Introduction to Embedded systems project report

DRIVE LINK : Embedded project team 9

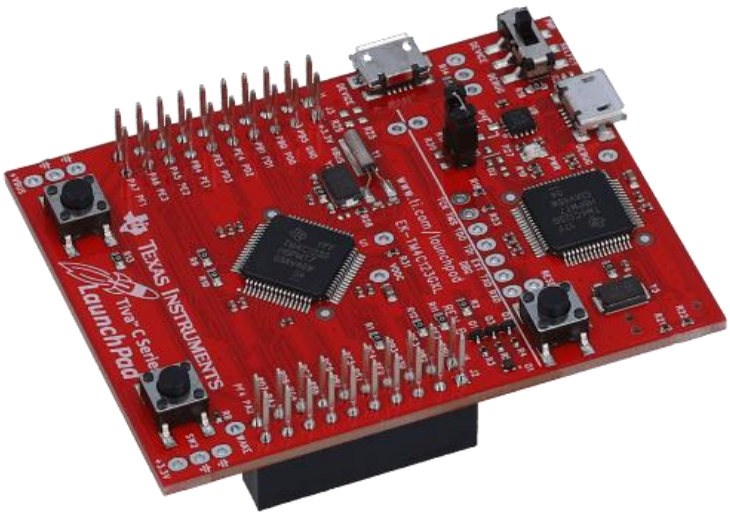
Nachaat fahmy Nachaat	**19p2460**
Shady Osama Wadeea	**19P2602**
Boules Emad Boules	**19p9291**
Kirollos Georges Boutros	**19p9058**
Ragui chawkat naguib	**20p1355**

Table of Contents

1.Intro

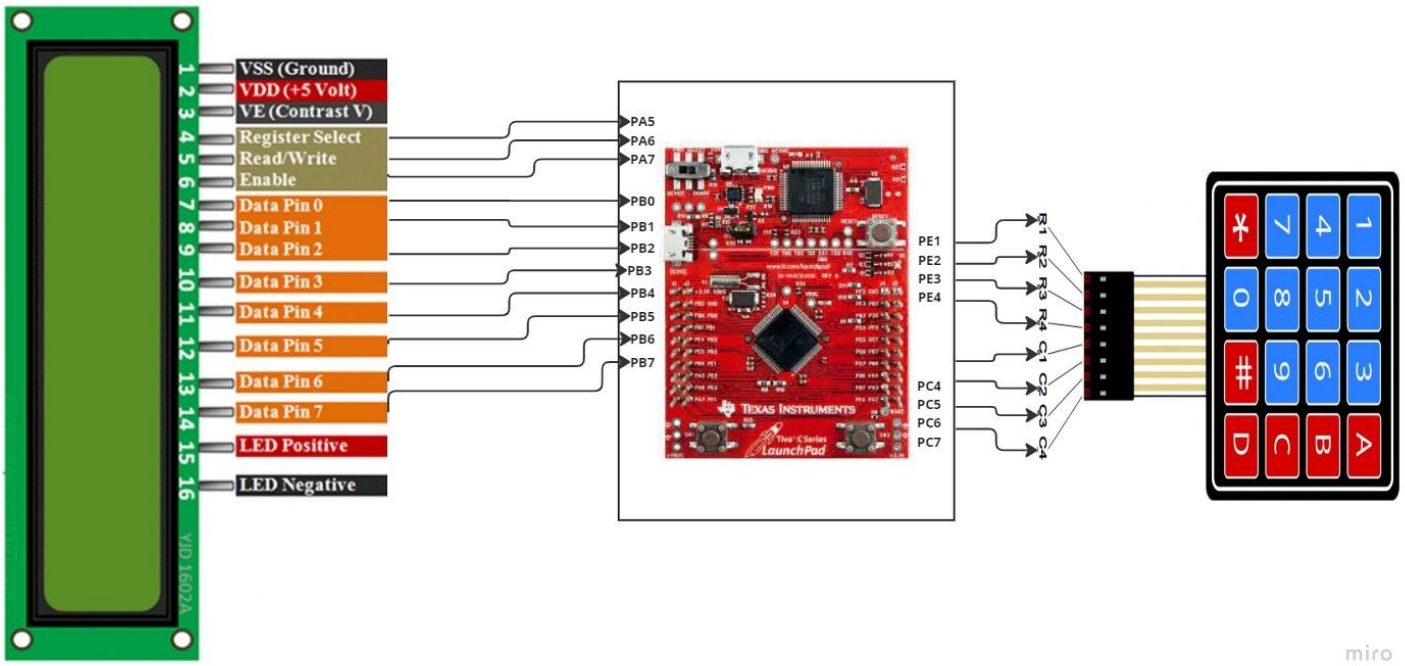
In this report we will discuss our embedded system project of the year where we have learnt about the microcontroller system and architecture .
We cannot work on a microcontroller without some good lab sessions on iar with C PROGRAMMING language . we have been working on the building process and digital I/O, timers, and interrupts applied on Arm Cortex M4 TivaC .

In this project we are going to show you on a LCD screen how we managed to do a calculator and timer and stopwatch using the tiva c and some jumpers and wires , bread boards , battery and potentiometer and a buzzer and a keypad .



