Universidad de Puerto Rico
Recinto de Mayaguez
Departamento de Ingenieria Electrica y Computadoras.
ICOM5217 - Interconexión de Microprocesadores

Experimento #2 - Reporte

Interrupts and Switch Debouncing

Juan Lebrón Betancourt
Anthony Llanos Velázquez
Profesor: Manuel Jimenez
Instructor: Luis Malavé
30 de septiembre de 2013

Exercise 1 and 2: Hardware and Software Debounce

The circuit used for hardware debouncing was mounted, using a capacitor and the Schmitt Triggered GPIO ports of the Tiva Microcontroller. In the Interrupt Handler Vector Table, a function named “switchPressed” was assigned to the GPIO Port E of the microcontroller and the following code was written in its declaration:

```
void switchPressed(){

	count ++; //Increase count.
	LcdCommandWrite(0x80); //Go to first line of LCD.
	writeLetter(count+48); //Write the number.
	IntFinish(); //Reset the interrupt status of port.
	return;

}
```

For Exercise 2, the button only needed to be connected using a resistor and a delay was added to the code.

```
void switchPressed(){

	count ++; //Increase count.
	LcdCommandWrite(0x80); //Go to first line of LCD.
	writeLetter(count+48); //Write the number.
	SysCtlDelay(300000); //Software delay.
	IntFinish(); //Reset the interrupt status of port.
	return;

}
```

Complete code is written below in the homework section.

Exercise:

