```c
#include "clockDisplay.h"
#include "supportFiles/display.h" //needed to access the LCD screen
#include "stdio.h"                //needed to make printf work
#include "supportFiles/utils.h"
#include "string.h"

#define CLOCK_TEXT_SIZE 6
#define HALF_SCREEN_HEIGHT DISPLAY_HEIGHT/2  //half the LCD display height
#define HALF_SCREEN_WIDTH DISPLAY_WIDTH/2    //half the LCD width
#define ONE_THIRD_WIDTH DISPLAY_WIDTH/3
#define TWO_THIRDS_WIDTH (DISPLAY_WIDTH*2)/3
#define RUNNING 1
#define CLOCK_DELAY 1000
#define TOUCH_DELAY 45

#define TRI_HEIGHT_OFFSET 17 //multiply by text size to get height of triangles relative to display center
#define LOWER_TRI_HEIGHT HALF_SCREEN_HEIGHT+(CLOCK_TEXT_SIZE*TRI_HEIGHT_OFFSET)//y-coord of lowest point on lower triangles
#define UPPER_TRI_HEIGHT HALF_SCREEN_HEIGHT-(CLOCK_TEXT_SIZE*TRI_HEIGHT_OFFSET)//y-coord of highest point on higher triangles
#define TRI_BASE_OFFSET 8 //multiply by text size to get y-coord of triangle bases
#define LOWER_TRI_BASE HALF_SCREEN_HEIGHT+(CLOCK_TEXT_SIZE*TRI_BASE_OFFSET) //y-coord of base of lower triangles
#define UPPER_TRI_BASE HALF_SCREEN_HEIGHT-(CLOCK_TEXT_SIZE*TRI_BASE_OFFSET) //y-coord of base of upper triangles

#define LEFT_LEFT_OFFSET 24                                             //multiply by text size to get x-coord of left point of left triangles
#define LEFT_LEFT_X HALF_SCREEN_WIDTH-(CLOCK_TEXT_SIZE*LEFT_LEFT_OFFSET)     //x-coord of left point of left triangles
#define LEFT_MIDDLE_OFFSET 18                                           //multiply by text size to get x-coord of middle point of left triangles
#define LEFT_MIDDLE_X HALF_SCREEN_WIDTH-(CLOCK_TEXT_SIZE*LEFT_MIDDLE_OFFSET)//x-coord of middle point of left triangles
#define LEFT_RIGHT_OFFSET 12                                            //multiply by text size to get x-coord of right point of left triangles
#define LEFT_RIGHT_X HALF_SCREEN_WIDTH-(CLOCK_TEXT_SIZE*LEFT_RIGHT_OFFSET)   //x-coord of right point of left triangles
#define MIDDLE_LEFT_OFFSET 6                                            //multiply by text size to get x-coord of left point of middle triangles
#define MIDDLE_LEFT_X HALF_SCREEN_WIDTH-(CLOCK_TEXT_SIZE*MIDDLE_LEFT_OFFSET)//x-coord of left point of middle triangle

#define MIDDLE_RIGHT_OFFSET 6 //multiply by text size to get x-coord of right point of middle triangles
#define MIDDLE_RIGHT_X HALF_SCREEN_WIDTH+(CLOCK_TEXT_SIZE*MIDDLE_RIGHT_OFFSET)//x-coord of right point of middle triangles
#define RIGHT_LEFT_OFFSET 12 //multiply by text size to get x-coord of left point of right triangles
#define RIGHT_LEFT_X HALF_SCREEN_WIDTH+(CLOCK_TEXT_SIZE*RIGHT_LEFT_OFFSET) //x-coord of left point of right triangles
#define RIGHT_MIDDLE_OFFSET 18 //multiply by text size to get x-coord of middle point of right triangles
#define RIGHT_MIDDLE_X
```

```c
HALF_SCREEN_WIDTH+(CLOCK_TEXT_SIZE*RIGHT_MIDDLE_OFFSET)//x-coord of middle point of right triangles
#define RIGHT_RIGHT_OFFSET 24 //multiply by text size to get x-coord of right point of right triangles
#define RIGHT_RIGHT_X HALF_SCREEN_WIDTH+(CLOCK_TEXT_SIZE*RIGHT_RIGHT_OFFSET) //x-coord of right point of right triangles

#define TEXT_CURSOR_OFFSET_Y 4                                          //multiply by text size to get y position for text cursor
#define TEXT_Y HALF_SCREEN_HEIGHT-(CLOCK_TEXT_SIZE*TEXT_CURSOR_OFFSET_Y)//y position for text cursor
#define CURSOR_MULTIPLIER 6                                             //scaling factor needed to get the right cursor x-coord
                                                                            //when comparing oldBuffer and newBuffer character by character

#define INC_HOURS 0       //identifier assigned to top left rectangle of display
#define DEC_HOURS 1       //identifier assigned to bottom left rectangle of display
#define INC_MIN 2         //identifier assigned to top middle rectangle of display
#define DEC_MIN 3         //identifier assigned to bottom middle rectangle of display
#define INC_SEC 4         //identifier assigned to top right rectangle of display
#define DEC_SEC 5         //identifier assigned to bottom right rectangle of display
#define NO_TOUCH_ERROR -1 //used to determine if a touch was detected even though it was not actually touched

//all values based on 12-hour clock model
#define HOURS_MIN 1  //lowest value the hours can have
#define MIN_MIN 0    //lowest value the minutes can have
#define SEC_MIN 0    //lowest value the seconds can have
#define HOURS_MAX 12 //max val for hours
#define MIN_MAX 59   //max val for minutes
#define SEC_MAX 59   //max val for seconds

#define ERROR_STRING "TOUCH ERROR\n\r"

#define BUFFER_SIZE 9 //the size of the clock strings we will be displaying

static int8_t hours = HOURS_MIN; //initialize hours to 1
static int8_t min = MIN_MIN;     //initialize minutes to 0
static int8_t sec = SEC_MIN;     //initialize seconds to 0
static char newBuffer[9];        //this will hold the updated string after an inc/dec function call or second increment
static char oldBuffer[9];        //this holds a copy of the previous string in order to compare it with the new one




// Called only once - performs any necessary inits.