Also how we managed to use all what we have learnt in this course to develop our skills as a software engineering student to do the required tasks the way it should be done . Of course , we have faced many problems due to the huge amount of details and how a single small mistake can destroy what we have done . So we will show you in the last section of this document how we have overcome our problems and how we have learnt from our mistakes

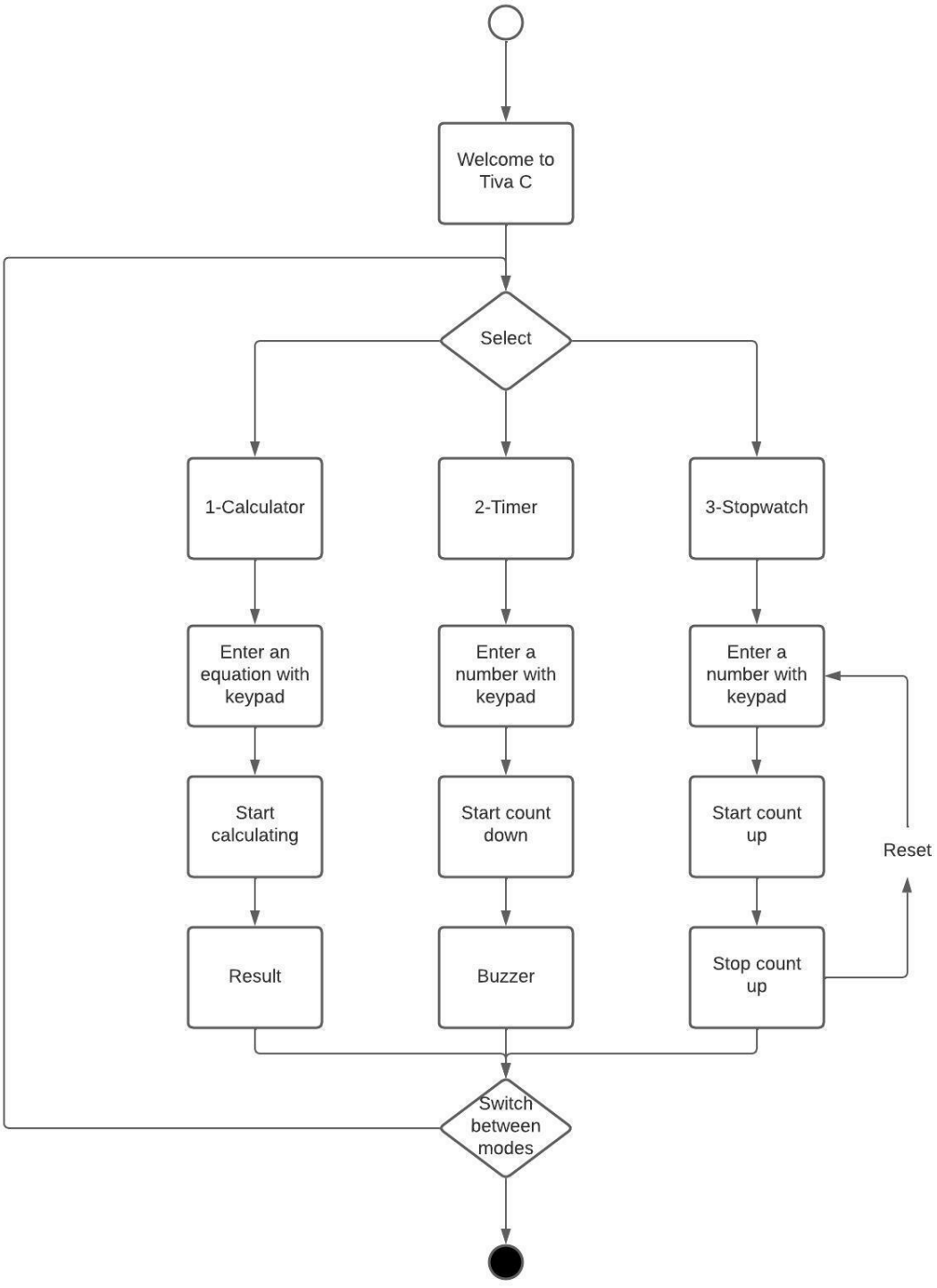
2. Circuits topology



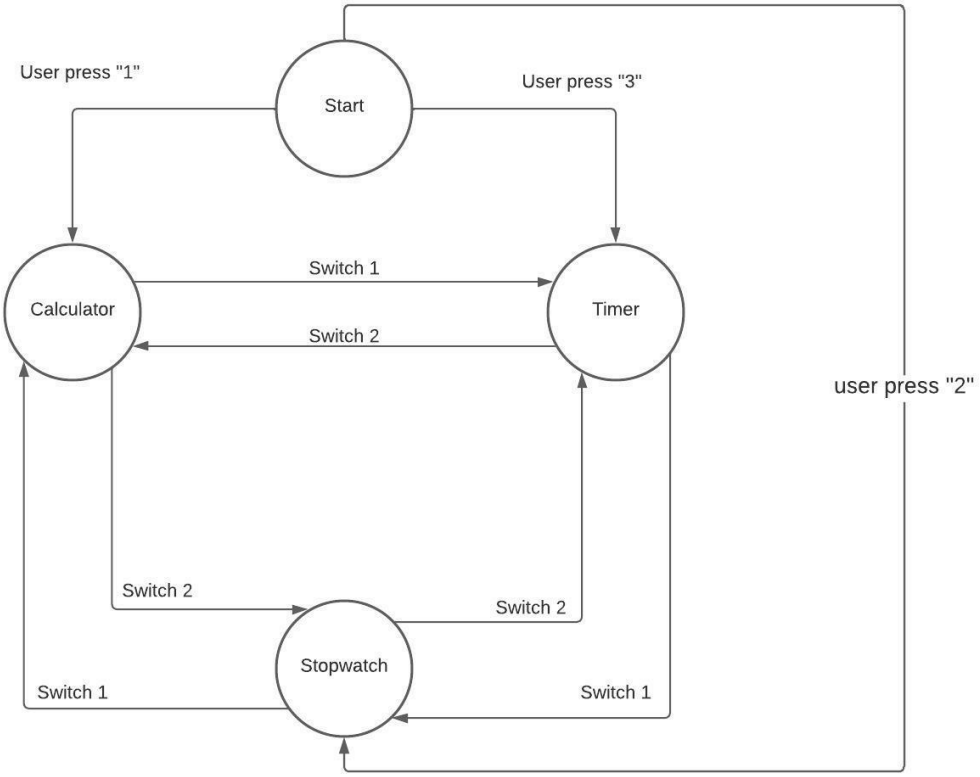
Hardware Connections :

