```
 2  * buttons.h
 7
 8 #ifndef BUTTONS_H
 9 #define BUTTONS_H
10
11 #include <stdint.h>
12 #define BUTTONS_INIT_STATUS_OK 1
13 #define BUTTONS_INIT_STATUS_FAIL 0
14 #define BUTTONS_BTN0_MASK 0x1
15 #define BUTTONS_BTN1_MASK 0x2
16 #define BUTTONS_BTN2_MASK 0x4
17 #define BUTTONS_BTN3_MASK 0x8
18
19 // Initializes the button driver software and hardware. Returns one of the defined
   status values (above).
20 int32_t buttons_init();
21
22 // Returns the current value of all 4 buttons as the lower 4 bits of the returned
   value.
23 // bit3 = BTN3, bit2 = BTN2, bit1 = BTN1, bit0 = BTN0.
24 int32_t buttons_read();
25
26 // Runs a test of the buttons. As you push the buttons, graphics and messages will be
   written to the LCD
27 // panel. The test will until all 4 pushbuttons are simultaneously pressed.
28 void buttons_runTest();
29
30 #endif
31
```

```c
 * buttons.c
#include "buttons.h"
#include "supportFiles/display.h"
#include "xil_io.h" // Helper function to read GPIO registers.

#define GPIO_DATA_OFFSET 0x00 //offset for the GPIO_DATA register
#define GPIO_TRI_OFFSET 0x04 //offset for the GPIO_TRI register
#define GPIO_TRI_SET_INPUT 1 //start input

#define BUTTONS_ALL_BUTTONS_PRESSED 0x0F // if all the buttons are pressed
#define BUTTONS_TEXT_SIZE 2 // The size of the text to display on screen

// How big the rectangle is
// They are all the same size, so use the same values
#define BUTTONS_ALL_BUTTONS_X_LENGTH DISPLAY_WIDTH/4 //how big to make the box in the x direction (aka one forth)
#define BUTTONS_ALL_BUTTONS_Y_LENGTH DISPLAY_HEIGHT/2 //how big to make the box in the y direction (aka half)
#define BUTTONS_ALL_BUTTONS_X_LENGTH_BASE 0 //where the screen starts x
#define BUTTONS_ALL_BUTTONS_Y_LENGTH_BASE 0 //where the screen starts y
#define BUTTONS_CURS_SHIFT 18 // how far to shift over


#define BUTTONS_TEXT_Y_POSITION 55 // All button's text is on the same line, so share the value

// Button 1's on screen rectangle values
#define BUTTONS_BUTTON0_X_CORD DISPLAY_WIDTH-(DISPLAY_WIDTH/4) //x cord for 1st box

#define BUTTONS_BUTTON1_X_CORD DISPLAY_WIDTH/2 //x cord for 2nd box

#define BUTTONS_BUTTON2_X_CORD DISPLAY_WIDTH/4 //x cord for 3rd box

#define BUTTONS_BUTTON3_X_CORD 0 //x cord for 3rd box


#define BUTTONS_GPIO_DEVICE_BASE_ADDRESS XPAR_PUSH_BUTTONS_BASEADDR //define base address


#define BUTTONS_TRUE 1 //when things are true return 1
#define BUTTONS_FALSE 0 //when things are false return 0
#define BUTTONS_BNT0 0 //matching number to text for box 1
#define BUTTONS_BNT1 1 //matching number to text for box 2
#define BUTTONS_BNT2 2 //matching number to text for box 3
#define BUTTONS_BNT3 3 //matching number to text for box 4

//**********Prototype_Functions**********
int32_t buttons_readGpioRegister(int32_t offset);//Reading from the register function prototype
void buttons_writeGpioRegister(int32_t offset, int32_t value);//Writing to the register function prototype
void display_buttons(int16_t x_cor, uint16_t box_color, int16_t text_color, int16_t btn_number);

//**********Main_Functions**********
// Initializes the button driver software and hardware. Returns one of the defined status values (above).
```

```
 56 int32_t buttons_init(){
 57     buttons_writeGpioRegister(GPIO_TRI_OFFSET, GPIO_TRI_SET_INPUT);
 58     if(GPIO_TRI_SET_INPUT == buttons_readGpioRegister(GPIO_DATA_OFFSET)){
 59         return BUTTONS_INIT_STATUS_OK; //register was updated
 60     }
 61     else{
 62         return BUTTONS_INIT_STATUS_FAIL;//register was not updated
 63     }
 64 }
 65
 66 // Returns the current value of all 4 buttons as the lower 4 bits of the returned
    value.
 67 // bit3 = BTN3, bit2 = BTN2, bit1 = BTN1, bit0 = BTN0.