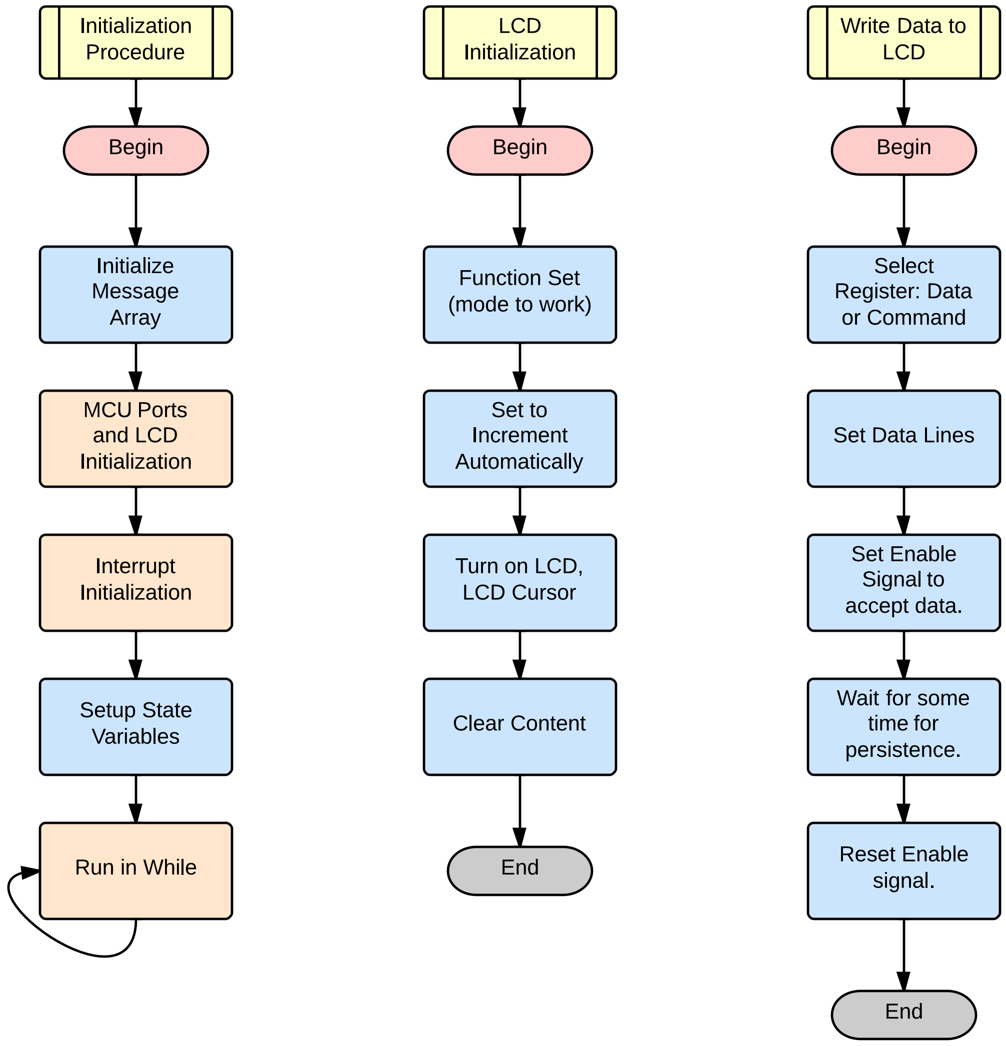
Homework

•

Code

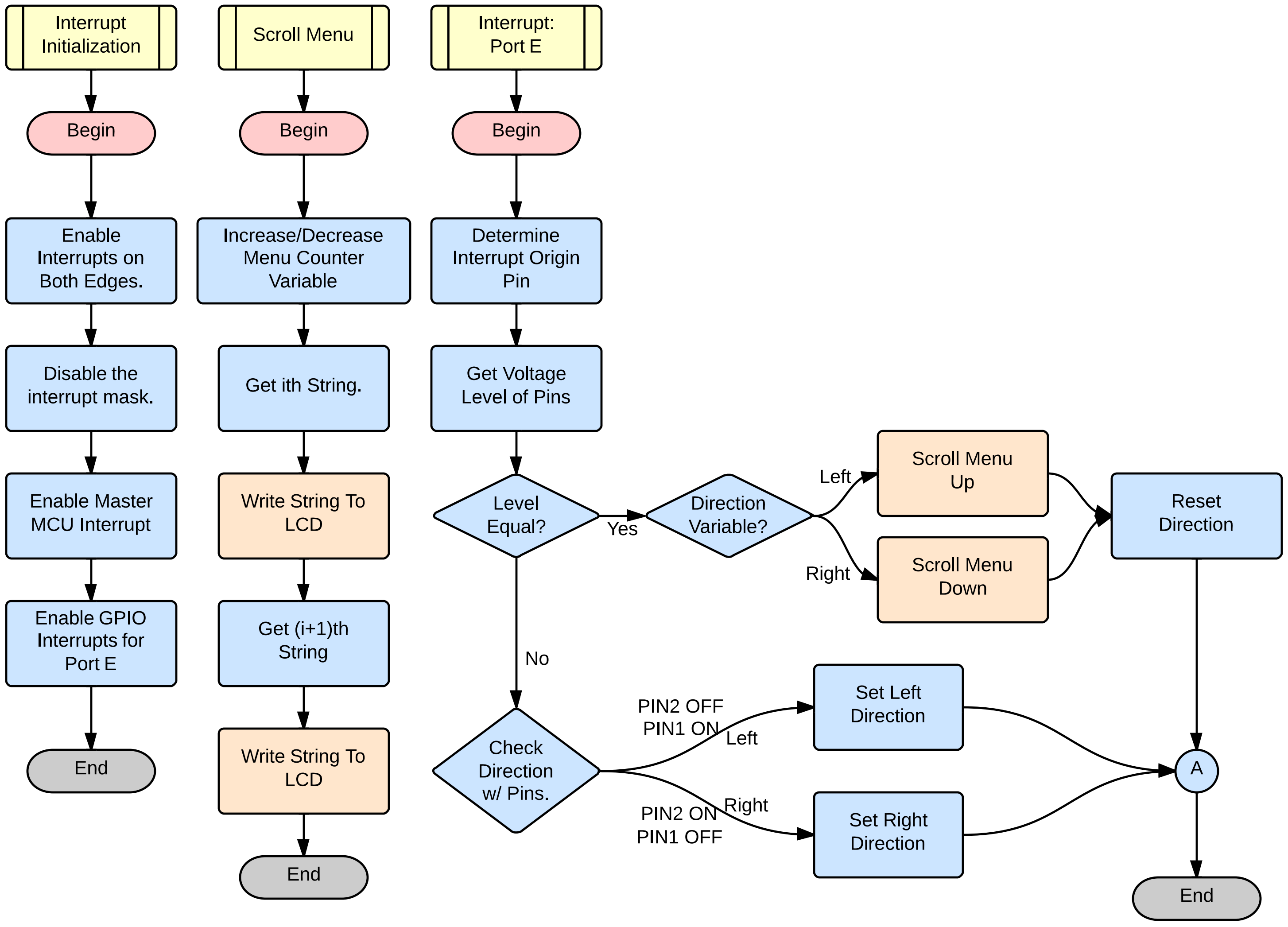
Software Plan - Experiment 2 - Homework

Juan Lebron Betancourt
Anthony Llanos Velazquez



Software Plan - Experiment 2 - Homework

Juan Lebron Betancourt
Anthony Llanos Velazquez



helloLCD.c Interrupts - Self-Made Libraries Omitted (lcd.h, gpio.h)

```c
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"
#include <stdbool.h>
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_nvic.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/fpu.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom.h"
#include "driverlib/sysctl.h"
#include "driverlib/systick.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"

//*******************************************
//Self-Made Libraries for easy-interfacing.
#include "gpio.h"
#include "lcd.h"

const int MESSAGE_SET_SIZE = 17;
char* c[] = { "Experiment 2    .", "LCD Interface   .", "with Tiva MCU   .",
            "Use the wheel   .", "to scroll up and.", "down through the.",
            "menu, thanks! :D.",
            "Hello World!    .", "Hello Anthony!  .", "Hello Juan!     .",
            "AeroBal Micro 2 .", "Hakuna Matata   .", "If this doesn't .",
            "work, it was    .", "Anthony's fault .", ",if it works,   .",
            "Juan did it! :) ." };
void nextString(int i);
void menuUp(int i);
void menuDown(int i);
void IntFinish();
void IntMaskEnable();
void interruptInit();
void switchPressed();

int cursor = 0;
int direction = 0;

void writeString(char* string){
      int i;
      for(i = 0; string[i] != '.' ;i++){
             lcdWriteData(string[i]);
      }
}

void nextString(int i){
      lcdCursorHome(); //First line.
      writeString(c[(i%MESSAGE_SET_SIZE)]); //Write message i.
      SysCtlDelay(200000);
      lcdCursorHomeDown(); //Second line.
      writeString(c[(i+1)%MESSAGE_SET_SIZE]); //Write message i+1.
      SysCtlDelay(200000);
```

```
}

//Stub for modularization. Interrupt calls. Up.
void menuUp(int i){
      cursor-- ;
      cursor = (cursor < 0) ? MESSAGE_SET_SIZE-1 : cursor;
      nextString(i);
}
//Stub for modularization. Interrupt calls. Down.
void menuDown(int i){
      cursor = (cursor+1)%MESSAGE_SET_SIZE;
      nextString(i);
}
void writeLetter(char letter){
      lcdWriteData(letter);
}
//Return Interrupt back to normal on PortE, pin 1 and 2.
void IntFinish(){
      HWREG(0x4002441C) = 0x06 ;
}
//Disable masking on Port E, pin 1 and 2.