* [PE1 - PE4] -> [R1 - R4] Rows
* [PC4 - PC7] -> [C1 - C4] Cols
* [PB0 - PB7] -> [BD0 - BD7] I/O Data Bus.
* PA5 -> RS Register Select.
* PA6 -> RW Read / Write Register.
* PA7 -> E Enable Pin.

3. Flow charts


Welcome to Tiva C
Select
1-Calculator
2-Timer
3-Stopwatch
Enter an equation with keypad
Enter a number with keypad
Enter a number with keypad
Start calculating
Start count down
Start count up
Reset
Result
Buzzer
Stop count up
Switch between modes

Our Code consists of three programs ,the calculator program , the timer program, and the stopwatch program. This Flow-chart above describe the general idea of the program .Each state with its input and output. Moreover ,it describe certain specific function for each state like the reset function in the stopwatch , the Buzzer mode in the timers .

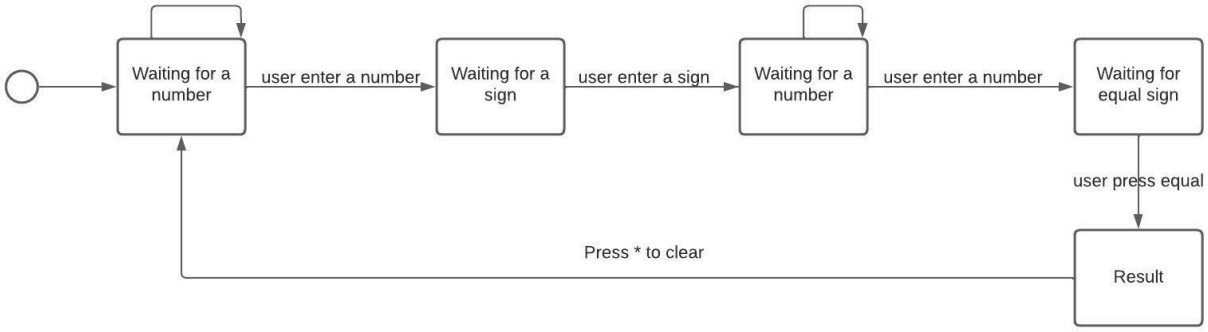


Finite State Machine

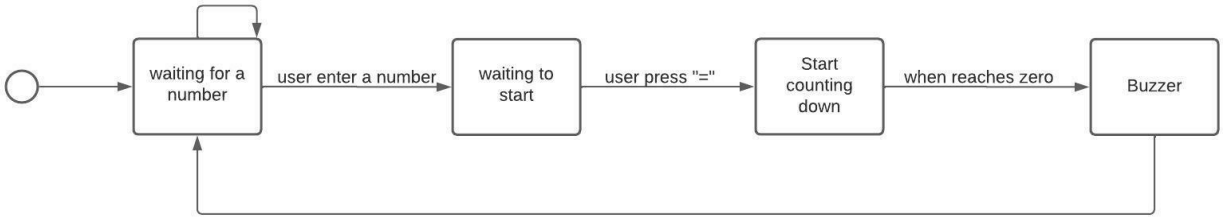
The program starts by choosing a number between one and three {1,2,3} ,then it goes to the desired state {Calculator,stopwatch,timer}and by using the two buttons in the TivaC we managed to switch between those three states as shown above .by enabling the interrupt on port F in the TivaC ,1st we initialized the portF ,2nd we enabled the interrupt on the processor level and enabled the NVIC for PortF and finally we enabled the interrupt for PortF and initialized specific functions in the vector table in the file (startup_ewarm.c) .In this stage the interrupt is waiting for the switch event to happen in order to go to its specific functions .

In the interrupt function related to the finite state machine we used a pointer to change the state through out the program .After finishing the interrupt functions ,the program checks the state pointer and accordingly goes to the calculator or the timer or the stopwatch .

