Embedded Final Project

Team 14

Omar Mohamed 19p4953

Moataz Eldeeb 19p6238

Shaimaa Mohamed 19p7484

Aya Ahmed 19p1689

Rana Mohamed 19p8994

Advanced Calculator

Table of Contents

Introduction

During this course, you studied microcontroller architecture along with basic embedded C programming. You have addressed project

building process, digital I/O, timers, and interrupts applied on Arm Cortex M4 TivaC. Group of course assignments delivered by individuals

are considered as the first corner stone in this project. It is intended to assess all these engineering design aspects along with other skills.

Objectives

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 both be handled separately by the hardware.

By the end of this project, you must master the following:

1. Timers: You will be using at least two timers one for the stopwatch mode and the other for the timer mode.

2. GPIO: You will be using the LCD and a keypad and some push buttons.

3. Interrupts: You will also use interrupts in this project for the push buttons and timers.

Features

In this project you will have 3 main Modes.

Calculator

In this mode user inputs number and op and data is printed on the LCD. User will input two numbers both less than 4 digits and a sign between them.

A button: +

B button: -

C button: /

D button: =

‘*’ button: x

The numbers and the sign are printed on the LCD.

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 time zero it will trigger red led for 1 second.

When the user switches 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 the timer starts as soon as the user presses on the D button on the keypad.

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 the user switches to this mode initially presents 00:00 on the LCD. Whenever the user presses on the start button start incrementing the stopwatch.

Code Implementation

Note: all code snapshot will be available at the end of the document.

Drivers

lcd.h/lcd.c

the lcd is a device used in embedded to display messages or anything that can be written using 16x2 characters to register data inside the lcd the enable must be enabled and disabled with some delays depending on the data sent if it's a command or a data to work properly.

The lcd has two modes command and data which can be changed using the RS pin '0' for command and '1' for data.

In the lcd.h the common commands are written to be easily called.

In the lcd.c the most important functions are:

- the initialization function to initialize the lcd ports and pins and activate the second line.
- display string function which prints all characters in the string sent by the user and automatically goes to the second line if line 1 is full
- clear screen function which automatically clear screen for new messages to be printed
- lcd command which process the lcd command declared in the lcd.h to work as intended

keypad.h/keypad.c

To reduce the microcontroller, I/O pin usage, keyboards are organized in a matrix of rows and columns. The CPU accesses both rows and columns through ports, when a key is pressed, a row and a column make a contact; otherwise, there is no connection between rows and columns.

In keypad.h the port number and pins are defined

In keypad.c there is 2 functions:

- keypad initialization where the ports and inputs are configured
- keypad read which returns the value of the button pressed and if none is pressed return 'T' as the error defined manually by us so the return value could be ignored and prevent any type of polling in the project

DIO.h/DIO.c

This driver's importance is to initialize ports easily and neatly as possible.

Using base addresses, the initialization function offsets the address to input data in the right registers easily.

Types.h

Hold all the defined types of data like unsigned integers and characters long and all

main.h

In main.h all includes from all the drivers is written and some function prototypes to avoid any problems during the runtime and compiling.

Main code

Timer_LCD/Stop_LCD

Both of these function is called whenever the program is supposed to show the mode on the screen .

Timer_Display/Stop_Display

This functions are called every interrupt to update the global variables of the seconds of each mode to either increment for stopwatch or decrement in timer mode except in timer mode when the timer reaches zero the red LED on tivaC is lit for 1 second then closed before disabling the GPIO_Timer.

Toggle_LED

Toggles the red led when called

Timer_init

Initialize ,configure,and load the number of ticks to the registers and register the function that will be called every interrupt .

Timers used in this project:

- GPIO_TIMER1 for the Timer mode
- SYSTICK timer for the stopwatch

Each is configured to interrupt every 1 second

Timer_enable

Enable timer's interrupts and the master interrupts at the beginning of the run

TivaIntButtons

Function that configure the GPIO_INT for the Push button connected to PORTF PIN4 in order to increment the mode number to change the mode of the calculator using the mode function.

Change_mode

Increment the mode variable to change the mode

Modes

Note: to be able to switch from one mode to another easily without any problems no polling was made in the project and by checking if the mode is changed in every while cycle the project changes smoothly from one mode to another smoothly

In every mode the code is divided into phases by using counters in case of the stopwatch and the timer modes the counter is global to return to the same phase without resetting anything unless the reset button is pressed which is "#" in the keypad.

Digit checking is done by the isdigit c function.

If checking for multiple characters, the functions strchr which returns null if the character is not found.

