```c
/*
 * clockDisplay.c
 *
 *  Created on: May 19, 2015
 *      Author: Taylor Cowley
 */

#include <stdio.h>
#include "clockDisplay.h"
#include "supportFiles/display.h"
#include "supportFiles/utils.h"

void draw_triangles();
void add_sec(bool add);
void add_min(bool add);
void add_hour(bool add);
void calculate_where_on_board(uint16_t x, uint16_t y);

// States for the screen touch.
enum touch_screen_place {
    hours_inc,  // the up arrow for hours
    hours_dec,  // the down arrow for hours
    mins_inc,   // up arrow for minutes
    mins_dec,   // down arrow for minutes
    secs_inc,   // up arrow for seconds
    secs_dec,   // down arrow for seconds
    nowhere     // not touched anywhere on screen
} currentPlace = nowhere;

uint32_t hours  = GOOD_LOOKING_CLOCK_NUMBER;    //init to something that looks GOOD
uint32_t mins   = GOOD_LOOKING_CLOCK_NUMBER;    //init to something that looks GOOD
uint32_t secs   = GOOD_LOOKING_CLOCK_NUMBER;    //init to something that looks GOOD
uint32_t prev_hours = hours;    //Standard clock init number
uint32_t prev_mins = mins;      //Standard clock init number
uint32_t prev_secs = secs;      //Standard clock init number


/** Called only once - performs any necessary inits. */
void clockDisplay_init(){
    display_init();                         //Inits the screen
    display_fillScreen(DISPLAY_BLACK);      //Blanks the screen

    display_setTextColor(TEXT_COLOR);       //we want our text green
    display_setTextSize(SIZE);              //sets the text size. To change, change SIZE up above
    display_setCursor(START_X,  START_Y);   //top-left of text

    char initTime[9];                       //the time string is 9 long
    sprintf(initTime, "%02hd:%02hd:%02hd", (int) hours, (int) mins, (int) secs); //print the init
time!
    display_println(initTime);              //display the init time on the screen
    draw_triangles();                       //draws the triangles
}


/** Updates the time display with latest time. */
void clockDisplay_updateTimeDisplay(bool forceUpdateAll){
    char timeChange[3];             //the little string we will use for printing
```

```
    if(hours != prev_hours || forceUpdateAll) {     //we want to print the hours
        display_setCursor(START_X + HOURS_OFFSET,START_Y);  //we are printing the hours
        display_setTextColor(DISPLAY_BLACK);                //the blanking color
        sprintf(timeChange, "%02hd", (int) prev_hours);     //print the old values to blank them
        display_println(timeChange);                        //print the old values to blank them
on screen

        display_setTextColor(TEXT_COLOR);                   //the print color!
        display_setCursor(START_X + HOURS_OFFSET,START_Y);  //we are printing the hours
        sprintf(timeChange, "%02hd", (int) hours);          //print the new hours
        display_println(timeChange);                        //print the new hours on the screen

        prev_hours = hours;                                 //and reset the hours number
    }

    if(mins != prev_mins || forceUpdateAll) {       //we want to print the minutes
        display_setCursor(START_X + MINS_OFFSET,START_Y);   //we are printing the mins
        display_setTextColor(DISPLAY_BLACK);                //the blanking color
        sprintf(timeChange, "%02hd", (int) prev_mins);      //print the old values to blank them
        display_println(timeChange);                        //print the old values to blank them
on screen

        display_setTextColor(TEXT_COLOR);                   //the print color!
        display_setCursor(START_X + MINS_OFFSET,START_Y);   //we are printing the mins
        sprintf(timeChange, "%02hd", (int) mins);           //print the new mins
        display_println(timeChange);                        //print the new mins on screen

        prev_mins = mins;                                   //and reset the hours number
    }

    if(secs != prev_secs || forceUpdateAll) {       //we want to print the seconds
        display_setCursor(START_X + SECS_OFFSET,START_Y);   //we are printing the secs
        display_setTextColor(DISPLAY_BLACK);                //the blanking color
        sprintf(timeChange, "%02hd", (int) prev_secs);      //print the old values to blank them
        display_println(timeChange);                        //print the old values to blank them
on screen

        display_setTextColor(TEXT_COLOR);                   //the print color!
        display_setCursor(START_X + SECS_OFFSET,START_Y);   //we are printing the secs
        sprintf(timeChange, "%02hd", (int) secs);           //print the new secs
        display_println(timeChange);                        //print the new secs on screen

        prev_secs = secs;                                   //and reset the secs number
    }

}


/** Performs the increment or decrement, depending upon the touched region. */
void clockDisplay_performIncDec(){
    int16_t x = 0;  //where x we are touched
    int16_t y = 0;  //where y we are touched
    uint8_t z = 0;  //the pressure of the touch (not used)
    display_getTouchedPoint(&x,&y,&z);  //get the touch data!
