```c
 * buttonHandler.c
#include "buttonHandler.h"
#include "simonDisplay.h"
#include "supportFiles/display.h"
#include "supportFiles/utils.h"
#include <stdio.h>

//**********CODE_PROVIDED_BY_PROFESSOR**********//
#define RUN_TEST_TERMINATION_MESSAGE1 "buttonHandler_runTest()"  // Info message.
#define RUN_TEST_TERMINATION_MESSAGE2 "terminated."              // Info message.
#define RUN_TEST_TEXT_SIZE 2                                     // Make text easy to see.
#define RUN_TEST_TICK_PERIOD_IN_MS 100                           // Assume a 100 ms tick period.
#define TEXT_MESSAGE_ORIGIN_X 0                                  // Text is written starting at the right, and
#define TEXT_MESSAGE_ORIGIN_Y (DISPLAY_HEIGHT/2)                 // middle.

//**********MY_VAR**********//
#define ENABLE_FLAG_ON 1                                            // VAR to set the ENABLE FLAG on
#define ENABLE_FLAG_OFF 0                                           // VAR to set the ENABLE FLAG off
#define TRUE 1                                                      // sets true to 1
#define FALSE 0                                                     // sets false to 0
uint8_t buttonEnableFlag = 0;                                       // Initializes the enable flag
uint8_t touchRelease = 0;                                           // initializes the touch_release VAR
uint8_t delayCounter = 0;                                           // counter to make the button_delay_st completely run
uint8_t region;                                                     // VAR to save the region where the screen is being pressed
uint8_t buttonInitFlag = 0;                                         // Flag to signal if the buttons have been printed

enum buttonHandler_st_m {                                           // sets the states of the state machine
    button_int_st,                                                  // state to init
    buttons_print_st,                                               // state to print the buttons to the screen
    button_wait_button_touch_st,                                    // state to wait until the screen is pressed
    button_delay_st,                                                // state to pause for a split second so the state can find region touched and then print the buttons
    button_touch_st,                                                // state for when the screen is touched
    button_touch_release_st,                                        // state for when the screen is release from the touch
    button_end_st                                                   // state to stop the SM until enable flag has been turned off
} buttonHandlerCurrentState = button_int_st;                        // sets the first state to buttons_init_st

uint8_t buttonHandler_getRegionNumber(){                            // function to find the position of the touch on the screen
    int16_t x = 0;                                                  // sets the x coordinate back to 0
    int16_t y = 0;                                                  // sets the x coordinate back to 0
    uint8_t z;                                                      // initializes the z coordinate
    display_getTouchedPoint(&x, &y, &z);                            // finds the new coordinates
    return simonDisplay_computeRegionNumber(x, y);                  // then returns them
}

void buttonHandler_enable(){                                        // Turn on the state machine. Part of the interlock.
    buttonEnableFlag = ENABLE_FLAG_ON;                              // sets the buttonsEnableFlag to on
}


void buttonHandler_disable(){                                       // Turn off the state machine. Part of the interlock.
    buttonEnableFlag = ENABLE_FLAG_OFF;                             // sets the buttonsEnableFlag to off
}

// The only thing this function does is return a boolean flag set by the buttonHandler state machine. To wit:
// Once enabled, the buttonHandler state-machine first waits for a touch. Once a touch is detected, the
// buttonHandler state-machine computes the region-number for the touched area. Next, the buttonHandler
// state-machine waits until the player removes their finger. At this point, the state-machine should
// set a bool flag that indicates the the player has removed their finger. Once the buttonHandler()
// state-machine is disabled, it should clear this flag.
// All buttonHandler_releasedDetected() does is return the value of this flag.
// As such, the body of this function should only contain a single line of code.
bool buttonHandler_releaseDetected(){                               // function to see if the screen has been touch
    if(touchRelease){                                               // checks if the screen has been touched
        touchRelease = FALSE;                                       // initializes the touchRelease VAR back to 0
        return TRUE;                                                // then return true
    }
    else{                                                           // if the screen was not touched
        return FALSE;                                               // then return false
    }
}

void buttonHandler_tick(){                                          // Standard tick function.