As for the Stop watch and the timer modes the minutes are calculated from the total second using remainders making it more easy to calculate without the need of million of ifs to calculate every number separately .

Calculator mode

Phase 1

User must input at least one integer maximum 4 before pressing any operation button to go the next phase.

Phase 2

User must input one integer maximum 4 to be able to press the “=” on the keypad to calculate and print it on the lcd

Phase 3

The two integers which are stored in char array are turned into integers then the operation picked by user is used on them to output the result on the lcd in case of division the float numbers are turned into integers and incase the dividor is zero a error message is printed

Phase 4

The calculator is waiting for the user to press the “#” to reset the calculator.

Stopwatch mode

Phase 1

The stop watch wait for either a * or # to start(enable timer) or pause(disable timer) or reset the stopwatch

Timer mode

Phase 1

The user input from right to left the time of the timer minute 1 then minute 2 then second 1 then second 2 all while updating the lcd

Phase 2

The Timer wait for either a * or # to start(enable timer) or pause(disable timer) or reset the timer

Lcd.h

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "types.h"
#include "DIO.h"
#include "tm4c123gh6pm.h"
//commands for the LCD from the datasheet
typedef enum
{
  CLEAR_DISPLAY = 0x01,
  RETURN_HOME = 0x02,
  CURSOR_LEFT = 0x04,
  SHIFT_RIGHT = 0x05,
  CURSOR_RIGHT = 0x06,
  SHIFT_LEFT = 0x07,
  Display_Off_Cursor_Off=0x08,
  LCD_ON_CURSOR_ON = 0x0F,
  Display_On_Cursor_Off=0x0C,
  Display_On_Cursor_Blinking_Off=0x0E,
  Display_On_Cursor_Blinking_On=0x0F,
  FORCE_AT_BEGIN = 0x80,
  TWO_LINES5x7 = 0x38,
  ACTIVATE_SECOND_LINE = 0x3C,
  Begin_At_First=0x80,
  Begin_At_Second=0xC0
} LCD_COMMANDS;
// for data from d0 to d7 in the lcd display
#define DataPortBase PORTBBase
//mode and enable
#define ModePortBase PORTABase
// for data from d0 to d7 in the lcd display
#define DataPort DataPortBase+0x3fc
//mode and enable
#define ModePort ModePortBase+0x3fc
//function to initialize the lcd
void LCD_init();
//function to excute commands
void LCD_Command(LCD_COMMANDS Command);
//function to print one character
void LCD_Data(uint8 Data);
//function used to print strings by sending one character by time to print one after another after 15 char the lcd goes to the second line auto
void LCD_Displaystring(const char *Str);
//to clear the display
void LCD_clear_screen(void);
```

Lcd.c

```c
#include "lcd.h"


int d=50;
void delay(int delay)
{

  for (int i =0;i<delay;i++)
    for(int j=0;j<delay;j++);

}

void LCD_init()
{
  SYSCTL_RCGCGPIO_R |= 0x03;
  DIO_Init_port(DataPortBase, 0xFF, 0xFF, 0x00, 0x00);
  DIO_Init_port(ModePortBase, 0xFF, 0xFF, 0x00, 0x00);
  LCD_Command(Display_On_Cursor_Off);
  LCD_clear_screen();
  LCD_Command(ACTIVATE_SECOND_LINE);
  LCD_Command(Display_On_Cursor_Blinking_Off);
}

void LCD_Command(LCD_COMMANDS Command)
{
  delay(d);
  DIO_WritePin(ModePort, 7, LOGIC_LOW);
  delay(d);
  DIO_WritePin(ModePort, 6, LOGIC_HIGH);
  delay(d);
  DIO_WritePort(DataPort, Command);
  delay(d);
  DIO_WritePin(ModePort, 6, LOGIC_LOW);
}
void LCD_Data(uint8 Data)
{
  delay(d);
  DIO_WritePin(ModePort, 7, LOGIC_HIGH);  // see here
  delay(d);
  DIO_WritePin(ModePort, 6, LOGIC_HIGH);
  delay(d);
  DIO_WritePort(DataPort, Data);
  delay(d);
  DIO_WritePin(ModePort, 6, LOGIC_LOW);
}

void LCD_Displaystring(const char *Str)
{
  uint8 i = 0;
  while(Str[i] != '\0')
  {
    if(i==16)
      LCD_Command(Begin_At_Second);
    LCD_Data(Str[i]);
    i++;
  }

}
void LCD_Intgertostring(int Data,char *string)
{
  snprintf(string, 16, "%d", Data);
}

void LCD_clear_screen(void){
  LCD_Command(CLEAR_DISPLAY);

