```c
 * clockDisplay.c
#include "clockDisplay.h" //include the header file for this .c file
#include "supportFiles/display.h" //include the display header so you can display to the board
#include "supportFiles/utils.h" //this includes a file that makes the clock work
#include "stdio.h"
//#include <stdio.h>

#define CLK_TEXT_SIZE 6 // set the text size to 6
#define CLOCK_SEPERATOR ":" //is the colon to separate the hours min and sec
#define DISPLAY_MIDDLE_X (DISPLAY_WIDTH / 2) //finds the middle of the board in the X direction 160 pixels
#define DISPLAY_MIDDLE_Y (DISPLAY_HEIGHT / 2) //finds the middle of the board in the Y direction 120 pixels
#define LINE_0_X (DISPLAY_WIDTH / 3)
#define LINE_1_X (2 * (DISPLAY_WIDTH / 3))

#define TEXT_0_CURSOR_X (DISPLAY_MIDDLE_X - ((135/6) * CLK_TEXT_SIZE))   //X location for hours
#define TEXT_1_CURSOR_X (DISPLAY_MIDDLE_X - ((30/6) * CLK_TEXT_SIZE))      //X location for minutes
#define TEXT_2_CURSOR_X (DISPLAY_MIDDLE_X + ((75/6) * CLK_TEXT_SIZE))   //X location for seconds
#define TEXT_0_COLON_X (DISPLAY_MIDDLE_X - ((70/6) * CLK_TEXT_SIZE))   //X location for hours to minutes colon 65
#define TEXT_1_COLON_X (DISPLAY_MIDDLE_X + ((40/6) * CLK_TEXT_SIZE))    //X location for minutes to seconds colon
#define TEXT_0_CURSOR_Y (DISPLAY_MIDDLE_Y - ((20/6) * CLK_TEXT_SIZE))   //Y location for all text

#define FAR_SCALER_X (105/6) // A scaler in the X direction for the farthest X point (divide by 6 because 6 is the standard text size)
#define FAR_SCALER_Y (90/6) // A scaler in the Y direction for the farthest Y point (divide by 6 because 6 is the standard text size)
#define MID_SCALER_X (30/6) // A scaler in the X direction for the mid X point (divide by 6 because 6 is the standard text size)
#define MID_SCALER_Y (40/6) // A scaler in the Y direction for the mid Y point (divide by 6 because 6 is the standard text size)
#define MID_SCALER_BASE_X (60/6) // A scaler in the X direction for the farthest X point (divide by 6 because 6 is the standard text size)


// TRIANGLE 0
#define TRI_0_X0 (DISPLAY_MIDDLE_X - (FAR_SCALER_X * CLK_TEXT_SIZE))   //Locate the X coordinate for the top of the triangle
#define TRI_0_Y0 (DISPLAY_MIDDLE_Y - (FAR_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the top of the triangle
#define TRI_0_X1 (TRI_1_X1 -         (FAR_SCALER_X * CLK_TEXT_SIZE))       //Locate the X coordinate for the left side of the triangle
#define TRI_0_Y1 (DISPLAY_MIDDLE_Y - (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the left side of the triangle
#define TRI_0_X2 (TRI_0_X1 +         (MID_SCALER_BASE_X * CLK_TEXT_SIZE)) //Locate the X coordinate for the right side of the triangle
#define TRI_0_Y2 (DISPLAY_MIDDLE_Y - (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the right side of the triangle

// TRIANGLE 1
#define TRI_1_X0 (DISPLAY_MIDDLE_X)                               //Locate the X
```

```c
coordinate for the top of the triangle
#define TRI_1_Y0 (DISPLAY_MIDDLE_Y - (FAR_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the top of the triangle
#define TRI_1_X1 (DISPLAY_MIDDLE_X - (MID_SCALER_X * CLK_TEXT_SIZE))    //Locate the X coordinate for the left side of the triangle
#define TRI_1_Y1 (DISPLAY_MIDDLE_Y - (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the left side of the triangle
#define TRI_1_X2 (DISPLAY_MIDDLE_X + (MID_SCALER_X * CLK_TEXT_SIZE))    //Locate the X coordinate for the right side of the triangle
#define TRI_1_Y2 (DISPLAY_MIDDLE_Y - (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the right side of the triangle

// TRIANGLE 2
#define TRI_2_X0 (DISPLAY_MIDDLE_X + (FAR_SCALER_X * CLK_TEXT_SIZE))   //Locate the X coordinate for the top of the triangle
