```c
#include <stdio.h>
#include "system.h"
#include <unistd.h>
#include <inttypes.h>
#include <stdlib.h>
#include "lcdAPI.h"

volatile uint16_t backBuffer[X_SIZE*Y_SIZE];

volatile unsigned int *frontBufferControl = (unsigned int*)FRONT_BUFFER_CONTROLLER;
volatile unsigned int *backBufferControl = (unsigned int*)BACK_BUFFER_CONTROLLER;
volatile unsigned int *resolutionControl = (unsigned int*)RESOLUTION_CONTROLLER;
volatile unsigned int *statusControl = (unsigned int*)STATUS_CONTROLLER;

volatile unsigned int *touchStatus = (unsigned int*)TOUCH_STATUS;
volatile unsigned int *touchPosition = (unsigned int*)TOUCH_COORDINATES;

/**
 * lcd_init
 * Purpose: initializes the LCD screen
 * args:
 *      nothing
 * returns:
 *      nothing
 */
void lcd_init(){

    *backBufferControl = backBuffer;
    //swap front and back buffer
    *frontBufferControl = backBuffer;
    while(((*statusControl)&BUFFER_SWAP_STATUS)&NOT_SWAPPED){

    }
    *backBufferControl = backBuffer;

}

/**
 * touch_init
 * Purpose: initializes the touch capabilities of the screen
 * args:
 *      nothing
 * returns:
 *      nothing
 */
void touch_init(){
    *touchStatus |= TOUCH_ENABLE;
}

/**
 * poll_position
 * Purpose: To poll for new coordinates
 * args:
 *      coords - an array to hold both coordinates
 * returns:
 *      X and Y coordinates in the coords array
 */
void poll_position(int* coords){
    coords[0] = (*touchPosition & (X_COORD));
    coords[1] = (*touchPosition & (Y_COORD))>>Y_CORRECTION;
    usleep(100);
}

/**
 * lcd_fill
 * Purpose: Fills the entire LCD screen
 * args:
 *      color - the 24-bit color code
 * returns:
```

```c
 *      nothing
 */
void lcd_fill(unsigned int color){
    unsigned int blue = (color&BLUE_MASK)>>3;
    unsigned int green = (color&GREEN_MASK)>>10;
    unsigned int red = (color&RED_MASK)>>19;
    unsigned int adjColor = blue|(green<<5)|(red<<11);

    for(int i = 0; i < (X_SIZE*Y_SIZE); i++){
        backBuffer[i]=adjColor;
    }

}

/**
 * set_pixel
 * Purpose: Colors one pixel at the given coordinates with the given 24-bit color
 * args:
 *      x - the X coordinate
 *      y - the Y coordinate
 *      color - the 24-bit color code
 * returns:
 *      nothing
 */
void set_pixel(unsigned int x, unsigned int y, unsigned int color){
    unsigned int blue = (color&BLUE_MASK)>>3;
    unsigned int green = (color&GREEN_MASK)>>10;
    unsigned int red = (color&RED_MASK)>>19;
    unsigned int adjColor = blue|(green<<5)|(red<<11);
    backBuffer[(y*320)+x]=adjColor;
}

/**
 * draw_rectangle
 * Purpose: Draws a rectangle starting at a given x and y of given length and width. It will either fill or outline
 *          the rectangle in a given 24-bit color
 * args:
 *      x - the starting X coordinate
 *      y - the starting Y coordinate
 *      width - the width of the rectangle
 *      height - the height of the rectangle
 *      color - the 24-bit color code
 *      fill - 1 to fill, else outline
 * returns:
 *      nothing
 */
void draw_rectangle(unsigned int x, unsigned int y, unsigned int width, unsigned int height, unsigned int color, int fill){
    unsigned int blue = (color&BLUE_MASK)>>3;
    unsigned int green = (color&GREEN_MASK)>>10;
    unsigned int red = (color&RED_MASK)>>19;
    unsigned int adjColor = blue|(green<<5)|(red<<11);
    if(fill == 1){
        for(int j = 0; j < height; j++){
                for(int i = (((y+j)*320)+x); i < ((((y+j)*320)+x)+width); i++){
                backBuffer[i]=adjColor;
            }
        }
    } else {
        for(int j = 0; j < height; j++){
            for(int i = (((y+j)*320)+x); i < ((((y+j)*320)+x)+width); i++){
                if(j == 0 | i == (((y+j)*320)+x) | i == ((((y+j)*320)+x)+width-1) | j == (height-1)){
                    backBuffer[i]=adjColor;
                }
            }
        }
    }
}
```