```c
 * flashSequence.c

#include "flashSequence.h"
#include "simonDisplay.h"
#include "supportFiles/display.h"
#include "supportFiles/utils.h"
#include "globals.h"
#include <stdio.h>


//**********CODE_FROM_THE_PROFFESOR**********//
// This will set the sequence to a simple sequential pattern.
// It starts by flashing the first color, and then increments the index and flashes the first
// two colors and so forth. Along the way it prints info messages to the LCD screen.
#define TEST_SEQUENCE_LENGTH 8  // Just use a short test sequence.
uint8_t flashSequence_testSequence[TEST_SEQUENCE_LENGTH] = {
    SIMON_DISPLAY_REGION_0,                                     // sets region 0 to 0
    SIMON_DISPLAY_REGION_1,                                     // sets region 1 to 1
    SIMON_DISPLAY_REGION_2,                                     // sets region 2 to 2
    SIMON_DISPLAY_REGION_3,                                     // sets region 3 to 3
    SIMON_DISPLAY_REGION_3,                                     // sets region 3 to 3
    SIMON_DISPLAY_REGION_2,                                     // sets region 2 to 2
    SIMON_DISPLAY_REGION_1,                                     // sets region 1 to 1
    SIMON_DISPLAY_REGION_0};                                    // sets region 0 to 0
#define INCREMENTING_SEQUENCE_MESSAGE1 "Incrementing Sequence"  // Info message.
#define RUN_TEST_COMPLETE_MESSAGE "Runtest() Complete"          // Info message.
#define MESSAGE_TEXT_SIZE 2                                     // Make the text easy to see.
#define TWO_SECONDS_IN_MS 2000                                  // Two second delay.
#define TICK_PERIOD 75                                          // 200 millisecond delay.
#define TEXT_ORIGIN_X 0                                         // Text starts from far left and
#define TEXT_ORIGIN_Y (DISPLAY_HEIGHT/2)                        // middle of screen.

//**********MY_CODE**********//
#define FLASH_ENABLE_FLAG_ON 1                                  // Var for when the FLAG is o
#define FLASH_ENABLE_FLAG_OFF 0                                 // Var for when the enable flag is off
#define TRUE 1                                                  // sets true to 1
#define FALSE 0                                                 // sets false to 0
#define timeDelay 10                                            // time to delay between each box shown
uint8_t flashEnableFlag = 0;                                    // declares the enable flag
uint8_t isCompleteFlag = 0;                                     // declare the complete flag
uint8_t waitCounter = 0;                                        // declare the flash counter
uint8_t series = 0;                                             // Var for what number in the array

enum flashSequence_st_m{                                        // number the state
    init_st,                                                    // start state
    print_st,                                                   // print the button state
```

```
52      wait_st,                                                // wait for the button
  to be printed state
53      delete_st,                                              // delete the button
  state
54      end_st                                                  // end state
55  }flashCurrentState = init_st;                               // set the starting
  state to iniyt_st
56
57  void flashSequence_enable(){                                // Turns on the state
  machine. Part of the interlock.
58      flashEnableFlag = FLASH_ENABLE_FLAG_ON;                 //sets the flag to on
59  }
60
61  void flashSequence_disable(){                               // Turns off the state
  machine. Part of the interlock.
62      flashEnableFlag = FLASH_ENABLE_FLAG_OFF;                //sets the flag to off
63  }
64
65  bool flashSequence_isComplete(){                            // Other state
  machines can call this to determine if this state machine is finished.
66      return isCompleteFlag;
67  }
68
69  void flashSequence_tick(){                                  // Standard tick
  function.
