```c
 2  * buttons.c
 7
 8 #include "buttons.h"
 9 #include "xparameters.h"          //needed to access base address of GPIOs
10 #include "supportFiles/display.h" //needed to access the LCD screen
11 #include "xil_io.h"               //includes the low-level Xilinx functions needed for
   reading and writing to GPIOs
12 #include "stdio.h"                //needed to make printf work
13
14 #define BUTTONS_GPIO_BASE_ADDRESS XPAR_PUSH_BUTTONS_BASEADDR //base address from
   xparameters.h
15 #define BUTTONS_DATA_OFFSET 0                                    //value based on register
   documentation provided by Xilinx for GPIO_DATA
16 #define BUTTONS_TRI_OFFSET 4                                     //value based on register
   documentation provided by Xilinx for GPIO_TRI
17 #define BUTTONS_VALUE 0xF                                        //value to be written to
   GPIO_TRI to make sure it behaves correctly
18 #define BUTTONS_ALL_ON 0xF
19
20 #define TEXT_SIZE 2
21
22 //to simplify rectangle display functions
23 #define RECTANGLE_WIDTH DISPLAY_WIDTH/4
24 #define RECTANGLE_HEIGHT DISPLAY_HEIGHT/2
25
26 //to avoid magic numbers
27 #define RECTANGLE_WIDTH_TIMES_2 RECTANGLE_WIDTH*2
28 #define RECTANGLE_WIDTH_TIMES_3 RECTANGLE_WIDTH*3
29 #define HALF_RECTANGLE_HEIGHT RECTANGLE_HEIGHT/2
30 #define QUARTER_RECTANGLE_WIDTH RECTANGLE_WIDTH/4
31
32 //BTN strings
33 #define BTN_0 " BTN0"
34 #define BTN_1 " BTN1"
35 #define BTN_2 " BTN2"
36 #define BTN_3 "BTN3"
37
38
39 //helper function to read GPIO registers
40 int32_t buttons_readGpioRegister(int32_t offset){
41     return Xil_In32(BUTTONS_GPIO_BASE_ADDRESS + offset); //using low-level Xilinx call
42 }
43
44 //helper function to write to GPIO registers
45 void buttons_writeGpioRegister(int32_t offset, int32_t value){
46     Xil_Out32(BUTTONS_GPIO_BASE_ADDRESS + offset, value); //low-level Xilinx call
47 }
48
49 //initializing software and hardware for buttons
50 int32_t buttons_init(){
51     buttons_writeGpioRegister(BUTTONS_TRI_OFFSET, BUTTONS_VALUE);      //writing only
   to GPIO_TRI
52     if(buttons_readGpioRegister(BUTTONS_TRI_OFFSET) == BUTTONS_VALUE){ //reads from
   GPIO_TRI to make sure the data was correctly written to it
53         return BUTTONS_INIT_STATUS_OK;
54     }                                                                 //GPIO_DATA
   doesn't need to be written to in order to behave correctly
55
```

```
56      return BUTTONS_INIT_STATUS_FAIL;                           //if the
    GPIO_TRI was written to incorrectly, we return this to indicate an error
57 }
58
59 int32_t buttons_read(){
60      return buttons_readGpioRegister(BUTTONS_DATA_OFFSET) & BUTTONS_VALUE; //need to
    bit-mask in order to get the last 4 bits to work with
61 }
62
63 void buttons_runTest(){
64      display_init();                            //Must init all of the software and underlying
    hardware for LCD.
65      display_fillScreen(DISPLAY_BLACK);  //Blank the screen.
66      display_setTextSize(TEXT_SIZE);     //set text size
67
68      int32_t readInVal = 0;                     //variable that will contain the values read
    from GPIO_DATA
69      int32_t oldVal = 0;                        //variable to store the former value read from
    GPIO_DATA
70
71      if(buttons_init()){                        //check to make sure the buttons were
    initialized without issue
72
73          while(readInVal != BUTTONS_ALL_ON){ //will run until all 4 buttons are pressed
    simultaneously
74
75              readInVal = buttons_read();        //getting value from GPIO_DATA, which
    buttons are being pressed
76
77              if(oldVal != readInVal){
78                  if((readInVal & BUTTONS_BTN3_MASK) == BUTTONS_BTN3_MASK){
    //bit mask to only get btn3's value
79                      //blue rectangle shape, color, and text
80                      display_fillRect(0, 0, RECTANGLE_WIDTH, RECTANGLE_HEIGHT,
    DISPLAY_BLUE); //draws the solid blue rect
81                      display_setCursor(RECTANGLE_WIDTH/4, HALF_RECTANGLE_HEIGHT);
    //starts text in correct place
82                      display_setTextColor(DISPLAY_WHITE);
    // Make the text white.
