00000002U) == 0){}; //checking
termination of clock enabling cycles
  GPIO_PORTB_DIR_R = 0xFF; //setting pin direction
  GPIO_PORTB_DEN_R = 0xFF; //digital enable

  //LCD initialization sequence
  delay_ms(20);
  LCD_nibble_write(0x30,0);
  delay_ms(5);
  LCD_nibble_write(0x30,0);
  delay_us(100);
  LCD_nibble_write(0x30,0);
  delay_us(40);

  LCD_nibble_write(0x20,0); //use 4-bits data bus
  delay_us(40);

  LCD_command(0x28); //set 4-bit data, 2-line, 5x7 font
  LCD_command(0x06); //move cursor right
  LCD_command(0x01); //clear screen, move cursor to home
  LCD_command(0x0F); //turn on display, cursor blinking

}

void LCD_nibble_write(uint8 data, uint8 control)
{
  data &= 0xF0; //clears lower nibble for control
  control &= 0x0F; //clears upper nibble for data


  GPIO_PORTB_DATA_R = data | control; // RS = 0, R/w = 0
  GPIO_PORTB_DATA_R = data | control | EN; //pulse E
  delay_us(0);
  GPIO_PORTB_DATA_R = data;
  GPIO_PORTB_DATA_R = 0;
}
```

```
void LCD_command(uint8 command)
{
  //sending upper then lower nibble
  LCD_nibble_write(command & 0xF0, 0);
  LCD_nibble_write(command<<4, 0);

  if (command < 4)
  {
    delay_ms(2);
  }
  else
  {
    delay_us(40);
  }
}

void LCD_data(uint8 data)
{
  LCD_nibble_write(data & 0xF0, RS);
  LCD_nibble_write(data<<4, RS);

  delay_us(40);
}

void LCD_string (char *str) //for printing whole strings
{
  int i;
  for(i=0;str[i]!=0;i++)
  {
    LCD_data(str[i]);
  }
}
```

# Calculator Functions

```
void Calculator(void){
  LCD_string("Calculator.");
  delay_ms(1000);
  LCD_command(1); //clears display

  uint32 value1 = 0;
  uint32 value2 = 0;
  uint32 result = 0;
  uint8 ip_state = 0;
  char x;
  uint8 val;
  uint8 clear_flag = 0;
  uint8 num_flag1 = 0;
  uint8 num_flag2 = 0;
  char op;
  char str1[20] = "0";
  char str2[20] = "0";
  char str3[20] = "0";
  while(1)
  {
    x = KeyPad_Read(1);
    if (clear_flag == 1)
    {
      LCD_command(1); //clears display
      clear_flag = 0;
    }
    val = x - '0';
    //if number and state is zero (1st number input)
    if ((val >= 0) && (val <= 9) && (ip_state == 0))
    {
      LCD_data(x);
      strcat(str1, (char[2]) { (char) x, '\0' });
      num_flag1 = 1;
    }
    //else if op, do it and increment state (operand)
    else if((x == 'A') || (x == 'B') || (x == 'C') || (x == 'D'))
    {
      op = x;
      switch(x) //printing appropriate character
      {
        case('A'):
          {
            LCD_data('A' - 22);
            break;
```

```
        }
      case('B'):
        {
          LCD_data('B' - 21);
          break;
        }
      case('C'):
        {
          LCD_data('C' - 25);
          break;
        }
      case('D'):
        {
          LCD_data('D' - 21);
          break;
        }
    }

    ip_state++;
  }
  //else if number and state is greater than 1
  else if ((val >= 0) && (val <= 9))
  {
    LCD_data(x);
    strcat(str2, (char[2]) { (char) x, '\0' });
    num_flag2 = 1;
  }
  //else if equals but syntax error
  else if ((x == '*') && ((ip_state > 1) || !(num_flag1 & num_flag2)))
  {
    LCD_command(1); //clears display
    LCD_string("Syntax Error.");
    delay_ms(1000);
    LCD_command(1); //clears display
    ip_state = 0;
    num_flag1 = 0;
    num_flag2 = 0;
    str1[0] = '\0';
    str2[0] = '\0';
    str3[0] = '\0';
  }
  //else if equals but not syntax error
  else if (x == '*')
  {
    //turn value strings to integers
    sscanf(str1, "%d", &value1);
    sscanf(str2, "%d", &value2);
```

```
      //perform operation
      switch(op)
      {
        case('A'):
          {
            result = value1 + value2;
            break;
          }

        case('B'):
          {
            result = value1 - value2;
            break;
          }

        case('C'):
          {
            result = value1 * value2;
            break;
          }

        case('D'):
          {
            result = value1 / value2;
            break;
          }
      }
      //print = and result
      LCD_data('=');
      sprintf(str3, "%d", result);
      LCD_string(str3);
      //terminating calculation
      ip_state = 0;
      clear_flag = 1;
      num_flag1 = 0;
      num_flag2 = 0;
      str1[0] = '\0';
      str2[0] = '\0';
      str3[0] = '\0';
    }
    //else if #, clear, break and switch mode
    else if(x == '#'){
      LCD_command(1); //clears display
      break;
    }
  }
}
```

# Timer Functions

```
void timer_init(void)
{
  SYSCTL_RCGCWTIMER_R |=0x2; //enabling clock
  WTIMER1_CTL_R = 0x0; //disabling timer
  WTIMER1_CFG_R = 0x4; //32-bit mode
  WTIMER1_TAMR_R = 0x1; //one-shot, down counter
  WTIMER1_TAPR_R = 63999; //prescale of 64000, 4 ms period, 250 tick = 1
sec
  WTIMER1_CTL_R = 0x2;