70      switch(flashCurrentState){
71      case init_st:                                           // Moore state action
  for state #1
72          break;                                              // exit state
73      case print_st:                                          // Moore state action
  for state #2
74          simonDisplay_drawSquare(globals_getSequenceValue(series),FALSE);// print out
  the squares
75          break;                                              // exit state
76      case wait_st:                                           // Moore state action
  for state #3
77          waitCounter++;                                      // increment the
  waitCounter
78          break;                                              // exit state
79      case delete_st:                                         // Moore state action
  for state #4
80          simonDisplay_drawSquare(globals_getSequenceValue(series),TRUE); //delete the
  square that was printed
81          break;                                              // exit state
82      case end_st:                                            // Moore state action
  for state #5
83          isCompleteFlag = TRUE;                              // raise the flag
  saying that the square has been printed and erased
84          break;                                              // exit state
85      }
86      switch(flashCurrentState){
87      case init_st:                                           // Mealy transition
  state action for state #1
88          if(flashEnableFlag){
89              flashCurrentState = print_st;                   // transition to next
  state
90          }
91          break;                                              // exit state
```

```
 92    case print_st:                                          // Mealy transition
   state action for state #2
 93        flashCurrentState = wait_st;                        // transition to next
   state
 94        break;                                              // exit state
 95    case wait_st:                                           // Mealy transition
   state action for state #3
 96        if(waitCounter >= timeDelay){                       // wait till counter
   reaches the timerDelay
 97            waitCounter = FALSE;                            // after the state has
   sat for 10 ticks then reset the counter
 98            flashCurrentState = delete_st;                  // transition to next
   state
 99        }
100        break;                                              // exit state
101    case delete_st:                                         // Mealy transition
   state action for state #4
102        if(series >= globals_getSequenceIterationLength()){ // when the series
   reaches the max iteration length of the array
103            flashCurrentState = end_st;                     // transition to next
   state
104        }
105        else{
106            series++;                                       // increment the spot
   in the series
107            flashCurrentState = print_st;                   // transition to next
   state
108        }
109        break;                                              // exit state
110    case end_st:                                            // Mealy transition
   state action for state #5
111        if(!flashEnableFlag){                               // when the enable
   flag is lowered reset back to begining
112            series = FALSE;                                 // reset the flag
113            isCompleteFlag = FALSE;                         // reset the flag
114            flashCurrentState = init_st;                    // go back the the
   initial state
115        }
116        break;                                              // exit state
117    }
118}
119
120    // Print the incrementing sequence message.
121void flashSequence_printIncrementingMessage() {
122  display_fillScreen(DISPLAY_BLACK);  // Otherwise, tell the user that you are
   incrementing the sequence.
123  display_setCursor(TEXT_ORIGIN_X, TEXT_ORIGIN_Y);// Roughly centered.
124  display_println(INCREMENTING_SEQUENCE_MESSAGE1);// Print the message.
125  utils_msDelay(TWO_SECONDS_IN_MS);   // Hold on for 2 seconds.
126  display_fillScreen(DISPLAY_BLACK);  // Clear the screen.
127}
128
129// Run the test: flash the sequence, one square at a time
130// with helpful information messages.
131void flashSequence_runTest() {
132  display_init();                   // We are using the display.
133  display_fillScreen(DISPLAY_BLACK);    // Clear the display.
134  globals_setSequence(flashSequence_testSequence, TEST_SEQUENCE_LENGTH);    // Set the
```

```
   sequence.
135   flashSequence_enable();             // Enable the flashSequence state machine.
136   int16_t sequenceLength = 1;         // Start out with a sequence of length 1.
137   globals_setSequenceIterationLength(sequenceLength);   // Set the iteration length.
138   display_setTextSize(MESSAGE_TEXT_SIZE); // Use a standard text size.
139   while (1) {                           // Run forever unless you break.
140     flashSequence_tick();             // tick the state machine.
141     utils_msDelay(TICK_PERIOD);   // Provide a 1 ms delay.
142     if (flashSequence_isComplete()) {   // When you are done flashing the sequence.
143       flashSequence_disable();          // Interlock by first disabling the state
   machine.
144       flashSequence_tick();             // tick is necessary to advance the state.
145       utils_msDelay(TICK_PERIOD);       // don't really need this here, just for
   completeness.
146       flashSequence_enable();           // Finish the interlock by enabling the state
   machine.
147       utils_msDelay(TICK_PERIOD);       // Wait 1 ms for no good reason.
148       sequenceLength++;                 // Increment the length of the sequence.
149       if (sequenceLength > TEST_SEQUENCE_LENGTH)  // Stop if you have done the full
   sequence.
150         break;
151       // Tell the user that you are going to the next step in the pattern.
152       flashSequence_printIncrementingMessage();
153       globals_setSequenceIterationLength(sequenceLength);  // Set the length of the
   pattern.
154     }
155   }
156   // Let the user know that you are finished.
157   display_fillScreen(DISPLAY_BLACK);              // Blank the screen.
158   display_setCursor(TEXT_ORIGIN_X, TEXT_ORIGIN_Y);// Set the cursor position.
159   display_println(RUN_TEST_COMPLETE_MESSAGE);     // Print the message.
160 }
161
162
```