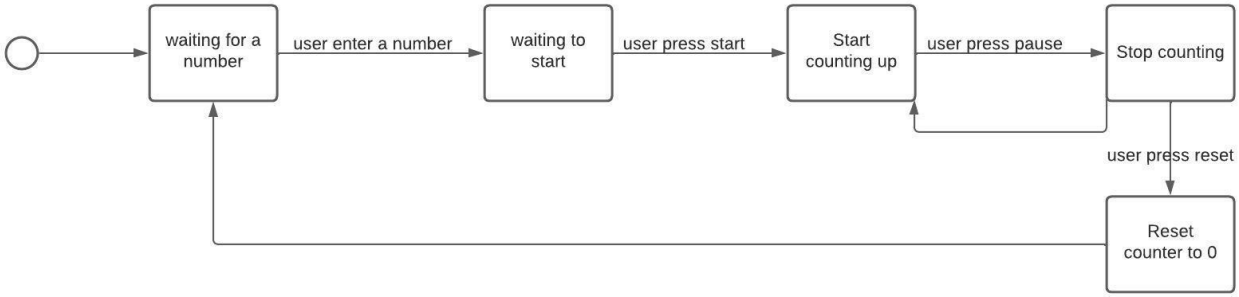
Calculator



Timer



Stopwatch



4. Code with comments

I. Adv_Calc.c (TIMER)

First of all we begin by :
timer flag is 0 since we do no want it to start its th interrupt until all the timer digits are filled
starting the timer on the LCD

```
int flag=0;
#define counter (*((volatile int *)0x2000774C))
int m , s ;


void start_timer(void){
  LCD_command(CURSOR_START_2ND_LINE);
  LCD_printString("00:00");
}
```

Void timer function

- ISR for the push button wo switch between modes
- if enters interrupt because of SW1 (PF4)
- change the state so we will be in the stopwatch state
- to let the keypad() know if we need a keypad input or just return because we want to switch modes
- to switch from the timer and clear the value
- to prevent the timer from interrupting other functions because we will switch the mode
- change the state so we will be in the stopwatch state
- to let the keypad() know if we need a keypad input or just return because we want to switch modes
- to switch from the timer and clear the value
- to prevent the timer from interrupting other functions because we will switch the mode

```
void
Timer(void)
{
if ( GPIO_PORTF_MIS_R &0x10 )
  {

    states_2=2;


   mode_flag=true;
    counter =0;
        MAP_IntDisable(INT_TIMER0A);
  }
  else if  ( GPIO_PORTF_MIS_R &0x01 )
  {
    states_2=0;


    mode_flag=true;                              //
    counter=0;                                   //
    MAP_IntDisable(INT_TIMER0A);                 //

  }



  GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_0);
   GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_4);
    MAP_IntDisable(INT_GPIOF);
  // MAP_IntMasterDisable();

}
```

Then we clear the interrupt pin

```
GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_0);
 GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_4);
  MAP_IntDisable(INT_GPIOF);
```

Void TimerReal function

Is a function to make the interrupt happen when the timer0a timeout in order to decrement the value of the timer on the lcd (counter variable)

- decrement the timer
- the following lines are to display the timer in the format 00:00
- then clear the timer interrupt
- then clear the interrupt flag

```
void TimerReal(void)
{
  if (counter != 0){

    --counter;
    LCD_command(CLEAR_DISPLAY);
       LCD_setcursorRowCol(0,5);
    LCD_printInt((counter/60)/10);
    LCD_setcursorRowCol(0,6);
    LCD_printInt((counter/60)%10);
    LCD_setcursorRowCol(0,7);
    LCD_printString(":");
    LCD_setcursorRowCol(0,8);
    LCD_printInt((counter-(counter/60)*60)/10);
    LCD_setcursorRowCol(0,9);
    LCD_printInt((counter-(counter/60)*60)%10);
    printf("%d \n", counter);
  }
  if (counter ==0){
    flag=0;


for(int i = 0; i < 10; ++i) {
        GPIO_PORTF_DATA_R ^= 0x2;
        delay_ms(500);
    }
                    }
  // Clear the timer interrupt.
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
```

Void Timer0_Init

- Enable the timer peripheral
- Enable the comp module
- configures the operating mode of the timer(s). The timer module is disabled before being configured and is left in the disabled state
- configures the timer load value; if the timer is running then the value is immediately loaded into the timer.
- registers the handler to be called when a timer interrupt occurs
- enables the indicated timer interrupt sources
- Enable the timer
- Enable global interrupts

```
void Timer0_Init(void){
    // Enable the timer peripheral.
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);  // En
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);   /
    TimerLoadSet(TIMER1_BASE, TIMER_A, SysCtlClockGet())
    TimerIntRegister(TIMER1_BASE, TIMER_A,TimerReal);//T
    IntEnable(INT_TIMER1A);              //This function
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);// E
    IntPrioritySet(INT_TIMER1A,4);


    //Enable global interrupts
     IntMasterEnable();

}
```

Void timer function

- to let the other functions that enters that I am in function not switch for interrupt mode
- I am in the timer state
- enable the interrupt to portf from NVIC
- This function allows the processor to respond to interrupts.
- processor status interrupt enable
- Enables the specified GPIO interrupts.
- Sets the priority of an interrupt.
- ensures that the interrupt handler specified by pfnIntHandler is called when an interrupt is detected from the selected GPIO port.
- Intialize the timer

```
void timer(){



mode_flag=false;                // to let the other func
  states_2 =1;                  //I am in the timer stat
  states_1=1;                   //I am in the timer stat


     MAP_IntEnable(INT_GPIOF);                  // enabl
     IntMasterEnable();                         //This f
  __asm("CPSIE I");                             // proce
       GPIOIntEnable(GPIO_PORTF_BASE,GPIO_INT_PIN_4) ;
      GPIOIntEnable(GPIO_PORTF_BASE,GPIO_INT_PIN_0) ;
IntPrioritySet(INT_GPIOF , 0) ;
 GPIOIntRegister(GPIO_PORTF_BASE,Timer);


 Timer0_Init();                                     /
```

