```c
 *  buttons.c


#include "stdio.h"
#include "buttons.h"
#include "supportFiles/leds.h"
#include "supportFiles/display.h"


// Initializes the button driver software and hardware. Returns one of the defined status values
(above).
int buttons_init() {
    uint32_t *ptr = (uint32_t *)  XPAR_GPIO_PUSH_BUTTONS_BASEADDR + BUTTON_OFFSET;
    *ptr = BUTTON_TRISTATE_SET;

    // This statement checks to make sure that the pointer was changed to indicated value
    if(*ptr != BUTTON_TRISTATE_SET) {

        // If the pointer is not the same value as the set value it will return false
        return BUTTONS_INIT_STATUS_FAIL;
    }

    return BUTTONS_INIT_STATUS_OK;
}

// Returns the current value of all 4 buttons as the lower 4 bits of the returned value.
// bit3 = BTN3, bit2 = BTN2, bit1 = BTN1, bit0 = BTN0.
int32_t buttons_read() {

    // Pointer declaration that is assigned the gpio push button address
    uint32_t *ptr = (uint32_t *) XPAR_GPIO_PUSH_BUTTONS_BASEADDR;
    return *ptr;
}


/*
*  Runs a test of the buttons. As you push the buttons, graphics and messages will be written to
the LCD
*  panel. The test will until all 4 pushbuttons are simultaneously pressed.
*/


void buttons_runTest() {

    buttons_init(); // Button driver activates

    display_init();     // Display init allows the display to function
    display_fillScreen(DISPLAY_WHITE);      // Screen blanked


    /*
    *  Test function will run until all buttons are pushed, or when the last 4 bits are 1111.
    *  Also note that the conditions call buttons_read() and perform bitwise operation & 0xF.
    *  This isolates the last four bits, and at the same time fetches the current push button
    *  values.
    */
```

```
    while((buttons_read() & BUTTON_ISOLATE_BITS)  < BUTTON_ISOLATE_BITS){

        // Case for button 0; circle turns red and displays button number when pressed.
        if ((buttons_read() & BUTTON_ISOLATE_BITS) & BUTTONS_ISOLATE_BIT_B0) {

            // Red circle with coordinates
            display_fillCircle( (BUTTONS_MAX_X /4) , (BUTTONS_MAX_Y / 4)
,BUTTONS_CIRCLE_RADIUS,BUTTONS_RED);

            // Coordinate of number and characteristics
            display_setCursor( (BUTTONS_MAX_X /4) ,  (BUTTONS_MAX_Y / 4) );
            display_setTextColor(DISPLAY_BLACK);
            display_setTextSize(BUTTONS_TEXT_SIZE );

            // Prints the associated number on LED screen
            display_println("0");

        }
        else {
            display_fillCircle( (BUTTONS_MAX_X /4) , (BUTTONS_MAX_Y / 4)
,BUTTONS_CIRCLE_RADIUS,BUTTONS_YELLOW);     // Circle remains yellow when button not pressed
        }


        /*
         *  The next three if statements correlate to the mentioned button in
         *  following comments.  Similar to button 0.
         */


        // Case for button 1; circle turns red and displays button number when pressed.
        if ((buttons_read() & BUTTON_ISOLATE_BITS) & BUTTONS_ISOLATE_BIT_B1) {

            display_fillCircle((BUTTONS_MAX_X /4), ((BUTTONS_MAX_Y * 3) /
4),BUTTONS_CIRCLE_RADIUS,BUTTONS_RED);

            display_setCursor((BUTTONS_MAX_X /4), ((BUTTONS_MAX_Y * 3) / 4));
            display_setTextColor(DISPLAY_BLACK);
            display_setTextSize(BUTTONS_TEXT_SIZE );
            display_println("1");

        }
        else {
            display_fillCircle((BUTTONS_MAX_X /4), ((BUTTONS_MAX_Y * 3) / 4)
,BUTTONS_CIRCLE_RADIUS,BUTTONS_YELLOW);
        }


        // Case for button 2; circle turns red and displays button number when pressed.
        if ((buttons_read() & BUTTON_ISOLATE_BITS) & BUTTONS_ISOLATE_BIT_B2) {

            display_fillCircle( ((BUTTONS_MAX_X * 3) / 4) , (BUTTONS_MAX_Y / 4) ,
BUTTONS_CIRCLE_RADIUS ,BUTTONS_RED);

            display_setCursor( ((BUTTONS_MAX_X * 3) / 4) , (BUTTONS_MAX_Y / 4) );
            display_setTextColor(DISPLAY_BLACK);
```

```
            display_setTextSize(BUTTONS_TEXT_SIZE );
            display_println("2");
        }
        else {
            display_fillCircle( ((BUTTONS_MAX_X * 3) / 4) , (BUTTONS_MAX_Y / 4) ,
BUTTONS_CIRCLE_RADIUS ,BUTTONS_YELLOW);
        }

        // Case for button 3; circle turns red and displays button number when pressed.
        if ((buttons_read() & BUTTON_ISOLATE_BITS) & BUTTONS_ISOLATE_BIT_B3) {

            display_fillCircle( ((BUTTONS_MAX_X * 3) / 4),((BUTTONS_MAX_Y * 3) / 4) ,
BUTTONS_CIRCLE_RADIUS ,BUTTONS_RED);

            display_setCursor( ((BUTTONS_MAX_X * 3) / 4) , ((BUTTONS_MAX_Y * 3) / 4) );
            display_setTextColor(DISPLAY_BLACK);
            display_setTextSize(BUTTONS_TEXT_SIZE );
            display_println("3");
        }
        else {
            display_fillCircle( ((BUTTONS_MAX_X * 3) / 4),((BUTTONS_MAX_Y *
3)/4),BUTTONS_CIRCLE_RADIUS ,BUTTONS_YELLOW);
        }

    }

    display_fillScreen(DISPLAY_WHITE);  // Screen cleared at the end of test.

}
```