INTRODUCTION TO EMBEDDED SYSTEMS

PROJECT

TEAM 5

Ahmed Gamil Fathy	19P4664
Mohammed Ehab Mansour	19P5241
Ahmed Osama Noaman	19P7926
Maria Mourad Elia	19P4894

CODE AND VIDEO:

https://drive.google.com/drive/folders/12MzTiXGbv2XDMzka49Tw6IUkjlMPRO2s?usp=share_link

CONTENTS

INTRODUCTION

In this project, we have 3 main Modes: Calculator, Timer, and Stopwatch. These 3 modes are switched using a push button.

Our first mode is Calculator. The program takes inputs from the user using a keypad and prints each of the user's input on a LCD. The user writes two numbers each at least 1 digit and a sign between these numbers. To finish solve the equation, the user is required to press on the D button of the keypad which corresponds to '=' character for the program to start solving the equation. If any incorrect inputs is given to the program, the result of the equation will always be 0. The equation is printed on the LCD before printing the result.

Our second mode is Timer. 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 the user switches to this mode initially, the LCD will print 00:00, then the program will take the input from the user as minutes and seconds. The input is written as minutes and seconds. The timer will start as soon as the user presses on the D button on the keypad.

Our third mode is Stopwatch. 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, the LCD will print 00:00. Whenever the user presses on the start button, the time will start incrementing.

This project includes: TIVA C, Keypad, LCD, Potentiometer, Buzzer, 4 Push buttons, Resistors, and Jumpers.

This project uses Tivaware libraries to facilitate some coding.

FILES

MAIN.C

This File is our main where the program starts running from. In this file, we initialize all needed modules such as the Keypad, LCD, Buttons, and timers. Then after initializing these modules, the program runs in a while(1) loop where the program keeps running endlessly. In the while loop, a switch case is used to determine what mode are we going to run. When a case is decided, the program starts by clearing the LCD (in case the a string is already printed on the LCD), then printing the mode's name such as calculator on the LCD. A delay is given then we clear the mode's name to start executing the function's mode. Such functions are on other files which we will see later.

KEYPAD_DRIVER.C

This file has two functions:

- void keypad.init()
- void keypad.getkey()

- Keypad.init function is used to initialize all needed ports and pins from the tiva c for the connection of the keypad. For the keypad, we use pins in ports A and E. The pins in portA are all used as outputs which are connected to the columns of the keypad. While portE are all used as inputs (Pull down resistors) for the rows of the keypad.

- Keypad_getkey function returns the value of a key have been pressed at the moment the function had been called. It simply makes 2 loops one to find the column which contains the button had been pressed and another loop to determine the row of that button. Having these data the we could determine a unique value for each button.

KEYPAD_DRIVER.H

This file is used to define some identifiers such as padRows and padColumns. It also includes libraries which will be discussed later.

LCD_DRIVER.C

This File has 8 functions:

- void LCD4bits_Init()
- void LCD_Write4bits(char data, char control)
- void LCD_WriteString(char *str)
- void LCD_WriteChar(char c)
- void LCD4bits_Cmd(unsigned char command)
- void LCD4bits_Data(unsigned char data)
- void LCD_Clear(void)
- void LCD_ClearT(void)

- LCD4bits_Init() is used to initialize portB that will be used to connect the LCD.
- LCD_Write4bits(char data, char control) writes the data to LCD and this data is specified using the control which takes either command or data for the LCD to execute.
- LCD_WriteString(char *str) is used to print on the LCD a string.

- LCD_WriteChar(char c) is used to print only a character.
- LCD4bits_Cmd(unsigned char command) is used to write a command to the LCD.
- LCD4bits_Data(unsigned char data) is used to write data to the LCD.
- LCD_Clear(void) is used to clear the LCD and move the cursor at the beginning of the line then delay 500 ms.
- LCD_ClearT(void) is the same as the previous but with a delay of 1 ms.

LCD_DRIVER.H

This file includes all libraries needed for the LCD_Driver.c file.

BUTTONINTERRUPTS.C

This File has 6 functions and 2 global variables (Boolean go, Boolean reset):

- bool resumed(void)
- bool is_reset(void)
- void set_and_leave(void)
- void resume_pause(void)
- void next_state(void)
- void buttonInterrupts_Init(void)

- resumed(void) is used to return the flag go.
- is_reset(void) is used to return the flag reset.
- set_and_leave(void) is used to reset all flags to false.
- resume_pause(void) is the ISR for the start, pause, and reset buttons which sets the go flag in case of start, clears the go flag in case of pause, and sets the reset flag in case of reset.
- next_state(void) is the ISR for the mode switch. It switches to the next mode everytime the button is pressed.

- buttonInterrupts_Init(void) initializes portF for mode switch button and portD for start, pause, and reset buttons. It also configures the interrupts for each button where all of them are falling edge type.