 68 int32_t buttons_read(){
 69     return buttons_readGpioRegister(GPIO_DATA_OFFSET);
 70 }
 71
 72 // Runs a test of the buttons. As you push the buttons, graphics and messages will be
    written to the LCD
 73 // panel. The test will until all 4 pushbuttons are simultaneously pressed.
 74 void buttons_runTest(){
 75     buttons_init(); //Initializes the buttons
 76     display_init(); //Initializes the screen
 77     display_fillScreen(DISPLAY_BLACK); //fills the screen black
 78     char B0_pressed = BUTTONS_FALSE; //initializing B0
 79     char B1_pressed = BUTTONS_FALSE; //initializing B1
 80     char B2_pressed = BUTTONS_FALSE; //initializing B2
 81     char B3_pressed = BUTTONS_FALSE; //initializing B3
 82     int32_t ButtonsPressed = buttons_read();
 83     while(ButtonsPressed != BUTTONS_ALL_BUTTONS_PRESSED){
 84         ButtonsPressed = buttons_read();
 85         if((ButtonsPressed & BUTTONS_BTN0_MASK) && !B0_pressed){ //checks against the
    Mask to see if button 0 is pressed
 86             display_buttons(BUTTONS_BUTTON0_X_CORD, DISPLAY_YELLOW, DISPLAY_BLACK,
    BUTTONS_BNT0);
 87             B0_pressed = BUTTONS_TRUE;
 88         }
 89         else if(!(ButtonsPressed & BUTTONS_BTN0_MASK) && B0_pressed){ //checks against
    the Mask to see if button 0 is not pressed
 90             display_buttons(BUTTONS_BUTTON0_X_CORD, DISPLAY_BLACK, DISPLAY_BLACK,
    BUTTONS_BNT0);
 91             B0_pressed = BUTTONS_FALSE;
 92         }
 93         if((ButtonsPressed & BUTTONS_BTN1_MASK) && !B1_pressed){ //checks against the
    Mask to see if button 1 is pressed
 94             display_buttons(BUTTONS_BUTTON1_X_CORD, DISPLAY_CYAN, DISPLAY_BLACK,
    BUTTONS_BNT1);
 95             B1_pressed = BUTTONS_TRUE;
 96         }
 97         else if(!(ButtonsPressed & BUTTONS_BTN1_MASK) && B1_pressed){ //checks against
    the Mask to see if button 1 is not pressed
 98             display_buttons(BUTTONS_BUTTON1_X_CORD, DISPLAY_BLACK, DISPLAY_BLACK,
    BUTTONS_BNT1);
 99             B1_pressed = BUTTONS_FALSE;
100         }
101         if((ButtonsPressed & BUTTONS_BTN2_MASK) && !B2_pressed){ //checks against the
    Mask to see if button 2 is pressed
102             display_buttons(BUTTONS_BUTTON2_X_CORD, DISPLAY_RED, DISPLAY_WHITE,
```

```
BUTTONS_BNT2);
103             B2_pressed = BUTTONS_TRUE;
104         }
105         else if(!(ButtonsPressed & BUTTONS_BTN2_MASK) && B2_pressed){ //checks against
the Mask to see if button 2 is not pressed
106             display_buttons(BUTTONS_BUTTON2_X_CORD, DISPLAY_BLACK, DISPLAY_BLACK,
BUTTONS_BNT2);
107             B2_pressed = BUTTONS_FALSE;
108         }
109         if((ButtonsPressed & BUTTONS_BTN3_MASK) && !B3_pressed){ //checks against the
Mask to see if button 3 is pressed
110             display_buttons(BUTTONS_BUTTON3_X_CORD, DISPLAY_BLUE, DISPLAY_WHITE,
BUTTONS_BNT3);
111             B3_pressed = BUTTONS_TRUE;
112         }
113         else if(!(ButtonsPressed & BUTTONS_BTN3_MASK) && B3_pressed){ //checks against
the Mask to see if button 3 is not pressed
114             display_buttons(BUTTONS_BUTTON3_X_CORD, DISPLAY_BLACK, DISPLAY_BLACK,
BUTTONS_BNT3);
115             B3_pressed = BUTTONS_FALSE;
116         }
117     }
118     display_fillScreen(DISPLAY_BLACK); //fills the screen black
119 }
120 //**********Helper_Functions**********
121 int32_t buttons_readGpioRegister(int32_t offset){// Reads from the register
122     //use the low-level Xilinx call.
123     return Xil_In32(BUTTONS_GPIO_DEVICE_BASE_ADDRESS + offset);
124 }
125 void buttons_writeGpioRegister(int32_t offset, int32_t value){// Writes to the
register
126     Xil_Out32(BUTTONS_GPIO_DEVICE_BASE_ADDRESS + offset, value);//use the low-level
Xilinx call.