- digit 1 for minute ,digit 0for minute,digit 1 for seconds ,digit 0for secondsDskj
- check if we have to cahnge the mdoeSdlmjskd,
- change state,loading screen ,go to calc
- the user have not yet have not filled the 4 values xx:xx
- we know if the mode have changed

```
while(1){
  int m1 ,m2,s1,s2 ;

  if (switch_to_timer_flag ==true){
    if (states_2==0){
                    loading();
                   calc();
                  }
 else if(states_2==2){
 loading();
 stopwatch();
 }
     }


  while(flag ==0){
    start_timer();

      if (switch_to_timer_flag ==true){
    if (states_2==0){
                    loading();
                   calc();
                  }
 else if(states_2==2){
 loading();
 stopwatch();
 }
     }
```

- wait while ther is no a keypad input
- ASCII Compensation
- put each digit in its mathematical place;
- waiting for the second input

```
   while (keypad_getkey()==0);
    m1 = keypad_getkey() - 48;
adjust the screen
   LCD_setcursorRowCol(1, 0);
   LCD_printInt(m1);
   delay_ms(250);

   while (keypad_getkey()==0);
    m2 = keypad_getkey() - 48;
   //adjust the screen
    LCD_setcursorRowCol(1, 1);
   LCD_printInt(m2);
   m = m1*10+m2;
   delay_ms(250);

   while (keypad_getkey()==0);
    s1 = keypad_getkey()- 48;
   LCD_setcursorRowCol(1, 3);
   LCD_printInt(s1);
   delay_ms(250);

   while (keypad_getkey()==0);
    s2 = keypad_getkey() - 48;
   LCD_setcursorRowCol(1, 4);
   LCD_printInt(s2);
   s = s1*10+s2;

   flag =1;

   }
```

II. Basic_Calc.c

- When you choose calculator mode, it will begin with “Let’s Go” in the first message and the cursor will blink in the second line. This is the function of void start_calc(void).
- void fini_calc(float result) is the function that ends the calculation it prints the result on the LCD and print “Press * to clear” if you want to do another calculation.
- void add(float num1, float num2) it takes 2 numbers from the user and add them. The function call void fini_calc(float result) after it finishes.
- void sub(float num1, float num2) it takes 2 numbers from the user and subtract them. The function call void fini_calc(float result) after it finishes.
- void mult(float num1, float num2) it takes 2 numbers from the user and multiply them. The function call void fini_calc(float result) after it finishes.

- void divi(float num1, float num2) it takes 2 numbers from the user at first, it checks the second number is 0 or not if it is 0 an error message appear, if it is not 0 it will check a second time if the first number is 0 or not if it is 0 the result will equal 0 if it is not 0 it will divide the 2 numbers. The function call void fini_calc(float result) after it finishes.
- void clear_data() this function clear all the data in all the variables.
- void Timer0IntHandler(void)
- void calc(void) this function do the calculations when we enter the calculator mode the '#' is the equal sign,

First of all we begin by :

- to store the number of different digits 10,100,1000,...
- initialize the first & second operand of calculation

```
float mem[10]  = {0};
float result   = 0;
//float num[10]  = {0};
//char sign[9]   = {0};
float num1 = 0;          //fir
float num2 = 0;          //sec
char opsympo  = '0';
char addition  = '0';
char subtract  = '0';
char multiply  = '0';
char division  = '0';
volatile unsigned char key1;
volatile float floatkey;
```

void start calc function is made to start the calculator screen shown after selecting option 1 from the home

```
void start_calc (void)
{
  LCD_command(DISPLAY_ON_CURSOR_ON);
  delay_ms(50);
  LCD_command(CLEAR_DISPLAY);
  delay_ms(50);
  LCD_command(CLEAR_DISPLAY);
  delay_ms(50);
  LCD_command(CLEAR_DISPLAY);
  delay_ms(50);
  LCD_command(CLEAR_DISPLAY);
  LED_blue();
  LCD_printString("Let's Go:");
  LCD_command(CURSOR_START_2ND_LINE);
  LCD_command(DISPLAY_ON_CURSOR_ON);
  LCD_command(DISPLAY_ON_CURSOR_BLINK);
}
```

void
Timer0IntHandler(void) is an ISR for the buttons to switch between modes

- Clears the specified interrupt sources, by clearing the interupt flag not to listen for interrupts as its job is done

```
void
Timer0IntHandler(void)                    //ISR for
{
  if ( GPIO_PORTF_MIS_R & 0x10 )
  {


    states_2=1;

    mode_flag=true;

  }
  else if  ( GPIO_PORTF_MIS_R & 0x01 )
  {
    states_2=2;

    mode_flag=true;
  }

  GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_0);
   GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_4);
//not to listen for interrupts as its job is done
   MAP_IntDisable(INT_GPIOF);
   MAP_IntMasterDisable();

}
```

Void calc (void)

- Wait for the 1st operand the the operator then the 2nd operand then the = sign and after its finished the clear to reset