}

```

Keypad.h

```c
#ifndef KEYPAD_H_
#define KEYPAD_H_
#include "DIO.h"
#include "tm4c123gh6pm.h"
#include "bitwise_operation.h"
#include "DIO.h"

#define Col0    4
#define Col1    5
#define Col2    6
#define Col3    7
#define PORTCOL PORTCBase

#define Row0    0
#define Row1    1
#define Row2    2
#define Row3    3
#define PORTROW PORTEBase

//initialize the keypad using the keypad.h configuration
void KEYPAD_init();

//return values if any keys is pressed else return 't' as error character
uint8 KEYPAD_READ();



#endif
```

Keypad.c

```
#include "KEYPAD.h"
uint8 array1[4][4]={{'1','2','3','+'},
{'4','5','6','-'},
{'7','8','9','/'},
{'*','0','#','='}};
void KEYPAD_init()
{
  SYSCTL_RCGCGPIO_R|=0x14;
  DIO_Init_port(PORTCOL,0xF0,0x00,0xF0,0x00);
  DIO_Init_port(PORTROW,0x0F,0x0F,0x00,0x00);
}
void delay3(int delay)
{

  for (int i = 0;i<delay;i++)
    for(int j = 0;j<delay;j++);
}

uint8 KEYPAD_READ()
{
  for(int i =0;i<4;i++)
  {
    (*((volatile unsigned long *)(PORTROW + 0x3FCU))) = ~(1<<i);
    for (int j=0;j<4;j++)
    {
      if(BIT_IS_CLEAR(GPIO_PORTC_DATA_R,j+4))
      {
        delay3(100);
        while(BIT_IS_CLEAR(GPIO_PORTC_DATA_R,j+4));
        return array1[i][j];
      }
    }

  }
  return 'T';
}

```

DIO.c

```c
#include "DIO.h"
void DIO_Init_port (uint32 portbase,uint8 cr,uint8 pd,uint8 pur,uint8 pdr)
{
  //LOCK key
  (*((volatile unsigned long *)(portbase+0x520U))) = GPIO_LOCK_KEY;
  //commit registry
  (*((volatile unsigned long *)(portbase+0x524U))) = cr;
  //port direction
  (*((volatile unsigned long *)(portbase+0x400U))) = pd;
  //pull up resistor
  (*((volatile unsigned long *)(portbase+0x510U))) = pur;
  //pull down resistor
  (*((volatile unsigned long *)(portbase+0x514U))) = pdr;
  //digital enabler
  (*((volatile unsigned long *)(portbase+0x51CU)))= cr;
}


void DIO_WritePort(uint32 portbase, uint8 value)
{
  (*((volatile unsigned long *)(portbase)))=value;
}

void DIO_WritePin (uint32 port_num, uint8 pin_num, uint8 value)
{
  if (value == LOGIC_HIGH)
    SET_BIT((*((volatile unsigned long *)(port_num))),pin_num);
  else
    CLEAR_BIT((*((volatile unsigned long *)(port_num))),pin_num);
}


uint8 DIO_ReadPin (uint32 port_num, uint8 pin_num)
{
  return GET_BIT((*((volatile unsigned long *)(port_num))),pin_num);

}

uint8 DIO_ReadPort (uint32 port_num)
{
  return (*((volatile unsigned long *)(port_num)));

}

```

DIO.h

```c


#ifndef DIO_H_
#define DIO_H_

#include "types.h"
#include "DIO.h"
#include "tm4c123gh6pm.h"
#include "bitwise_operation.h"
/*******************************************************************************
*                                Definitions                                   *
*******************************************************************************/
#define PORTABase                   0x40004000
#define PORTBBase                       0x40005000
#define PORTCBase                   0x40006000
#define PORTDBase                   0x40007000
#define PORTEBase           0x40024000
#define PORTFBase           0x40025000




#define PIN0                    0
#define PIN1                    1
#define PIN2                    2
#define PIN3                    3
#define PIN4                4
#define PIN5                    5
#define PIN6                6
#define PIN7                    7



typedef enum
{
    PIN_INPUT,PIN_OUTPUT
}GPIO_PinDirectionType;

void DIO_Init_port (uint32 portbase,uint8 cr,uint8 pd,uint8 pur,uint8 pdr);
void DIO_WritePort(uint32 portbase, uint8 value);
void DIO_WritePin (uint32 port_num, uint8 pin_num, uint8 value);
uint8 DIO_ReadPin (uint32 port_num, uint8 pin_num);
uint8 DIO_ReadPort (uint32 port_num);

#endif /* DIO_H_ */
```