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

Experimento #1 - Reporte

IDE, ASM/C Programming & IO

Juan Lebrón Betancourt
Anthony Llanos Velázquez

Profesor Manuel Jimenez
Instructor: Luis Malave

24 de septiembre de 2013

Exercise:

Polling a Switch

•

Code

```c
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"

int main(void){

    volatile uint32_t ui32Loop;
    int ledStatus = 0;

    // Enable the GPIO port that is used for the on-board LED.
    SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOF;
    SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOD;

    // From Tiva Tutorial.
    // Do a dummy read to insert a few cycles after enabling the peripheral.
    ui32Loop = SYSCTL_RCGC2_R;

    // Enable the GPIO pins for the LED (PF1,PF2,PF3,PD1).
    // Set the direction as output, and
    // enable the GPIO pin for digital function(PD0).
    GPIO_PORTF_DIR_R = 0x0E;
    GPIO_PORTF_DEN_R = 0x0E;
    GPIO_PORTD_DIR_R = 0x02;
    GPIO_PORTD_DEN_R = 0x03;

    // Loop forever.
    int wasPressed = 0;
    while(1){
      //Turn on LED when off.
      if(((GPIO_PORTD_DATA_R & 0x01) == 0x01) && !ledStatus && !wasPressed)
      {
            GPIO_PORTF_DATA_R = 0x0E;     //Turn On onboard leds
            GPIO_PORTD_DATA_R |= 0x02;    //Turn On external led
            ledStatus = 1;
            wasPressed = 1;
            SysCtlDelay(9000000);

      }
      //Turn off LED when on.
      else if(((GPIO_PORTD_DATA_R & 0x01) == 0x01)
                        && ledStatus && !wasPressed){
            GPIO_PORTF_DATA_R = 0x00;                 //Turn OFF onboard leds
            GPIO_PORTD_DATA_R &= 0xFD;                //Turn OFF external led
            ledStatus = 0;
            wasPressed = 1;
            SysCtlDelay(9000000);
      }
      //To help eliminate debouncing.
      else if(!(GPIO_PORTD_DATA_R & 0x01)){
            wasPressed = 0 ;
      }
    }
}
```

Homework:

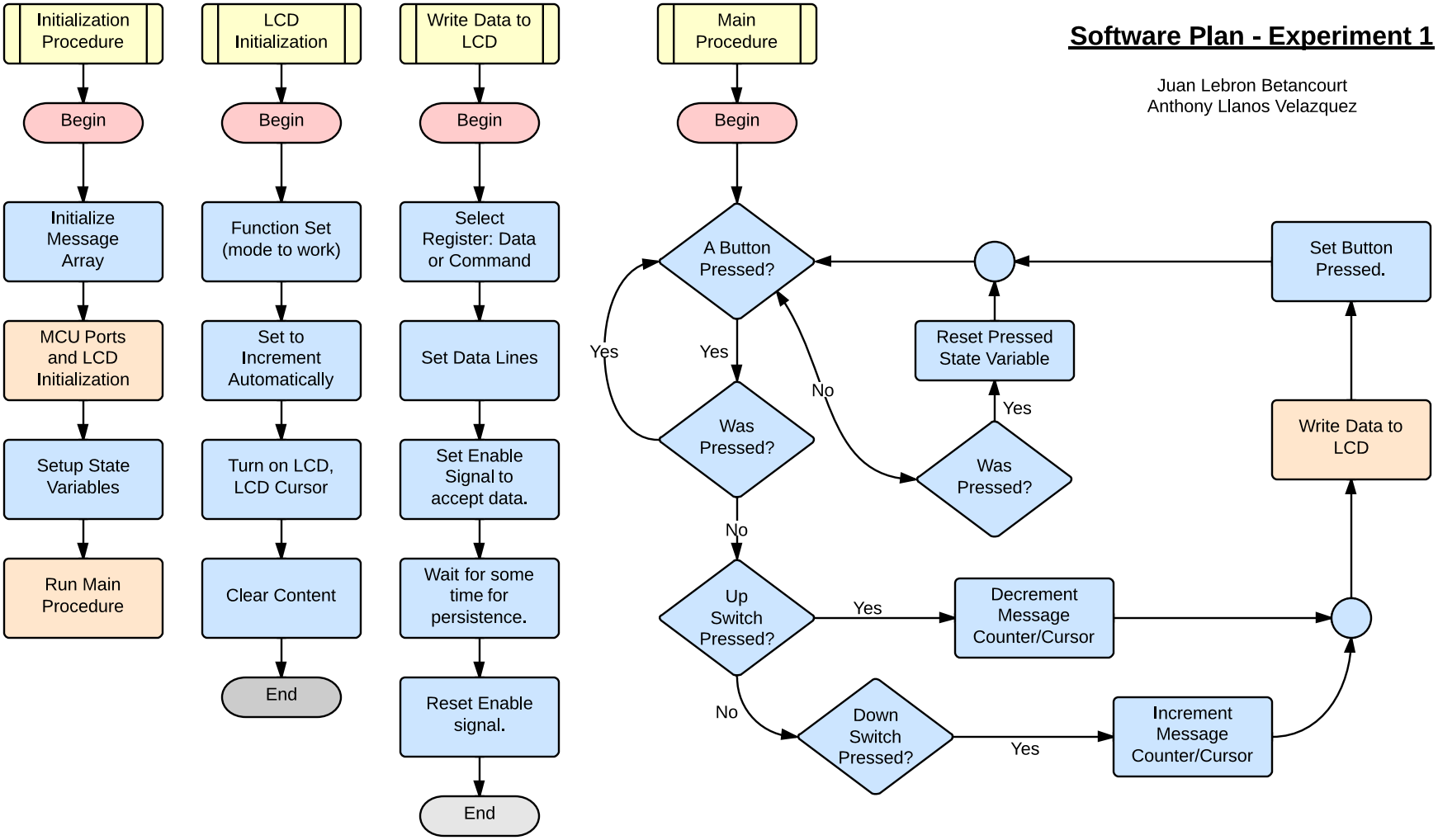
LCD Scroll List

•

Software Plan, Code

Software Plan - Experiment 1

Juan Lebron Betancourt
Anthony Llanos Velazquez



```c
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"

typedef int bool ;

void writeLetter(char letter);
void writeString(char* string);
void LcdCommandWrite(int command);
void LcdDataWrite(char letter);
void wait(uint32_t amount);
void writeDataToPort(int data);
void init();
void nextString(int i);
void menuUp(int i);
void menuDown(int i);
void delay(uint32_t amount);

bool isPressedUp();
bool isPressedDown();

const int MESSAGE_SET_SIZE = 17;
char* c[] = { "Experiment 1    .", "LCD Interface   .", "with Tiva MCU  .",
              "Use the buttons .","to scroll up and.", "down through the.",
              "menu            .",
              "Hello World!    .", "Hello Anthony!  .", "Hello Juan!     .",
              "AeroBal Micro 2 .", "Hakuna Matata   .", "If this doesn't .",
              "work it was     .", "Anthony's fault .", ",if it work      .",
              "Juan did it! :) ." };


int main(void){

      volatile uint32_t ui32Loop;

      // There is a control register that enables interfacing with the
registers
      // Enable the GPIO ports that are used for the on-board LED.
      SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOA;
      SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOC;
      SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOE;
      SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOD;

      // Using Tiva example:
      // Do a dummy read to insert a few cycles after
      // enabling the peripheral.
      ui32Loop = SYSCTL_RCGC2_R;

      //DIR - Direction.
      //Set Direction for each register port.
      GPIO_PORTA_DIR_R = 0x0C;
      GPIO_PORTE_DIR_R = 0x00;
      GPIO_PORTC_DIR_R = 0xF0;
      GPIO_PORTD_DIR_R = 0x0F;

      //DEN - Digital Enable.
      //Enable digital interface with
```

```
	//these particular register ports.
	GPIO_PORTA_DEN_R = 0x0C;
	GPIO_PORTE_DEN_R = 0x06;
	GPIO_PORTC_DEN_R = 0xF0;
	GPIO_PORTD_DEN_R = 0x0F;

	//Write Commands to Initialize LCD.
	LcdCommandWrite(0x38); // Mode to work (5x7 display)
	LcdCommandWrite(0x06); // Set to Increment automatically.
	LcdCommandWrite(0x0E); // Turn on LCD and cursor.
	LcdCommandWrite(0x01); // Clear content.

	init();

}

void init(){