    switch(buttonHandlerCurrentState){                              // set your state that your on to buttonHandlerCurrentState
    case button_int_st:                                             // Moore state action for state #1
```

```c
        break;                                                  // ends case
    case buttons_print_st:                                      // Moore state action for state #2
        simonDisplay_drawAllButtons();                          // prints all the buttons to the screen
        break;                                                  // ends case
    case button_wait_button_touch_st:                           // Moore state action for state #3
        break;                                                  // ends case
    case button_delay_st:                                       // Moore state action for state #4
        display_clearOldTouchData();                            // clear the old data so the SM can read in the new touch data
        delayCounter++;                                         // increase the delay counter
        break;                                                  // ends case
    case button_touch_st:                                       // Moore state action for state #5
        break;                                                  // ends case
    case button_touch_release_st:                               // Moore state action for state #6
        break;                                                  // ends case
    case button_end_st:                                         // Moore state action for state #7
        break;                                                  // ends case
    };
    switch(buttonHandlerCurrentState){
    case button_int_st:                                         // Mealy transition state action for state #1
        if(buttonEnableFlag && buttonInitFlag){                 // if the flag has been raised then enter the state and the buttons have been printed
            display_clearOldTouchData();                        // clears the old data from the screen
            buttonHandlerCurrentState = button_wait_button_touch_st;// move to next state
        }
        else if (buttonEnableFlag){                             //if the only the enableFlag is raised and the buttons have not been printed then go to the print buttons state
            buttonHandlerCurrentState = buttons_print_st;       // move to next state
        }
        break;                                                  // ends case
    case buttons_print_st:                                      // Mealy transition state action for state #2
        buttonInitFlag = TRUE;                                  // set the buttonInitFlag to true because you entered the print state
        buttonHandlerCurrentState = button_wait_button_touch_st;    // move to next state
        break;                                                  // ends case
    case button_wait_button_touch_st:                           // Mealy transition state action for state #3
        if(display_isTouched()){                                // if the srceen is touched then move to the delay state to the touched region can be read
            buttonHandlerCurrentState = button_delay_st;        // move to next state
        }
        break;                                                  // ends case
    case button_delay_st:                                         // Mealy transition state action for state #4
        if (delayCounter == TRUE){                                // after the little wait to
            display_clearOldTouchData();                          // clears the old data from the screen
            delayCounter = FALSE;                                   // reset the delayCounter for the next time through the SM
            simonDisplay_drawSquare(buttonHandler_getRegionNumber(), FALSE);// draw the square associated with the region that was touched
            region = buttonHandler_getRegionNumber();             // read the new touch data in from the screen to a VAR for later
            buttonHandlerCurrentState = button_touch_st;          // move to next state
        }
        break;                                                    // ends case
    case button_touch_st:                                         // Mealy transition state action for state #5
        if(!display_isTouched()){                                 //when the display is released enter the state
            buttonHandlerCurrentState = button_touch_release_st;  // move to next state
        }
        break;                                                    // ends case
    case button_touch_release_st:                                 // Mealy transition state action for state #6
        simonDisplay_drawSquare(region, TRUE);                    //clear the boxes
        simonDisplay_drawButton(region);                          //then draw the buttons
        touchRelease = TRUE;                                      //set the release VAR to one
        buttonHandlerCurrentState = button_end_st;                // move to next state
        break;                                                    // ends case
    case button_end_st:                                           // Mealy transition state action for state #7 //set the fifth state for moore
        if(!buttonEnableFlag){                                    // wait until the the flag is lowered
            buttonInitFlag = FALSE;                               // set the initFlag off
            buttonHandlerCurrentState = button_int_st;            // move to next state
        }
        break;                                                    // ends case
    }
}

// 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.

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.
    // Draw all the buttons for the first time so the buttonHandler doesn't need to do this in an init state.
    // Ultimately, simonControl will do this when the game first starts up.
    simonDisplay_drawAllButtons();
    buttonHandler_enable();
    while (touchCount < touchCountArg) {    // Loop here while touchCount is less than the touchCountArg
        buttonHandler_tick();               // Advance the state machine.
        utils_msDelay(RUN_TEST_TICK_PERIOD_IN_MS);
        if (buttonHandler_releaseDetected()) {  // If a release is detected, then the screen was touched.
            touchCount++;                       // Keep track of the number of touches.
            // Get the region number that was touched.
            printf("button released: %d\n\r", buttonHandler_getRegionNumber());
            // Interlocked behavior: handshake with the button handler (now disabled).
            buttonHandler_disable();
            utils_msDelay(RUN_TEST_TICK_PERIOD_IN_MS);
            buttonHandler_tick();               // Advance the state machine.
            buttonHandler_enable();             // Interlocked behavior: enable the buttonHandler.
            utils_msDelay(RUN_TEST_TICK_PERIOD_IN_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(TEXT_MESSAGE_ORIGIN_X, TEXT_MESSAGE_ORIGIN_Y); // 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);
}


```