```c
/*
 * buttonHandler.c
 *
 *  Created on: Jun 4, 2015
 *      Author: Taylor Cowley
 */

#include "buttonHandler.h"
#include "simonDisplay.h"

#define RUN_TEST_TERMINATION_MESSAGE1 "buttonHandler_runTest()"
#define RUN_TEST_TERMINATION_MESSAGE2 "terminated."
#define RUN_TEST_TEXT_SIZE 2

//this is to tell ourselves and everyone else whether we are active
bool buttonHandler_enable_flag = false;

//We have detected a button release!
bool buttonHandler_release_detected = false;

//this is the variable that stores the button the user pushed last
uint8_t buttonHandler_last_pushed_button = 0;

//Stores the current state of the state machine
buttonHandler_st_t currentState = init_st;


// Get the simon region numbers. See the source code for the region numbering scheme.
uint8_t buttonHandler_getRegionNumber(){
    return buttonHandler_last_pushed_button;
}

// Turn on the state machine. Part of the interlock.
void buttonHandler_enable(){
    buttonHandler_enable_flag = true;   //so we true the enable flag
}

// Turn off the state machine. Part of the interlock.
void buttonHandler_disable(){
    buttonHandler_enable_flag = false;  //so we false the enable flag
    currentState = init_st;
}

// Other state machines can call this function to see if the user has stopped touching the pad.
bool buttonHandler_releaseDetected(){
    return buttonHandler_release_detected;
}

// Standard tick function.
void buttonHandler_tick(){


    //This is the timer to let the touch sensor cool
    static int16_t touch_ad_timer = 0;

    //Execute the state function
    switch(currentState){
    case init_st:               //init what we need
        buttonHandler_release_detected = false;     //We haven't detected a button release yet
        break;

    case wait_for_enable_st:    //wait to be enabled
        //we can't do anything
        break;

    case wait_for_touch_st:     //waiting to be touched
        //still can't do anything
        break;

    case touch_ad_timer_st:     //Horray! waiting for the touch sensor to cool
        touch_ad_timer = touch_ad_timer - 1;    //count down our beautiful timer
        break;

    case record_touch_st:
        int16_t x, y;   //for recording where the touch was
        uint8_t z;      //necessary for getTouchedPoint, but not used
        display_getTouchedPoint(&x,&y,&z);      //get where our touch was!
        //Calculate which region that touch was and record it.
        buttonHandler_last_pushed_button = simonDisplay_computeRegionNumber(x, y);
        simonDisplay_drawSquare(buttonHandler_last_pushed_button, false); //Draw the square

    case wait_for_release_st:   //Just waiting for the user to let go
        //can't do anything
        break;

    case end_st:                //We have finished! Time to finalize things
        buttonHandler_release_detected = true;                          //We've detected a 
button release!
        simonDisplay_drawSquare(buttonHandler_last_pushed_button, true);//Draw the square
        simonDisplay_drawButton(buttonHandler_last_pushed_button);      // reraw the button.

        break;

    case wait_for_disable_st:   //Now we do nothing until we are disabled
        //doing nothing
        break;

    default:    //This is an error
        printf("Invalid state");    //print the error
        break;

    }

    //Perform state update now
    switch(currentState){
        case init_st:               //here we init things
            currentState = wait_for_enable_st;      //init only lasts 1 tick
            break;

        case wait_for_enable_st:    //we chill until we are enabled
            if(buttonHandler_enable_flag){
                currentState = wait_for_touch_st;   //and so we move on
                simonDisplay_drawAllButtons();      // Draw all of the buttons.
            }
            break;

        case wait_for_touch_st:     //now we are active, but waiting for a touch
            if(display_isTouched()){                                //we are touched!
                display_clearOldTouchData();                        //Clear data
                touch_ad_timer = BUTTON_HANDLER_TOUCH_COOLDOWN;     //and start the timer
                currentState = touch_ad_timer_st;                   //move to next state
            }
            break;

        case touch_ad_timer_st:     //waiting for touch sensor to cool
            if(touch_ad_timer <= 0){                //timer is yet done
                currentState = record_touch_st;     //Next state= record the touch
            }
            break;

        case record_touch_st:       //We've recorded the touch, now to wait for release
            currentState = wait_for_release_st;     //next state- wait for them to let go
            break;

        case wait_for_release_st:   //waiting for user to stop touching :/
            if(!display_isTouched()){   //They stopped touching us!
                currentState = end_st;  //We are done now
            }
            break;

        case end_st:                //We are done!
            currentState = wait_for_disable_st;     //time to move on
            break;

        case wait_for_disable_st:   //now we wait to be disabled
            if(!buttonHandler_enable_flag){         //they disabled us
                currentState = init_st;             //now we wait to be enabled
            }
            break;

        default:                    //This is an error
            printf("Invalid state");    //print the error
            break;

        }

}







// buttonHandler_runTest(int16_t touchCount) runs the test until
// the user has touched the screen touchCount times. It indicates
// that a button was pushed by drawing a large square while
// the button is pressed and then erasing the large square and
// redrawing the button when the user releases their touch.

```

```c
void buttonHandler_runTest(int16_t touchCountArg) {
  int16_t touchCount = 0;              // Keep track of the number of touches.
  display_init();                      // Always have to init the display.
  display_fillScreen(DISPLAY_BLACK);  // Clear the display.
  simonDisplay_drawAllButtons();      // Draw the four buttons.
  buttonHandler_enable();
  while (touchCount < touchCountArg) {  // Loop here while touchCount is less than the touchCountArg
    buttonHandler_tick();                // Advance the state machine.
    utils_msDelay(1);           // Wait here for 1 ms.
    if (buttonHandler_releaseDetected()) {  // If a release is detected, then the screen was touched.
      touchCount++;                         // Keep track of the number of touches.
      printf("button released: %d\n\r", buttonHandler_getRegionNumber());  // Get the region number that was touched.
      buttonHandler_disable();             // Interlocked behavior: handshake with the button handler (now disabled).
      utils_msDelay(1);                     // wait 1 ms.
      buttonHandler_tick();                 // Advance the state machine.
      buttonHandler_enable();               // Interlocked behavior: enable the buttonHandler.
      utils_msDelay(1);                     // wait 1 ms.
      buttonHandler_tick();                 // Advance the state machine.
    }
  }
  display_fillScreen(DISPLAY_BLACK);            // clear the screen.
  display_setTextSize(RUN_TEST_TEXT_SIZE);      // Set the text size.
  display_setCursor(0, display_height()/2);     // Move the cursor to a rough center point.
  display_println(RUN_TEST_TERMINATION_MESSAGE1);   // Print the termination message on two lines.
  display_println(RUN_TEST_TERMINATION_MESSAGE2);
}

```