	int cursor = 0;
	nextString(cursor);
	int wasPressed = 0;

	//Init Infinite Loop.
	while(1){
		//Scroll Up.
		if(isPressedUp() && !wasPressed){

			//Move up in circular array.
			cursor-- ;
			cursor = (cursor < 0) ? MESSAGE_SET_SIZE-1 : cursor;

			menuUp(cursor);

			//Delay and set software debouncing variable.
			SysCtlDelay(300000);
			wasPressed = 1;

		}
		else if(isPressedDown() && !wasPressed){
			//Move Down in circular array.
			cursor = (cursor+1)%MESSAGE_SET_SIZE;

			menuDown(cursor);

			//Delay and set software debouncing variable.
			SysCtlDelay(300000);
			wasPressed = 1 ;
		}
		else if(!isPressedDown() && !isPressedUp() && wasPressed){
			//If none is pressed.
			wasPressed = 0;
		}
	}
}
void nextString(int i){
	writeString(c[(i%MESSAGE_SET_SIZE)]);
	SysCtlDelay(200000);
	LcdCommandWrite(0xA8);
	writeString(c[(i+1)%MESSAGE_SET_SIZE]);
```

```
	SysCtlDelay(200000);
	LcdCommandWrite(0x80);

}

//Stub for modularization.
void menuUp(int i){
	nextString(i);
}

//Stub for modularization.
void menuDown(int i){
	nextString(i);
}

void writeLetter(char letter){
	LcdDataWrite(letter);
}

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

}

void LcdCommandWrite(int command){

	//SET RS to low
	//SET E to low
	//PA2 = RS
	//PA3 = E
	GPIO_PORTA_DATA_R &= 0xF3;

	writeDataToPort(command);

	//set E line high to begin write cycle.
	GPIO_PORTA_DATA_R |= 0x08;

	//Pause to allow LCD to accept data.
	//
	// Delay for a bit
	//
	SysCtlDelay(20000);


	//Turn low to finish write cycle.
	GPIO_PORTA_DATA_R &= 0xF7;

}

void LcdDataWrite(char letter){

	//SET RS to High
	//SET E to low
	//PA2 = RS
	//PA3 = E
```

```
	GPIO_PORTA_DATA_R |= 0x04;
	GPIO_PORTA_DATA_R &= 0xF7;

	//Write.
	writeDataToPort(letter);

	//set E line high to begin write cycle.
	GPIO_PORTA_DATA_R |= 0x08;

	//Pause to allow LCD to accept data.
	//
	// Delay for a bit
	//
	SysCtlDelay(20000);

	//Turn low to finish write cycle.
	GPIO_PORTA_DATA_R &= 0xF7;

}

void writeDataToPort(int data){

	GPIO_PORTC_DATA_R &= 0x0F;
	GPIO_PORTD_DATA_R &= 0xF0;

	GPIO_PORTC_DATA_R |= (data & 0xF0) ; //C
	GPIO_PORTD_DATA_R |= (data & 0x0F) ; //D

}



//Helper Functions
bool isPressedUp(){
	return (GPIO_PORTE_DATA_R & 0x04) ;
}

bool isPressedDown(){
	return (GPIO_PORTE_DATA_R & 0x02);
}
// Not Yet Implemented.
void delay(uint32_t amount){
	SysCtlDelay(amount);
}
```