- to prevent the timer from interrupting other functions because we will switch the mode
- enable the interrupt to portf from NVIC
- Enables the processor interrupt.
- Enables the specified GPIO interrupts.
- Registers an interrupt handler for a GPIO port.
- check the states to go for
- flag to stopr the interrupt
- we have to set the mode flag to false so the functions know we are making a function not interrupted by a switch yet
- wait for first operand of calculation
- this if statement checks, if the character 'f' is returned then we have entered the interrupt and need to change the state
-

```
void calc(void)
{
  states_2=0;
  mode_flag=false;
  states_1=0;
  MAP_IntDisable(INT_GPIOF);
   MAP_IntMasterDisable();
    MAP_IntEnable(INT_GPIOF);
     IntMasterEnable();
     __asm("CPSIE I");
      GPIOIntEnable(GPIO_PORTF_BASE,GPIO_INT_PIN_4)
       GPIOIntEnable(GPIO_PORTF_BASE,GPIO_INT_PIN_0)
                if (isEnter == true ){
 GPIOIntRegister(GPIO_PORTF_BASE,Timer0IntHandler);
                                    }
  if (states_2==1){
                    loading();
                    timer();
  }
  else if(states_2==2){
  loading();
  stopwatch();
  }
isEnter = true ;                                    // f
  start_calc ();
  while(1)
  { mode_flag=false;                                // w
    keyl = keypad_getkey();                         //wa
    if (keyl=='f'){                                //thi
      if (states_2==1){                            //wan
                    loading();
                    timer();
     }
```

```
 if (keyl=='f'){                          //this if statement checks, if the character 'f' is returned the
   if (states_2==1){                      //wanna Go to Timer?
                loading();
                timer();
   }
else if(states_2==2){                 // wanna go to topwatch?
loading();
stopwatch();
}
  }

  else if      (keyl != '.')           // wanna reset?
  {
    if      (addition == '1')
    {
      LCD_data(keyl);
      if(keyl == '=')
      {
        num2 = memfinalize(mem);     //to tranform the number from separate digits to a single number value
        add(num1,num2);
        clear_data();
      }
      else
      {
        floatkey = keyl - 48;       //////////ASCI Compensation
        store(floatkey, mem);
      }
      delay_ms(500);
    }
    else if (subtract == '1')
    {
      LCD_data(keyl);
      if(keyl=='=')
      {
    }
    else
    {
      floatkey = keyl - 48;      //////////ASCI Compensation
      store(floatkey, mem);
    }
    delay_ms(500);
  }
  else if (multiply == '1')
  {
    LCD_data(keyl);
    if(keyl == '=')
    {
      num2 = memfinalize(mem);      //to tranform the number from separate digits to a single number value
      mult(num1,num2);
      clear_data();
    }
    else
    {
      floatkey = keyl - 48;     //////////ASCI Compensation
      store(floatkey, mem);
    }
    delay_ms(500);
  }
  else if (division == '1')
  {
    LCD_data(keyl);
    if(keyl=='=')
    {
      num2 = memfinalize(mem);              //to tranform the number from separate digits to a single number value
      divi(num1,num2);
      clear_data();
    }
    else
    {
```

```
  }
  else
  {
    LCD_data(keyl);
    switch (keyl)
    {
  case '+':
       addition = '1';
       numl = memfinalize(mem);            //to tranform the number from separate digits to a single number value
       break;
  case '-':
       subtract = '1';
       numl = memfinalize(mem);            //to tranform the number from separate digits to a single number value
       break;
  case '*':
       multiply = '1';
       numl = memfinalize(mem);            //to tranform the number from separate digits to a single number value
       break;
  case '/':
       division = '1';
       numl = memfinalize(mem);           //to tranform the number from separate digits to a single number value
       break;
  default:
      floatkey = keyl - 48;   //////////ASCI Compensation
      store(floatkey, mem);
    }
    delay_ms(500);
  }
}
else if (keyl == '.')
```

```
      ,
      delay_ms(500);
    }
  }
  else if (keyl == '.')
  {
    start_calc();
    clear_data();
    delay_ms(50);
    clear_data();
    delay_ms(50);
    clear_data();
    delay_ms(50);
    clear_data();
  }
}
```

- void fini_calc(float result) after finishiing the calculation to print the result in their specific places shown below
- void add(float num1, float num2) make addition by the 2 entered numbers
- void sub(float num1, float num2) make subtraction by the 2 entered numbers
- void mult(float num1, float num2) make multiplication by the 2 entered numbers

```
void fini_calc(float result)
{
  LCD_command(DISPLAY_ON_CURSOR_OFF);
  LCD_command(CURSOR_START_2ND_LINE);
  LCD_printFloat(result);
  LED_green();
  LCD_command(CURSOR_START_1ST_LINE);
  LCD_printString("Press * to clear");
}

void add(float num1, float num2)
{
  //result = num[0] + num[1];
  result = num1 + num2;
  fini_calc(result);
}

void sub(float num1, float num2)
{
  //result = num[0] - num[1];
  result = num1 - num2;
  fini_calc(result);
}

void mult(float num1, float num2)
{
  //result = num[0] * num[1];
  result = num1 * num2;
  fini_calc(result);
}
```