// This is a good place to draw the triangles and any other
// parts of the clock display that will never change.
void clockDisplay_init(){
    display_init();                         //Must init all of the software and underlying hardware for LCD.
    display_fillScreen(DISPLAY_BLACK);      //Blank the screen.
    display_setTextSize(CLOCK_TEXT_SIZE);   //set text size
    display_setTextColor(DISPLAY_GREEN);    //text color given by lab specs

    display_fillTriangle(LEFT_LEFT_X, UPPER_TRI_BASE, LEFT_MIDDLE_X, UPPER_TRI_HEIGHT, //create the triangle for incrementing hours
            LEFT_RIGHT_X, UPPER_TRI_BASE, DISPLAY_GREEN);
    display_fillTriangle(MIDDLE_LEFT_X, UPPER_TRI_BASE, HALF_SCREEN_WIDTH, UPPER_TRI_HEIGHT, //create the triangle for incrementing minutes
            MIDDLE_RIGHT_X, UPPER_TRI_BASE, DISPLAY_GREEN);
    display_fillTriangle(RIGHT_LEFT_X, UPPER_TRI_BASE, RIGHT_MIDDLE_X, UPPER_TRI_HEIGHT,     //create the triangle for incrementing seconds
            RIGHT_RIGHT_X, UPPER_TRI_BASE, DISPLAY_GREEN);

    display_fillTriangle(LEFT_LEFT_X, LOWER_TRI_BASE, LEFT_MIDDLE_X, LOWER_TRI_HEIGHT, //create triangle for decrementing hours
                LEFT_RIGHT_X, LOWER_TRI_BASE, DISPLAY_GREEN);
    display_fillTriangle(MIDDLE_LEFT_X, LOWER_TRI_BASE, HALF_SCREEN_WIDTH, LOWER_TRI_HEIGHT, //create triangle for decrementing minutes
                MIDDLE_RIGHT_X, LOWER_TRI_BASE, DISPLAY_GREEN);
    display_fillTriangle(RIGHT_LEFT_X, LOWER_TRI_BASE, RIGHT_MIDDLE_X, LOWER_TRI_HEIGHT,     //create triangle for decrementing seconds
                RIGHT_RIGHT_X, LOWER_TRI_BASE, DISPLAY_GREEN);

    sprintf(oldBuffer, "%2hd:%02hd:%02hd", hours, min, sec); //what will first appear on the screen when initialized
    display_setCursor(LEFT_LEFT_X, TEXT_Y);                  //set cursor based on display midpoint
    display_print(oldBuffer);                                //printing to the display
}

// Updates the time display with latest time, making sure to update only those digits that
// have changed since the last update.
// if forceUpdateAll is true, update all digits.
void clockDisplay_updateTimeDisplay(bool forceUpdateAll){
    if(newBuffer != oldBuffer){ //only need to update time on display if the time to be displayed is different from current time
        for(int i = 0; i < BUFFER_SIZE; i++){ //for loop to go through the two buffers character by character
            if(oldBuffer[i] != newBuffer[i]){ //only enter if statement if the characters at ith position in buffer are different
                display_drawChar(LEFT_LEFT_X+(i*CLOCK_TEXT_SIZE)*CURSOR_MULTIPLIER, TEXT_Y, oldBuffer[i], DISPLAY_BLACK, DISPLAY_BLACK, CLOCK_TEXT_SIZE); //black out the older character so it can't be seen

                display_drawChar(LEFT_LEFT_X+(i*CLOCK_TEXT_SIZE)*CURSOR_MULTIPLIER, TEXT_Y, newBuffer[i], DISPLAY_GREEN, DISPLAY_BLACK, CLOCK_TEXT_SIZE); //draw over old character with green text
            }
        }
        strcpy(oldBuffer, newBuffer); //built in string function to copy the updated buffer just put on display to the old buffer
    }
}


//helper function to get the coordinates of the touch and determine which region of the display it is
int8_t getQuadrant(int16_t x, int16_t y){ //we need the x and y-coords to determine region of the display

```

```c

//helper function for ensuring the sec, min, and hours rollover when necessary
void incDecRollover(){
    if(sec > SEC_MAX){ //are the seconds greater than 59?