    display_clearOldTouchData();        //and clear the touch data for future use
    calculate_where_on_board(x,y);
```

```
    switch(currentPlace) {  //do things depending on where we detect the touch
    case hours_dec:     //touched in the hours down arrow
        add_hour(0);    //decrease hours by 1
        break;
    case hours_inc:     //touched in the hours up arrow
        add_hour(1);    //increase hours by 1
        break;
    case mins_dec:      //touched in the mins down arrow
        add_min(0);     //decrease mins by 1
        break;
    case mins_inc:      //touched in the mins up arrow
        add_min(1);     //increase mins by 1
        break;
    case secs_dec:      //touched in the seconds down arrow
        add_sec(0);     //decrease secs by 1
        break;
    case secs_inc:      //touched in the seconds up arrow
        add_sec(1);     //increase secs by 1
        break;
    case nowhere:
    default:    //this is an error
        printf("we are touched nowhere on the board?\n\r");     //print the error
    }
}

/** Calculates where the coordinate is on the board and stores it in currentPlace */
void calculate_where_on_board(uint16_t x, uint16_t y) {
    if(x < ONE_THIRD_WIDTH) {           //in the hours domain
        if(y > HALF_HEIGHT) {       //in the decrement domain
            currentPlace = hours_dec;       //so we decrement the hours
        }else {                     //in the increment domain
            currentPlace = hours_inc;       //so we increment the hours
        }
    } else if (x > TWO_THIRD_WIDTH) {   //in the seconds domain
        if(y > HALF_HEIGHT) {       //in the decrement domain
            currentPlace = secs_dec;        //so we decrement the seconds
        }else {                     //in the increment domain
            currentPlace = secs_inc;        //so we increment the seconds
        }
    } else {                            //in the minutes domain
        if(y > HALF_HEIGHT) {       //in the decrement domain
            currentPlace = mins_dec;        //so we decrement the minutes
        }else {                     //in the increment domain
            currentPlace = mins_inc;        //so we increment the minutes
        }
    }
}

/** Advances the time forward by 1 second. */
void clockDisplay_advanceTimeOneSecond(){
    if(secs == SEC_MAX) {       //we need to update minutes too!
        secs = 0;
        if(mins == MIN_MAX) {   //we need to update hours too!
            mins = 0;
            if(hours == HOUR_MAX) { //hour overflow!
                hours = 0;          //So goes to zero
            }else{                  //normal
```

```
                hours = hours + 1;  //So add one to hours
            }
        } else {    //Minutes didn't overflow
            mins = mins + 1;
        }
    } else { //seconds didn't overflow
        secs = secs + 1;
    }
    clockDisplay_updateTimeDisplay(0);
}


/** Run a test of clock-display functions. */
void clockDisplay_runTest(){
    utils_msDelay(100);//delays by milliseconds.

    for(int i = 0; i < 100; i++) {  //100 increments is a good idea
        add_sec(1);                 //increment seconds by 1
        utils_msDelay(50);//delays by milliseconds.
    }
    for(int i = 0; i < 100; i++) {  //100 decrements is a good idea
        add_sec(0);                 //decrement seconds by 1
        utils_msDelay(50);//delays by milliseconds.
    }
    for(int i = 0; i < 100; i++) {  //100 increments is a good idea
        add_min(1);                 //increment minutes by 1
        utils_msDelay(50);//delays by milliseconds.
    }
    for(int i = 0; i < 100; i++) {  //100 decrements is a good idea
        add_min(0);                 //decrement minutes by 1
        utils_msDelay(50);//delays by milliseconds.
    }
    for(int i = 0; i < 100; i++) {  //100 increments is a good idea
        add_hour(1);                //increment minutes by 1
        utils_msDelay(50);//delays by milliseconds.
    }
    for(int i = 0; i < 100; i++) {  //100 decrements is a good idea
        add_hour(0);                //decrement hours by 1
        utils_msDelay(50);//delays by milliseconds.
    }
    for(int i = 0; i < 1000; i++) { //1000 seconds advance
        clockDisplay_advanceTimeOneSecond();    //and advance the seconds
        utils_msDelay(50);//delays by milliseconds.
    }


}


/** adds/subtracs one to the secs. Takes care of overflow, etc. subtracts if add is 0 */
void add_sec(bool add){
    if(add == 0) {              //we subtract
        if(secs == 0){      //subtract past 0
            secs = SEC_MAX; //make it to the max (59)
        } else {            //normal subtract
            secs = secs - 1;//subtract 1
        }
```

```
    } else {                    //we add
        if(secs == SEC_MAX) {//add past max
            secs = 0;       //overflow makes it 0
        } else {            //add without overflow
            secs = secs + 1;//add 1
        }
    }
    clockDisplay_updateTimeDisplay(0);//gotta update the display!