Void Clear_data is to clear the data

```
void clear_data()
{
//  for(int i = 0; i < 10; i++)
//    num[i] = 0;
  num1 = 0;
  num2 = 0;
  result    = 0;
  floatkey  = 0;
  addition  = '0';
  subtract  = '0';
  multiply  = '0';
  division  = '0';
}
```

I. stopwatch_2.c

we will begin by void TimerIntHandler the interrupt by the switch

- is Sw 1 the interrupt source?
- change the state to calculator
- tell the functions below I am in the states mode
- is SW2 the interrupt source
- tell the functions below I am in the states mode
- then clear the interrupts flag

```
void
Timer1IntHandler(void)
{

 if ( GPIO_PORTF_MIS_R &0x10)
  {

    states_2=0;


   mode_flag=true;

  }
  else if  ( GPIO_PORTF_MIS_R &0x01 )
  {
    states_2=1;

    mode_flag=true;

  }



    //Clear the interrupts flags so that when

GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_0);
GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_4);
MAP_IntDisable(INT_GPIOF);
MAP_IntMasterDisable();



}
```

Void oneshot configure function

- activate Timer 0
- timer first
- bit mode for timer then we use timer a only
- D4=1 -> COUNT UP ///// D1D0=01 -->one shot
- making the prescale = 244

- making the Interval Load register =65535 then we have 244*65535~16000000 MHZ = 1 second

```
void oneShotConfigure(void){

    SYSCTL_RCGCTIMER_R |= 0x01;
    TIMER0_CTL_R = 0x0;
    TIMER0_CFG_R = 0x4;
    TIMER0_TAMR_R = 0x11;
    TIMER0_TAPR_R |= 0xF4;
    TIMER0_TAILR_R = 0xffff;
}
```

- void enableTimer() enable the timer
- void disnableTimer() disnable the timer
- bool getOvrFlwFlg() checking if the timer value has reached time out or not .. to clear RIS
- void displayTime_1(int s, int m, int h){ to display time LCD

```
void enableTimer(){
    TIMER0_CTL_R |= 0x03;
    return;

}

void disableTimer(){
    TIMER0_CTL_R &= ~(0x01);
    return;
}


bool getOvrFlwFlg(){
    if((TIMER0_RIS_R & 0x1) != 0){
        TIMER0_ICR_R = 0x1;
        return true;
    }
    else
        return false;

}
void displayTime_1(int s, int m, int h){
  LCD_setcursorRowCol(0, 3);
  LCD_printInt(h/10);
  LCD_setcursorRowCol(0, 4);
  LCD_printInt(h%10);
  LCD_setcursorRowCol(0, 5);
  LCD_data(':');
  LCD_setcursorRowCol(0, 6);
  LCD_printInt(m/10);
  LCD_setcursorRowCol(0, 7);
  LCD_printInt(m%10);
  LCD_setcursorRowCol(0, 8);
  LCD_data(':');
  LCD_setcursorRowCol(0, 9);
  LCD_printInt(s/10);
  LCD_setcursorRowCol(0, 10);
  LCD_printInt(s%10);
```

void displaySecFractions() to display the fractions <decimal>
void BlinkLed (void) to blink the led and starting the buzzer when it is stopped

```
void displaySecFractions(){
    // val of current timer with prescaler
    unsigned int Val = TIMER0_TAV_R;
    double x = 0;
    int y = 0;


    x = (Val / 15990540.0);
    LCD_printString(".");


    y = (int)(x * 100000);
    LCD_printInt(y);
}

void BlinkLed (void){
    for(int i = 0; i < 10; ++i) {
        GPIO_PORTF_DATA_R ^= 0x2;
        SysTick_Wait1ms(500);
    }
}
```

The stopwatch function which is the main function of the stopwatch

- set the state to 2 because we are in timer
- we are making a function not alternating between modes
- disable and re Enable to make sure no interrupts already running
- Enables the processor interrupt.
- Enables the specified GPIO interrupts.
- Registers an interrupt handler for a GPIO port
- systick configurations described in the function
-

```
void stopwatch (void)
{
  states_2=2;                                          // s
  mode_flag=false;                                     //we
 states_1 =2;
   states_1=2;
 //disable and re Enable to make sure no interrupts
  MAP_IntDisable(INT_GPIOF);
   MAP_IntMasterDisable();
    MAP_IntEnable(INT_GPIOF);
     IntMasterEnable();
     __asm("CPSIE I");
      GPIOIntEnable(GPIO_PORTF_BASE,GPIO_INT_PIN_4)
       GPIOIntEnable(GPIO_PORTF_BASE,GPIO_INT_PIN_0)
GPIOIntRegister(GPIO_PORTF_BASE,Timer1IntHandler);
  isEnter3=true;
  LCD_command(CLEAR_DISPLAY);
  Configure_SysTick();
    oneShotConfigure();
    // sec ,minute and hour are initially 0
        int sec = 0;
    int min = 0;
    int hour = 0;
    while(1){

            //adjust LCD
```