void IntMaskEnable(){
      HWREG(0x40024410) = 0x06; //Activating Port B
      //_asm("MOV R1, R2");
}
//Procedure to initiate the interrupt framework.
void initInterruptModule(){
       IntMaskEnable(); //Disable masking procedure.
       HWREG(0x40024408)= 0xFF; //Enable edge interrupts for both edges.
       IntMasterEnable(); //Enable interrupts on controller.
       IntEnable(INT_GPIOE); //Enable interrupts on port E.
}
//**************************************
//Interrupt Handler
void switchPressed(){

      uint32_t ris = HWREG(0x40024414); //Interrupt status of ports.
      uint32_t data = HWREG(0x400243FC); //Port Low or High Level.
      data = data & 0x06; //Only get levels of pin 1 and 2.

      if((data & 0x04) && (data & 0x02)){ //Levels equal?
            if(direction < 0 && (ris & 0x04)){ //Left direction.
                  //Move up in circular array.
                  menuUp(cursor);
                  SysCtlDelay(300000);
            }
            else if(direction > 0 && (ris & 0x02)){ //Right direction.
                  //Move down in circular array.
                  menuDown(cursor);
                  SysCtlDelay(300000);
            }
            //Reset direction variable.
            direction = 0;
      }
      //Set right direction.
      else if(ris & 0x04){
            direction = 1 ;
```

```
        }
        //Moving left direction.
        else if(ris & 0x02){
              direction = -1 ;
        }
        IntFinish();
        return;
}

int main(void){

        // Enable the GPIO ports that are used for the on-board LED.
        portActivate(GPIO_PORTA);
        portActivate(GPIO_PORTC);
        portActivate(GPIO_PORTE);
        portActivate(GPIO_PORTD);

        //Set Direction for each register port.
        portDirection(GPIO_PORTA, 0x0C);
        portDirection(GPIO_PORTC, 0xF0);
        portDirection(GPIO_PORTD, 0x0F);
        portDirection(GPIO_PORTE, 0x00);

        //Digital Enable.
        portDigitalEnable(GPIO_PORTA,0x0C);
        portDigitalEnable(GPIO_PORTC,0xF0);
        portDigitalEnable(GPIO_PORTD,0x0F);
        portDigitalEnable(GPIO_PORTE,0xFF);

        //Write Commands to Initialize LCD.
        lcdInit(GPIO_PORTA|GPIO_OFFSET_DATA,
                    GPIO_PORTC|GPIO_OFFSET_DATA,
                    GPIO_PORTD|GPIO_OFFSET_DATA);

        cursor = 0 ;
        nextString(cursor);

        initInterruptModule();
        while(1);

}
```