  //interrupt related registers
  WTIMER1_IMR_R = 0x1; //enabling GPTM interrupt
  NVIC_EN3_R = 0x1; //enabling NVIC interrupt
  NVIC_PRI24_R &=~0xE0; //setting interrupt priority to be the highest
}
void timer(void)
{
  LCD_string("Timer.");
  delay_ms(1000);
  LCD_command(1); //clears display

  //initializing variables and strings
  static uint8 state = 0;
  uint8 x = 0;
  uint8 mins;
  uint8 secs;
  uint8 break_flag = 1;
  uint8 broke = 0;
  uint8 start_flag = 1;
  char ip_str[10] = "--:--";
  char str1[20] = "0";
  char str2[20] = "0";

  while(break_flag)
  {
    //state 0: user input
    if(state == 0)
    {
      uint8 val;
      uint8 counter = 0;

      LCD_command(0x80);
      LCD_string(ip_str);
      while(counter < 5)
```

```
  {

    if (counter == 2)
    {
      counter++;
      continue;
    }

    x = KeyPad_Read(1); //in blocking mode
    val = x - '0';
    if ((val >= 0) && (val <= 9))
    {
      ip_str[counter] = x;
      counter++;
    }

    if (x == '#')
    {
      start_flag = 0;
      break_flag = 0;
      broke = 1;
      break;
    }
    LCD_command(0x80);
    LCD_string(ip_str);
  }
  if (broke == 1) //exits main while loop if # is pressed in state 0
  {
    LCD_command(1); //clears display
    break;
  }
  //convert string to number and fill interal load register
  mins = ((ip_str[0]-'0') * 10) + (ip_str[1] - '0');
  secs = ((ip_str[3]-'0') * 10) + (ip_str[4] - '0');
  WTIMER1_TAILR_R= ((mins * 60) + secs) * 250;

  state = 1;
  start_flag = 1;
}

//state 1: buttons and display
//buttons
x = KeyPad_Read(0); //in non-blocking mode
switch(x)
{
  case('A'):
    {
```

```
        WTIMER1_CTL_R = 0x3; //start
        LCD_command(1); //clears display
        break;
      }
    case('B'):
      {
        WTIMER1_CTL_R = 0x2; //stop
        break;
      }
    case('C'):
      {
        timer_init();//reset
        LCD_command(1); //clears display
        ip_str[0] = '-';
        ip_str[1] = '-';
        ip_str[3] = '-';
        ip_str[4] = '-';
        state = 0;
        start_flag = 0;
        break;
      }
    case('#'):
      {
        LCD_command(1); //clears display
        start_flag = 0;
        break_flag = 0;
        break;
      }
  }

  //printing
  if(start_flag == 1)
  {
    int i = WTIMER1_TAV_R / 250;
    secs = i%60;
    mins  = i/60;

    sprintf(str1, "%d", mins);
    sprintf(str2, "%d", secs);

    if(mins < 10)
    {
      str1[2] = '\0';
      str1[1] = str1[0];
      str1[0] = '0';
    }
```

```
      if(secs < 10)
      {
        str2[2] = '\0';
        str2[1] = str2[0];
        str2[0] = '0';
      }

      strcat(str1,":");
      strcat(str1,str2);
      LCD_command(0x80);
      LCD_string(str1);
    }
  }
}

void WTIMER1A_handler(void)
{
  WTIMER1_ICR_R = 0x1; //clearing interrupt flag
  GPIO_PORTB_DATA_R |= 0x8; //enabling buzzer given that the port is
initialized
  delay_ms(2000);
  GPIO_PORTB_DATA_R &=~0x8;
}
```

# Stopwatch Functions

```
void stopwatch_init(void)
{
  SYSCTL_RCGCWTIMER_R |=0x1; //enabling clock
  WTIMER0_CTL_R = 0x0; //disabling timer
  WTIMER0_CFG_R = 0x4; //32-bit mode
  WTIMER0_TAMR_R = 0x1; //one-shot, down counter
  WTIMER0_TAILR_R= 0xFFFFFFFE;
  WTIMER0_TAPR_R = 63999; //prescale of 64000, 4 ms period, 250 tick = 1
sec
  WTIMER0_CTL_R = 0x2;
}

void stopwatch(void)
{
  LCD_string("Stopwatch.");
  delay_ms(1000);
  LCD_command(1); //clears display

  //initialize variables and strings
  uint8 x;
  uint8 break_flag = 1;
  uint8 start_flag = 1;
  uint8 secs;
  uint8 mins;
  char str1[20] = "0";
  char str2[20] = "0";
  while(break_flag)
  {
    //buttons
    x = KeyPad_Read(0); //in non-blocking mode
    switch(x)
    {
      case('A'):
        {
          WTIMER0_CTL_R = 0x3; //start
          LCD_command(1); //clears display
          break;
        }
      case('B'):
        {
          WTIMER0_CTL_R = 0x2; //stop
          break;
        }
      case('C'):
```

```
        {
          stopwatch_init();//reset
          LCD_command(1); //clears display
          LCD_string("00:00");
          break;
        }
      case('#'):
        {
          LCD_command(1); //clears display
          start_flag = 0;
          break_flag = 0;
          break;
        }
    }
    //printing
    if(start_flag == 1)
    {
      int i = (0xFFFFFFFE - WTIMER0_TAV_R) / 250;
      secs = i%60;
      mins  = i/60;

      sprintf(str1, "%d", mins);
      sprintf(str2, "%d", secs);

      if(mins < 10)
      {
        str1[2] = '\0';
        str1[1] = str1[0];
        str1[0] = '0';
      }

      if(secs < 10)
      {
        str2[2] = '\0';
        str2[1] = str2[0];
        str2[0] = '0';
      }

      strcat(str1,":");
      strcat(str1,str2);
      LCD_command(0x80);
      LCD_string(str1);
    }
  }
}
```