127 }
128 void display_buttons(int16_t x_cor, uint16_t box_color, int16_t text_color, int16_t
btn_number){ //function to display boxes and text on screen
129     display_fillRect(x_cor, BUTTONS_ALL_BUTTONS_Y_LENGTH_BASE,
BUTTONS_ALL_BUTTONS_X_LENGTH, BUTTONS_ALL_BUTTONS_Y_LENGTH, box_color); //fills the
box
130     display_setCursor(x_cor + BUTTONS_CURS_SHIFT, BUTTONS_TEXT_Y_POSITION); //sets the
cursor
131     display_setTextColor(text_color); //sets the color of the text
132     display_setTextSize(BUTTONS_TEXT_SIZE); // sets the size of the text
133     display_print("BTN"); // writes out the name of the box
134     display_println(btn_number); //writes out the number of the box
135 }
136
137
```

```
 2 * switches.h
 7
 8#ifndef SWITCHES_H
 9#define SWITCHES_H
10
11#include <stdint.h>
12#define SWITCHES_INIT_STATUS_OK 1
13#define SWITCHES_INIT_STATUS_FAIL 0
14#define SWITCHES_SW0_MASK 0x1
15#define SWITCHES_SW1_MASK 0x2
16#define SWITCHES_SW2_MASK 0x4
17#define SWITCHES_SW3_MASK 0x8
18
19// Initializes the SWITCHES driver software and hardware. Returns one of the STATUS
   values defined above.
20int32_t switches_init();
21
22// Returns the current value of all 4 switches as the lower 4 bits of the returned
   value.
23// bit3 = SW3, bit2 = SW2, bit1 = SW1, bit0 = SW0.
24int32_t switches_read();
25
26// Runs a test of the switches. As you slide the switches, LEDs directly above the
   switches will illuminate.
27// The test will run until all switches are slid upwards. When all 4 slide switches are
   slid upward,
28// this function will return.
29void switches_runTest();
30
31#endif
32
```

```c
 * switches.c
#include <stdio.h>
#include "switches.h"
#include "supportFiles/leds.h"
#include "xil_io.h"
#include "xparameters.h"

#define GPIO_DATA_OFFSET 0x00 //offset for the GPIO_DATA register
#define GPIO_TRI_OFFSET 0x04 //offset for the GPIO_TRI register
#define GPIO_TRI_SET_INPUT 1 //start input
#define SWITCHES_ALL_SWITCHES_ON 0x0F
#define SWITCHES_ALL_SWITCHES_OFF 0x00
#define TRUE 1
#define FALSE 0

//**********Prototype_Functions**********
int32_t switches_readGpioRegister(int32_t offset);
void switches_writeGpioRegister(int32_t offset, int32_t value);

//**********Switches.h_Functions**********
int32_t switches_init(){// Initializes the SWITCHES driver software and hardware. Returns one of the STATUS values defined above.
    switches_writeGpioRegister(GPIO_TRI_OFFSET, GPIO_TRI_SET_INPUT);
    if(GPIO_TRI_SET_INPUT == switches_readGpioRegister(GPIO_DATA_OFFSET)){
        return SWITCHES_INIT_STATUS_OK; //register was updated
    }
    else{
        return SWITCHES_INIT_STATUS_FAIL;//register was not updated
    }
}

// Returns the current value of all 4 switches as the lower 4 bits of the returned value.
// bit3 = SW3, bit2 = SW2, bit1 = SW1, bit0 = SW0.
int32_t switches_read(){
    return switches_readGpioRegister(GPIO_DATA_OFFSET);
}

// Runs a test of the switches. As you slide the switches, LEDs directly above the switches will illuminate.
// The test will run until all switches are slid upwards. When all 4 slide switches are slid upward,
// this function will return.
void switches_runTest(){
    switches_init(); //initialize the switches
    int32_t SwitchesOn = switches_read(); //read in what switches are being slide
    while (TRUE){
        SwitchesOn = switches_read();//read in what switches are being slide
        leds_init(FALSE); //turn all switches off to start
        switch(SwitchesOn){ //checking for all switches to be on
        case (SWITCHES_ALL_SWITCHES_ON): //if all the switches are on enter case
                leds_write(SWITCHES_ALL_SWITCHES_OFF); // turn of all the leds and exit the program
        return;
        break;
        default: leds_write(SwitchesOn); //as long as not all the switches are on then continue writing to the board
        }
```

```
58      }
59  }
60
61  //************************Helper_Functions*******************************
62
63  int32_t switches_readGpioRegister(int32_t offset){// Reads from the register
64      //use the low-level Xilinx call.
65      return Xil_In32(XPAR_SLIDE_SWITCHES_BASEADDR + offset);
66  }
67  void switches_writeGpioRegister(int32_t offset, int32_t value){// Writes to the
    register
68      Xil_Out32(XPAR_SLIDE_SWITCHES_BASEADDR + offset, value);//use the low-level Xilinx
    call.
69  }
70
71
```