        sec = SEC_MIN; //set it back to 0
        if(min == MIN_MAX){ //also check are the minutes at 59?
            min = MIN_MIN; //set them back to 0
            hours++; //in which case we also need to increment the hours
        }
        else{
            min++; //if the minutes aren't yet at 59, increment the minutes
        }
    }
    else if(sec < SEC_MIN){ //are the seconds less than 0?
        sec = SEC_MAX; //set it back to 59
        if(min == MIN_MIN){ //also check if minutes are at 0
            min = MIN_MAX; //if so, set them back to 59
            hours--; //in which case we also need to decrement the hours
        }
        else{
            min--; //if the minutes are't yet at 0, just decrement them by 1
        }
    }


    if(min > MIN_MAX){ //are the minutes greater than 59?
        min = MIN_MIN; //set them back to 0
        if(hours == HOURS_MAX){ //also check if hours are at 12
            hours = HOURS_MIN; //if so, set that back to 1
        }
        else{
            hours++; //if the hours aren't at 12 yet, just increment them by 1
        }
    }
    else if(min < MIN_MIN){ //are minutes less than 0?
        min = MIN_MAX; //set them back to 59
        if(hours == HOURS_MIN){ //also check if hours are at 1
            hours = HOURS_MAX; //if so set them back to 12
        }
        else{
            hours--; //if hours aren't at 1 yet, just decrement them by 1
        }
    }

    //final hours check after incrementing or decrementing may have happened
    if(hours > HOURS_MAX){ //are hours greater than 12?
        hours = HOURS_MIN; //set them back to 1
    }
    else if(hours < HOURS_MIN){ //are hours less than 1?
        hours = HOURS_MAX; //set them back to 12
    }
}

// Reads the touched coordinates and performs the increment or decrement,
// depending upon the touched region.
void clockDisplay_performIncDec(){
    int16_t x, y;  //need x,y coordinates
    uint8_t z;     //need z for touch pressure as argument for function. Will not be
```

```c
used again.
    display_getTouchedPoint(&x, &y, &z); //used to get the x,y coordinates for the touch
    int8_t whereTouched = getQuadrant(x,y); //whereTouched is assigned to the correct region of the display based on

    switch(whereTouched){ //switch to check the quadrant and carry out corresponding action
    case INC_HOURS:
        hours++; //increments hours if the top left region was touched
        break;
    case DEC_HOURS:
        hours--; //decrements hours if the bottom left region was touched
        break;
    case INC_MIN:
        min++; //increments minutes if top middle region was touched
        break;
    case DEC_MIN:
        min--; //decrements minutes if bottom middle region was touched
        break;
    case INC_SEC:
        sec++; //increments seconds if top right region was touched
        break;
    case DEC_SEC:
        sec--; //decrements seconds if bottom right region was touched
        break;
    case NO_TOUCH_ERROR:
        printf(ERROR_STRING); //if there was a touch error, then print it out
        break;
    default:
        break;
    }

    incDecRollover(); //after the right thing is incremented or decremented, we need to check for rollover
    sprintf(newBuffer, "%2hd:%02hd:%02hd", hours, min, sec); //send the updated hours/minutes/seconds into the newBuffer character array
    clockDisplay_updateTimeDisplay(false); //now we update the display with the updated time character array
}

// Advances the time forward by 1 second and update the display.
void clockDisplay_advanceTimeOneSecond(){
    sec++; //always increment the seconds first. All the checks for whether it should actually be incremented or not have already been done up to this point
    incDecRollover(); //we have to check for rollover again in case it changed the minutes as well
    sprintf(newBuffer, "%2hd:%02hd:%02hd", hours, min, sec); //send the updated hours/minutes/seconds into the newBuffer character array
    clockDisplay_updateTimeDisplay(false); //now we update the display with the updated time character array
}

// Run a test of clock-display functions.
void clockDisplay_runTest(){
    clockDisplay_init(); //must initialize the clock display, which includes the triangles and setting the time as 1:00:00

    while(RUNNING){ //run as long as we want it to, constantly looking for new touch input
        clockDisplay_performIncDec(); //this will translate the touch into which part of the time to be incremented or decremented
        clockDisplay_advanceTimeOneSecond(); //this must be happening whether we are getting touch input or not
    }
}

```