#define TRI_2_Y0 (DISPLAY_MIDDLE_Y - (FAR_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the top of the triangle
#define TRI_2_X1 (TRI_1_X1 +         (FAR_SCALER_X * CLK_TEXT_SIZE))   //Locate the X coordinate for the left side of the triangle
#define TRI_2_Y1 (DISPLAY_MIDDLE_Y - (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the left side of the triangle
#define TRI_2_X2 (TRI_2_X1 +         (MID_SCALER_BASE_X * CLK_TEXT_SIZE))    //Locate the X coordinate for the right side of the triangle
#define TRI_2_Y2 (DISPLAY_MIDDLE_Y - (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the right side of the triangle

// TRIANGLE 3
#define TRI_3_X0 (DISPLAY_MIDDLE_X - (FAR_SCALER_X * CLK_TEXT_SIZE))   //Locate the X coordinate for the top of the triangle
#define TRI_3_Y0 (DISPLAY_MIDDLE_Y + (FAR_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the top of the triangle
#define TRI_3_X1 (TRI_1_X1 -         (FAR_SCALER_X * CLK_TEXT_SIZE))   //Locate the X coordinate for the left side of the triangle
#define TRI_3_Y1 (DISPLAY_MIDDLE_Y + (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the left side of the triangle
#define TRI_3_X2 (TRI_0_X1 +         (MID_SCALER_BASE_X * CLK_TEXT_SIZE))    //Locate the X coordinate for the right side of the triangle
#define TRI_3_Y2 (DISPLAY_MIDDLE_Y + (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the right side of the triangle

// TRIANGLE 4
#define TRI_4_X0 (DISPLAY_MIDDLE_X)                                 //Locate the X coordinate for the top of the triangle
#define TRI_4_Y0 (DISPLAY_MIDDLE_Y + (FAR_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the top of the triangle
#define TRI_4_X1 (DISPLAY_MIDDLE_X - (MID_SCALER_X * CLK_TEXT_SIZE))    //Locate the X coordinate for the left side of the triangle
#define TRI_4_Y1 (DISPLAY_MIDDLE_Y + (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the left side of the triangle
#define TRI_4_X2 (DISPLAY_MIDDLE_X + (MID_SCALER_X * CLK_TEXT_SIZE))    //Locate the X coordinate for the right side of the triangle
#define TRI_4_Y2 (DISPLAY_MIDDLE_Y + (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the right side of the triangle

//TRIANGLE 5
#define TRI_5_X0 (DISPLAY_MIDDLE_X + (FAR_SCALER_X * CLK_TEXT_SIZE))   //Locate the X coordinate for the top of the triangle
#define TRI_5_Y0 (DISPLAY_MIDDLE_Y + (FAR_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y
```

```
coordinate for the top of the triangle
#define TRI_5_X1 (TRI_1_X1 +         (FAR_SCALER_X * CLK_TEXT_SIZE))   //Locate the X coordinate for the left side of the triangle
#define TRI_5_Y1 (DISPLAY_MIDDLE_Y + (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the left side of the triangle
#define TRI_5_X2 (TRI_2_X1 +         (MID_SCALER_BASE_X * CLK_TEXT_SIZE))    //Locate the X coordinate for the right side of the triangle
#define TRI_5_Y2 (DISPLAY_MIDDLE_Y + (MID_SCALER_Y * CLK_TEXT_SIZE))    //Locate the Y coordinate for the right side of the triangle

//*********VAR_ FOR_TIME_KEEPING*********
#define HOURS_MAX 12                            //MAX number that hours can be
#define MAX_MINUTES_AND_SECONDS 59              //MAX number minutes and seconds can be
#define MIN_TIME 0                              //MIN number that time can be
#define NUM_OF_LOOPS 10                         //number of time the for loops will increment the clock
#define DELAY_X_TEN 100                         //how long it take to tick when going 10x as fast
#define DELAY_NORMAL 500                        //how long it take to tick when going at a normal speed
#define FALSE 0