BUTTONINTERRUPTS.H

- This file includes “includes.h”, “state.h” and has prototypes for the above functions.

STATE.C

This file includes two variables (int states[] = {calculator, timer, stopwatch}, int state_count = calculator) and 3 functions:

- Int getCurrentState(void)
- Void incrementState(void)
- Void setCurrentState(int st)

- getCurrentState(void) returns the current mode if the program
- incrementState(void) increments the mode for to switch
- setCurrentState(int st) can be used to set the state with brute force

STATE.H

This file includes defines such as calculator 0, timer 1, and stopwatch 2. It also has prototypes for the above functions.

DELAY.C

This file includes 2 functions:

- void delayMs(int n)
- void delayUs(int n)
- delayMs(int n) takes n in milliseconds.

- delayUs(int n) takes n in microseconds.

DELAY.H

This file includes prototypes for the functions mentioned above.

CALCULATOR.C

This File contains 4 functions:

- bool isDigit(char d)
- void mystrcat(char eq[], char key[])
- void solve(char eq[], char sol[])
- void calc(void)

- isDigit(char d) returns true if the char is a digit
- mystrcat(char eq[], char key[]) appends string key to string eq
- solve(char eq[], char sol[]) solves the eq string then appends sol string with the solution
- calc(void) is the main function that calls all the previous functions and prints on the LCD accordingly

CALCULATOR.H

This file includes: "state.h", "LCD_driver.h", "keypad_driver.h", "delay.h", <string.h>, "includes.h". It also implements all functions presented above.

TIMER.C

This file includes 8 functions:

- void show_msg(char c)
- void time_cat(char min[], char sec[], char time[])
- void timer_init(void)
- int str_to_int(char s[])
- bool timeOut(char min[], char sec[])
- void set_newTime(char min[], char sec[])
- void ring_buzz()
- void tim(void)

- show_msg(char c) shows the message of either ‘m’ (min message) or ‘s’ (second message)
- time_cat(char min[], char sec[], char time[]) concatenates strings min and sec to a new string timer
- str_to_int(char s[]) casting string to int
- timeOut(char min[], char sec[]) returns true if timer is 00:00
- set_newTime(char min[], char sec[]) decrements the timer each second
- ring_buzz() rings the buzzer once timeout
- timer_init(void) initializes the timer module to use for the timer mode
- tim(void) is the main function that calls all the functions above and manages the LCD accordingly

TIMER.H

This file includes: "state.h", "LCD_driver.h", "keypad_driver.h", "delay.h", <string.h>, "includes.h". it also has prototypes for all functions used above.

STOPWATCH.C

This file has 3 functions:

- void timer_st_init(void)
- void increment_time(char min[], char sec[])
- void stw(void)

- timer_st_init(void) initialized the timer used for stopwatch mode
- increment_time(char min[], char sec[]) increments the time in stopwatch each second
- stw(void) is the main function for stopwatch which uses all functions above and manages the LCD accordingly.

STOPWATCH.H

This file includes: <string.h>, "includes.h", "state.h", "LCD_driver.h", "keypad_driver.h", "timer.h", "buttonInterrupts.h", "delay.h". It also has prototypes for each function used above.

INCLUDES.H

This file includes:

#include "tm4c123gh6pm.h"

#include <stdint.h>

#include <stdbool.h>

#include "stdio.h"

#include "inc/hw_types.h"

#include "driverlib/sysctl.h"

#include "driverlib/systick.h"

#include "driverlib/timer.h"

#include "driverlib/gpio.h"

#include "driverlib/interrupt.h"

#include "inc/hw_memmap.h"

#include "driverlib/pin_map.h"

#include "driverlib/rom_map.h"