}

/** adds/subtracs one to the mins. Takes care of overflow, etc. subtracts if add is 0 */
void add_min(bool add){
    if(add == 0){               //we subtract
        if(mins == 0) {     //subtract past 0
            mins = MIN_MAX; //make it to the max (59)
        } else {            //normal subtract
            mins = mins -1; //subtract 1
        }
    } else {                    //we add
        if(mins == MIN_MAX) {//add past max
            mins = 0;       //overflow makes it 0
        } else {            //add without overflow
            mins = mins + 1;//add 1
        }
    }
    clockDisplay_updateTimeDisplay(0);//gotta update the display!
}

/** adds/subtracs one to the hours. Takes care of overflow, etc. subtracts if add is 0 */
void add_hour(bool add){
    if(add == 0) {          //we subtract
        if(hours == 0) {        //subtract past 0
            hours = HOUR_MAX;   //make it to the max (12)
        } else {                //normal subtract
            hours = hours - 1;  //subtract 1
        }
    } else {                //we add
        if(hours == HOUR_MAX) {//add past max
            hours = 0;      //overflow makes it 0
        } else {            //add without overflow
            hours = hours + 1;//add 1
        }
    }
    clockDisplay_updateTimeDisplay(0);//gotta update the display!
}

/** Draws the up/down triangles everywhere */
void draw_triangles() {
#define TRIANGLE_COLOR DISPLAY_GREEN    //I like this color

    uint16_t x = START_X;           //make a variable for the x coordinates, start with hours

    //hours top- vertices left, top, right, and the triangle color
    display_fillTriangle(   x,              START_Y - TRIANGLE_VERTICAL_SPACE,
                            x + TEXT_WIDTH,     START_Y - TRIANGLE_VERTICAL_SPACE - TEXT_HEIGHT,
                            x + TEXT_WIDTH + TEXT_WIDTH , START_Y - TRIANGLE_VERTICAL_SPACE,
            TRIANGLE_COLOR);
```

```
    //hours bottom- vertices left, bottom, right, and the triangle color
    display_fillTriangle(   x,              START_Y + TEXT_HEIGHT + TRIANGLE_VERTICAL_SPACE,
                            x + TEXT_WIDTH,     START_Y + TRIANGLE_VERTICAL_SPACE + TEXT_HEIGHT +
TEXT_HEIGHT,
                            x + TEXT_WIDTH + TEXT_WIDTH , START_Y + TEXT_HEIGHT +
TRIANGLE_VERTICAL_SPACE,
            TRIANGLE_COLOR);


    x = x + 3 * TEXT_WIDTH;         //move to minutes
    //minutes top- vertices left, top, right, and the triangle color
    display_fillTriangle(   x,              START_Y - TRIANGLE_VERTICAL_SPACE,
                            x + TEXT_WIDTH,     START_Y - TRIANGLE_VERTICAL_SPACE - TEXT_HEIGHT,
                            x + TEXT_WIDTH + TEXT_WIDTH , START_Y - TRIANGLE_VERTICAL_SPACE,
            TRIANGLE_COLOR);
    //minutes bottom- vertices left, bottom, right, and the triangle color
    display_fillTriangle(   x,              START_Y + TEXT_HEIGHT + TRIANGLE_VERTICAL_SPACE,
                            x + TEXT_WIDTH,     START_Y + TRIANGLE_VERTICAL_SPACE + TEXT_HEIGHT +
TEXT_HEIGHT,
                            x + TEXT_WIDTH + TEXT_WIDTH , START_Y + TEXT_HEIGHT +
TRIANGLE_VERTICAL_SPACE,
            TRIANGLE_COLOR);


    x = x + 3 * TEXT_WIDTH;         //move to seconds
    //seconds top - vertices left, top, right, and the triangle color
    display_fillTriangle(   x,              START_Y - TRIANGLE_VERTICAL_SPACE,
                            x + TEXT_WIDTH,     START_Y - TRIANGLE_VERTICAL_SPACE - TEXT_HEIGHT,
                            x + TEXT_WIDTH + TEXT_WIDTH , START_Y - TRIANGLE_VERTICAL_SPACE,
            TRIANGLE_COLOR);
    //seconds bottom- vertices left, bottom, right, and the triangle color
    display_fillTriangle(   x,              START_Y + TEXT_HEIGHT + TRIANGLE_VERTICAL_SPACE,
                            x + TEXT_WIDTH,     START_Y + TRIANGLE_VERTICAL_SPACE + TEXT_HEIGHT +
TEXT_HEIGHT,
                            x + TEXT_WIDTH + TEXT_WIDTH , START_Y + TEXT_HEIGHT +
TRIANGLE_VERTICAL_SPACE,
            TRIANGLE_COLOR);
}
```