#define TRUE 1
    uint8_t hours = 12; //var to store and init the hours string
    uint8_t minutes = 59; //var to store and init the minutes string
    uint8_t seconds = 59; //var to store and init the seconds string
    uint8_t updatedHours = 12; //var to store and init the  updated hours string
    uint8_t updatedMinutes = 59; //var to store and init the updated minutes string
    uint8_t updatedSeconds = 59; //var to store and init the  updated seconds string
    char hourString[2];          // char to display the hours on the board
    char minuteString[2];        // char to display the minutes on the board
    char secondString[2];        // char to display the seconds on the board

void clockDisplay_init(){

    display_init(); //initializes the display board
    display_fillScreen(DISPLAY_BLACK); //displays the screen as black
    display_setTextSize(CLK_TEXT_SIZE);   //set the text height
    display_setCursor(TEXT_0_CURSOR_X, TEXT_0_CURSOR_Y);    //set the text cursor for hours

    display_setTextColor(DISPLAY_GREEN, DISPLAY_BLACK);     //set the text color to green and the background to black
    display_setCursor(TEXT_0_COLON_X, TEXT_0_CURSOR_Y);     //set the text to write a colon between the hours and min
    display_println(CLOCK_SEPERATOR);                       //set the colon between hours and min
    display_setCursor(TEXT_1_COLON_X, TEXT_0_CURSOR_Y);     //set the text to write a colon between the min and sec
    display_println(CLOCK_SEPERATOR);                       //set the colon between min and sec

    display_fillTriangle(TRI_1_X0, TRI_1_Y0, TRI_1_X1, TRI_1_Y1, TRI_1_X2, TRI_1_Y2, DISPLAY_GREEN); //create and fill triangle 0
    display_fillTriangle(TRI_0_X0, TRI_0_Y0, TRI_0_X1, TRI_0_Y1, TRI_0_X2, TRI_0_Y2, DISPLAY_GREEN); //create and fill triangle 1
    display_fillTriangle(TRI_2_X0, TRI_2_Y0, TRI_2_X1, TRI_2_Y1, TRI_2_X2, TRI_2_Y2, DISPLAY_GREEN); //create and fill triangle 2
    display_fillTriangle(TRI_3_X0, TRI_3_Y0, TRI_3_X1, TRI_3_Y1, TRI_3_X2, TRI_3_Y2,
```

```c
DISPLAY_GREEN); //create and fill triangle 3
    display_fillTriangle(TRI_4_X0, TRI_4_Y0, TRI_4_X1, TRI_4_Y1, TRI_4_X2, TRI_4_Y2, DISPLAY_GREEN); //create and fill triangle 4
    display_fillTriangle(TRI_5_X0, TRI_5_Y0, TRI_5_X1, TRI_5_Y1, TRI_5_X2, TRI_5_Y2, DISPLAY_GREEN); //create and fill triangle 5

    clockDisplay_updateTimeDisplay(TRUE);

}

// 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((hours != updatedHours) || forceUpdateAll){
        hours = updatedHours; // update var hours to new hours
        sprintf(hourString, "%2d", hours); // save hours to a string
        display_setCursor(TEXT_0_CURSOR_X, TEXT_0_CURSOR_Y); //set cursor to the hours coordinate
        display_println(hourString); //print hours to the screen
    }
    if((minutes != updatedMinutes) || forceUpdateAll){
        minutes = updatedMinutes; // update var hours to new hours
        sprintf(minuteString, "%02d", minutes); // save hours to a string
        display_setCursor(TEXT_1_CURSOR_X, TEXT_0_CURSOR_Y); //set cursor to the hours coordinate
        display_println(minuteString); //print minutes to the screen
    }
    if((seconds != updatedSeconds) || forceUpdateAll){
        seconds = updatedSeconds; // update var hours to new hours
        sprintf(secondString, "%02d", seconds); // save hours to a string
        display_setCursor(TEXT_2_CURSOR_X, TEXT_0_CURSOR_Y); //set cursor to the hours coordinate
        display_println(secondString); //print seconds to the screen
    }
}
// Reads the touched coordinates and performs the increment or decrement,
// depending upon the touched region.