83                      display_println(BTN_3);
84                  }
85                  else if((readInVal & BUTTONS_BTN3_MASK) != BUTTONS_BTN3_MASK){
    //when button is no longer pressed
86                      display_fillRect(0, 0, RECTANGLE_WIDTH, RECTANGLE_HEIGHT,
    DISPLAY_BLACK); //black out that section of the LCD
87                  }
88
89                  if((readInVal & BUTTONS_BTN2_MASK) == BUTTONS_BTN2_MASK){
    //bit mask to get only btn2's value
90                      //red rectangle shape, color, and text
91                      display_fillRect(RECTANGLE_WIDTH, 0, RECTANGLE_WIDTH,
    RECTANGLE_HEIGHT, DISPLAY_RED); //draws solid red rect
92                      display_setCursor(RECTANGLE_WIDTH, HALF_RECTANGLE_HEIGHT);
    //starts cursor in correct place
93                      display_setTextColor(DISPLAY_WHITE);
    //Make the text white.
94                      display_println(BTN_2);
95                  }
```

```
 96                 else if((readInVal & BUTTONS_BTN2_MASK) != BUTTONS_BTN2_MASK){
    //when button is no longer pressed
 97                     display_fillRect(RECTANGLE_WIDTH, 0, RECTANGLE_WIDTH,
    RECTANGLE_HEIGHT, DISPLAY_BLACK); //black out that section of the LCD
 98                 }
 99
100                 if((readInVal & BUTTONS_BTN1_MASK) == BUTTONS_BTN1_MASK){ //bit mask
    to get only btn1's value
101                     //green rectangle shape, color, and text
102                     display_fillRect(RECTANGLE_WIDTH_TIMES_2, 0, RECTANGLE_WIDTH,
    RECTANGLE_HEIGHT, DISPLAY_GREEN); //draws solid green rect
103                     display_setCursor(RECTANGLE_WIDTH_TIMES_2, HALF_RECTANGLE_HEIGHT);
    //starts cursor in correct place
104                     display_setTextColor(DISPLAY_BLACK);
    //Make the text black.
105                     display_println(BTN_1);
106                 }
107                 else if((readInVal & BUTTONS_BTN1_MASK) != BUTTONS_BTN1_MASK){
    //when button is no longer pressed
108                     display_fillRect(RECTANGLE_WIDTH_TIMES_2, 0, RECTANGLE_WIDTH,
    RECTANGLE_HEIGHT, DISPLAY_BLACK); //black out that section of the LCD
109                 }
110
111                 if((readInVal & BUTTONS_BTN0_MASK) == BUTTONS_BTN0_MASK){ //bit mask
    to get only btn0's value
112                     //yellow rectangle shape, color, and text
113                     display_fillRect(RECTANGLE_WIDTH_TIMES_3, 0, RECTANGLE_WIDTH,
    RECTANGLE_HEIGHT, DISPLAY_YELLOW); //draws solid yellow rect
114                     display_setCursor(RECTANGLE_WIDTH_TIMES_3, HALF_RECTANGLE_HEIGHT);
    //starts cursor in correct place
115                     display_setTextColor(DISPLAY_BLACK);
    //Make the text black.
116                     display_println(BTN_0);
117                 }
118                 else if((readInVal & BUTTONS_BTN0_MASK) != BUTTONS_BTN0_MASK){
    //when button is no longer pressed
119                     display_fillRect(RECTANGLE_WIDTH_TIMES_3, 0, RECTANGLE_WIDTH,
    RECTANGLE_HEIGHT, DISPLAY_BLACK); //black out that section of the LCD
120                 }
121             }
122             oldVal = readInVal; //store the value just used to check it against the
    one about to be read in
123         }
124         display_fillScreen(DISPLAY_BLACK);  // Blank the screen when all 4 buttons are
    pressed simultaneously
125     }
126     else{
127         printf("%s\n\r", "Button initialization FAILED");
128     }
129 }
130
131
```