The while loop in the stopwatch function where all the steps is taken and shown with comments aside

```
while(1){

        //adjust LCD
    LCD_command(CLEAR_DISPLAY);
    displayTime_1(sec, min, hour);
    unsigned int sw1 = GPIO_PORTF_DATA_R & 0x10;
    unsigned int sw2 = GPIO_PORTF_DATA_R & 0x01;
             mode_flag=false;                  //we are in a function not alternating between modes
           char c=keypad_getkey();             //waiting for the start
           if (c=='f'){                        // returned when a sw  pressed and we are in need to transit between modes
            if (states_2==0){                  //Go to calculator
                loading();
                calc();
              }
            else if(states_2==1){              // returned when a sw  pressed and we are in need to transit between modes
            loading();                         //Load and Go to timer
            timer();
                        }
                    }

    else if(c=='1'){                                    //1 pressed to start
        while(1){


            displayTime_1(sec, min, hour);       //display the tome value in the xx:xx format on LCD
            enableTimer();
                                                        //flags for reset and stop conditions
            bool stopped = false;
                        bool reset_flag=false;
            while(getOvrFlwFlg() == false){          // check the timer has timed out

                            char k =keypad_getkey_2();           //similar to keypad() but without a (  while(1){}  )
                            if (k=='f'){
                             if (states_2==0){

                               loading();
                               calc();



                               calc();
                                        }
                              else if(states_2==1){
                                loading();

                                timer();
                                        }
                             }



                        else if(k == '4'){                    //press 4 to stop
                    disableTimer();
                    displaySecFractions();  //****** when stopeed the stopwatch the fractions are shown****(not the case while counting)
                    stopped = true;         // seting the stopped flag ,Because It will be checked below
                    break;
                }
                        else if( k=='.'){                             //the user wants to reset

                            disableTimer();                           //disabe the timer to reset
                            sec=0;
                            min=0;
                            hour=0;
                            reset_flag=true;
                            break;

                        }
            }
            if(stopped){                            //if the user has pessed 4 then we want to stop here by checking the stoppped flag
                BlinkLed();                     //Blinking the red led an beeping when stoppped
                break;
            }
                        if (reset_flag==true)                   // if the reset "*" is pressed do not count ,just break
                        {
                          break;
                        }
                        // increment the stopwatch
            if((++sec) == 60) {
```

```
                    }
                    // increment the stopwatch
            if((++sec) == 60) {
                sec = 0;
                if((++min) == 60){
                    min = 0;
                    hour++;
                }
            }

        }

    }

            else if (c=='.'){               // this condtidion is true when reset pressed before start counting
              disableTimer();
              sec=0;
              min=0;
              hour=0;


            }

    }
```

5. The problems

There are many problems that we have face during working on this enormous project due to the large amount of details that must be perfectly correct just to function. The problems are divided to two section of problems :

- the first one is hardware (circuits and physical problems)
- the second one is in the software (coding)

I. hardware problems (physical)

At First, while trying to configure the hardware wires (jumpers) between the LCD and the TivaC we went through some problem with the connections as we were not sure which GPIO port we will use and which connection type to use with the LCD (the 4-lines LCD display or the 8-line LCD display).

After making all the hardware connections we went into a problem of displaying our code on the LCD .The code was compiled well and LCD was working but it didn't print the code data ,then we found that we need to connect the TivaC ground with the breadboard ground in order to make the LCD dislay the desired code .

The LCD displayed many unwanted number and character at first but then we knew that we should clear the previous lcd display in order to stop the overrides .then we needed to set the cursor on a certain column and row in the LCD .

II. software problems (coding)

- The first issue that we have faced that we have to hold on the button to keep the stopwatch functioning .So we have overcome this problem by starting the stopwatch when we press on #1 on the keypad and stop the stopwatch when we press on #4 on the keypad .

- The second problem is when we use external variables always the output is wrong values so we try to solve this problem by using pointers and it worked very well .

- At first when we start using pointers we have placed it in a wrong place in the memory where the stack overwrite in it like :
  #define states_2 (*((volatile unsigned long *)0x20000FF4))
  #define mode_flag (*((volatile bool *)0x200004F8))
  #define switch_to_timer_flag (*((volatile bool *)0x200002FC))

- So we managed to put them in another location where it can be stable like :
  #define states_2 (*((volatile unsigned long *)0x20007FF4))
  #define mode_flag (*((volatile bool *)0x20007FF8))
  #define switch_to_timer_flag (*((volatile bool *)0x20007FFC))

The Timer code problems

The timer at first did not want to display the code as intended but we then configured a function that did not return any value until a keypad button is pushed .2nd we made a timer and initialized it but not completely as we wanted to activate the timer timeout interrupt only after the user enters the 4 numbers to count down ,only then the flag value will be equal to one and the timer interrupt will be completed with the function TimerEnable-> (TIMER0_BASE, TIMER_A); Then the counter will start counting down. Then the LCD displayed wrong values again ,then we changed the value datatype of the counter value displayed on the LCD screen to a pointer and it worked .we tried the global and static and other values ,but all of the these different datatypes displayed different unwanted values on the LCD screen .
We tried to replace the while(1) infinite loop with another idea of looping with a flag that is = 0 untill the interrupt of the timer happenes then it exits this loop but it didn't work so we made the flag loop in the

while(1) infinite loop then it worked.

At this point we had two interrupts that may work together .the TivaC button interrupt used to switch to the calculator or the stopwatch and the timer timeout interrupt that trigger the timer to start counting down .We used the priority interrupt set function and made the buttons interrupt with a stronger priority but it didn’t work as expected as the next working program either the calculator or the stopwatch displayed error ,then we need to disable the interrupt timer before switching to another program state and it worked fine .

Finally, we wanted to make the buzzer works with PORTD (D7 pin) but it didn’t work so we configured the buzzer on PORTF (Pin 1 REDLED) and it worked .