void clockDisplay_performIncDec(){
    int16_t x = 0; //Initialize x var (x location on the board)
    int16_t y = 0; //Initialize y var (y location on the board)
    uint8_t z = 0; //Initialize z var (z location on the board)
    display_getTouchedPoint(&x, &y, &z); //inputs the coordinates of the screen touch
    if (y < DISPLAY_MIDDLE_Y){ // check to see if the touch is in the top half of the screen
        if(x < LINE_0_X){ //check if x is in 0 box
            updatedHours += 1;// increment hours by one
            if (updatedHours >= (HOURS_MAX + 1)){ // check if hours are above MAX
                updatedHours = ((MIN_TIME) + 1); // if hours are above MAX, reset it to 1
            }
        }
        else if((x > LINE_0_X) && (x < LINE_1_X)){ //check if x is in 1 box
            updatedMinutes += 1;// increment hours by one
            if (updatedMinutes >= (MAX_MINUTES_AND_SECONDS + 1)){ // check if minutes are above MAX
                updatedMinutes = (MIN_TIME);// if minutes are above MAX, reset it to 0
            }
        }
        else if((x > LINE_1_X) && (x < DISPLAY_WIDTH)){ //check if x is in 2 box
            updatedSeconds += 1;// increment hours by one
            if (updatedSeconds >= (MAX_MINUTES_AND_SECONDS + 1)){ // check if seconds are above MAX
                updatedSeconds = (MIN_TIME); // if seconds are above MAX, reset it to 0
            }
        }
        else{
            printf("%s", "ERROR invalid board touch"); // print out error if it touch is invalid
        }
    }
    else if(y > DISPLAY_MIDDLE_Y){  // check to see if the touch is in the bottom half of the screen
        if(x < LINE_0_X){ //check if x is in 3 box
            updatedHours -= 1; // decrement hours by one
            if (updatedHours == MIN_TIME){ // check if hours are below MIN time
                updatedHours = HOURS_MAX; // if hours are below, reset it to max hours
            }
        }
        else if((x > LINE_0_X) && (x < LINE_1_X)){ //check if x is in 4 box
            if (updatedMinutes == MIN_TIME){  // check if minutes are below MIN time
                updatedMinutes = MAX_MINUTES_AND_SECONDS;  // if minutes are below, reset it to max minutes
            }
            else{
                updatedMinutes -= 1; // decrement minutes by one
            }
        }
        else if((x > LINE_1_X) && (x < DISPLAY_WIDTH)){ //check if x is in 5 box
            if (updatedSeconds == MIN_TIME){  // check if seconds are below MIN time
                updatedSeconds = MAX_MINUTES_AND_SECONDS;  // if seconds are below, reset it to max second
            }
            else{
                updatedSeconds -= 1; // decrement second by one
            }
        }
        else{
            printf("%s", "ERROR invalid board touch"); // print out error statement if the touch isn't valid
        }
    }
    else{
        printf("%s", "ERROR invalid board touch");  // print out error statement if the touch isn't valid
    }
    clockDisplay_updateTimeDisplay(FALSE);
}

// Advances the time forward by 1 second and update the display.
void clockDisplay_advanceTimeOneSecond(){
    updatedSeconds += 1; // advance seconds 1
    if (updatedSeconds >= MAX_MINUTES_AND_SECONDS + 1){  // check if second is equal to or greater than 59
```

```c
        updatedSeconds = MIN_TIME; //set the seconds back to 0
        updatedMinutes += 1;// advance minutes 1
    }
    if (updatedMinutes >= MAX_MINUTES_AND_SECONDS + 1){  // check if minutes is equal to or greater than 59
        updatedMinutes = MIN_TIME; //set the minutes back to 0
        updatedHours += 1; // advance hours 1
    }
    if (updatedHours >= HOURS_MAX + 1){ // check if hours is equal to or greater than 13
        updatedHours = ((MIN_TIME) + 1); //set the seconds back to 1 (because hours can be 0)
    }
};

// Run a test of clock-display functions.
void clockDisplay_runTest(){
    clockDisplay_init(); // Initialize the clock
    clockDisplay_updateTimeDisplay(TRUE); //update the display to 12:59:59
    for (int i = 0; i < NUM_OF_LOOPS; i++){ //decrement hours minutes and seconds 10 times
        updatedHours -= 1; // decrement hours
        updatedMinutes -= 1; // decrement minutes
        updatedSeconds -= 1; // decrement second
        clockDisplay_updateTimeDisplay(FALSE); //update the display
        utils_msDelay(DELAY_NORMAL);// wait 500 m/s
    }
    for (int i = 0; i < NUM_OF_LOOPS; i++){//increment hours minutes and seconds 10 times
        updatedHours += 1; // increment hours
        updatedMinutes += 1; // increment minutes
        updatedSeconds += 1; // increment second
        clockDisplay_updateTimeDisplay(FALSE); //update the display
        utils_msDelay(DELAY_NORMAL); // wait 500 m/s
    }
    for (int i = 0; i < (NUM_OF_LOOPS*NUM_OF_LOOPS); i++){
        clockDisplay_advanceTimeOneSecond(); //increment every tick
        clockDisplay_updateTimeDisplay(FALSE); // update the display
        utils_msDelay(DELAY_X_TEN); //increment at x10
    }
};

```