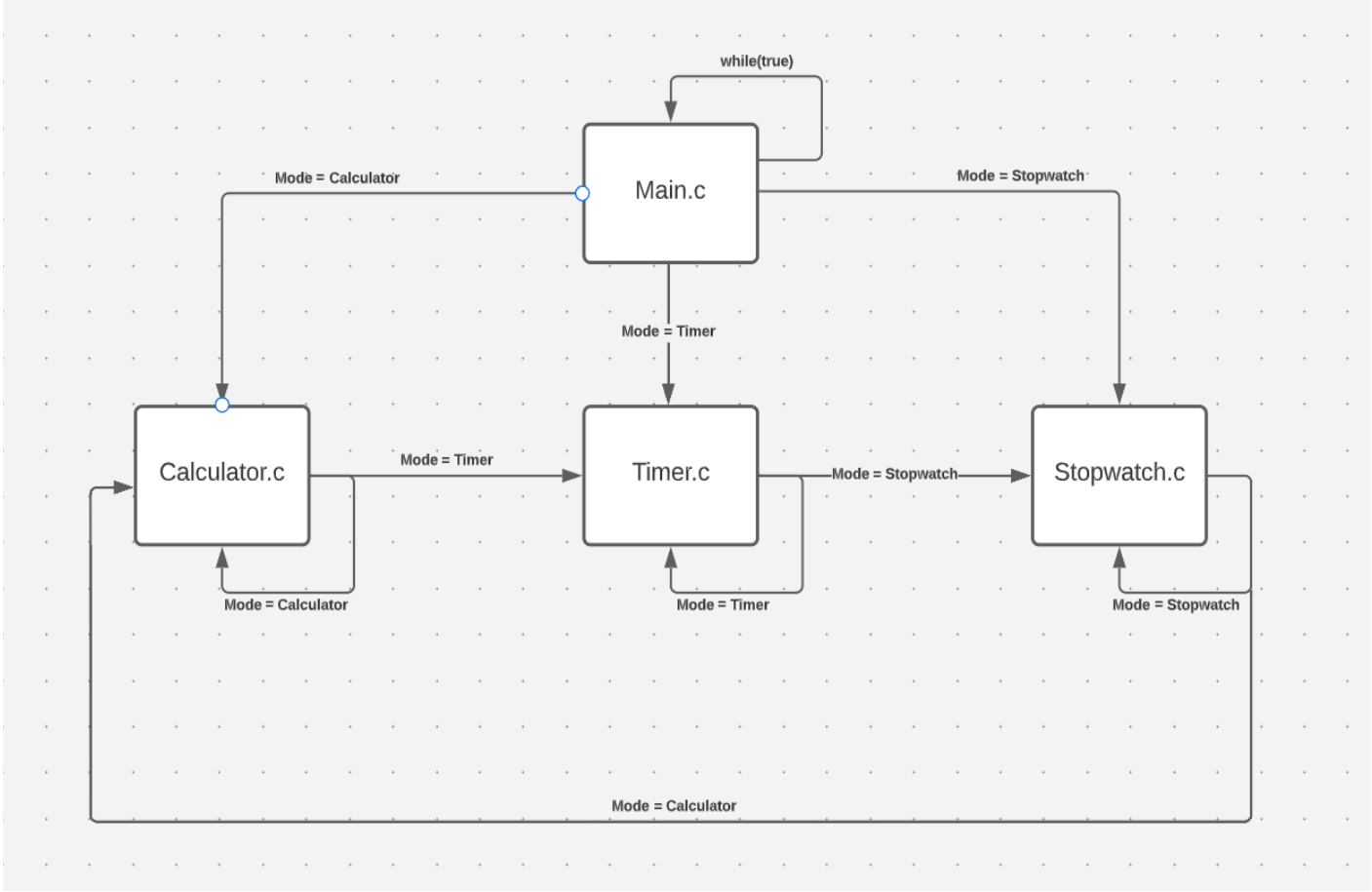
#include "driverlib/rom.h"

#include "inc/hw_gpio.h"

TM4C123GH6PM.H

This file is also included in our project to facilitate coding using macros.

FLOW DIAGRAM



USER GUIDE

- The mode is initially set to Calculator. So as soon as the program starts to run, a “Calculator” string will be printed on the LCD. Then the LCD clears out and waits for inputs from user. The User starts pressing from the keypad with each printed on the LCD to form an equation with two operands and one operation from the four. As soon as the user pressed D “=”, the LCD will clear out to print out the solution to the equation. The LCD will wait for any input from the user to clear out the solution and loop once again to calculator mode if the mode the button has been pressed.

- The User can switch to the next mode at any given time unless if the program is executing a keypad function which the user needs to give an input for the mode to be switched

- If Button mode is pressed, the mode will increment to its next state which is the Timer. The timer mode starts by printing “Timer” string on the LCD then clears out and starts executing the tim function. The LCD will initially print “00:00” until any input keypad is pressed to clear the LCD. Then, a message will be printed on the LCD for the user to write the minutes needed first. Then, after writing 2 digits for the minutes, the user needs to press any key to proceed to the second message which the user now needs to write 2 digits for seconds. After that, the user needs to press any button to proceed. The timer in minutes and seconds will be printed in the form of “00:00”. The user needs to press D to finally start the timer. When timeout, a buzzer will ring.

- The User can switch to the next mode at any given time except when an input keypad is required.

- If Button mode switch is pressed, the program will run the stopwatch mode. In this mode, the LCD will print out “Stopwatch”

initially then clears out and finally print “00:00”. The user can start the stopwatch by pressing the start button where the time will start incrementing each second. The user can pause the stopwatch by pressing the pause button where the LCD will freeze at the time last printed. The user can also reset the LCD during either the stopwatch is resumed or paused. The reset will start executing the whole stopwatch function again which always initially prints “00:00”.

- The User can switch to Calculator mode by pressing the button mode at any giver time except when the stopwatch is paused.