EE445L Lab 1 Preparation

**fixed.c**

// fixed.c

// Created by: Sijin Woo

// UTEID: sw34773

// Section: 15665

// Contact: sijin.woo@utexas.edu

// Date Created: 1/18/2018

// Date of Last Revision: 1/21/2018

// Description: source file for the outputs onto the ST7735 LCD - Fixed point decimal, fixed point binary,

// xy graph initialization, scatter plot

#include "fixed.h"

#include "ST7735.h"

#define OUTPUT\_CMD\_LENGTH 6 // Number of characters that will be on the LCD screen to make everything look nice

// Private function prototypes:

int32\_t absValue(int32\_t n); // absolute value of n

void plotXYpoint(int32\_t x, int32\_t y); // Plots the xy point onto the LCD

void drawPixels(uint32\_t x, uint32\_t y, uint32\_t size, uint16\_t color); // Draws pixel onto LCD according to size

// Private gobal variables:

static int32\_t XMinLim = 0; // MIN value of x point that can be graphed

static int32\_t XMaxLim = 0; // MAX value of x point that can be graphed

static int32\_t YMinLim = 0; // MIN value of y point that can be graphed

static int32\_t YMaxLim = 0; // MAX value of y point that can be graphed

static uint32\_t XScale = 0; // scaling of x axis (distance between each pixel)

static uint32\_t YScale = 0; // scaling of y axis

static uint32\_t XOrigin = 0; // x point of origin of graph on LCD (0 <= x < 128)

static uint32\_t YOrigin = 32; // y point of origin of graph on LCD (32 <= y < 160)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: ST7735\_sDecOut3

// If the signed 32-bit number is less than -9999 or greater than 9999, then \*.\*\*\* will be outputted.

// If number within acceptable range, then the decimal is shifted to the left three places.

void ST7735\_sDecOut3(int32\_t n){

if(n > 9999 || n < -9999){

char nErrorStr[] = " \*.\*\*\*"; // If error, \*.\*\*\* is outputted no matter sign

ST7735\_OutString(nErrorStr); // Output to LCD

}else{

char nFixedPointStr[] = " . "; // Template for what is going to be outputted

// If n is negative, place negative sign

if(n < 0){

nFixedPointStr[0] = '-';

n = -n;

}

for(int i = OUTPUT\_CMD\_LENGTH - 1; i > 0; i--){

// Since at index 2 there must be a decimal, skip if this occurs

if(i != 2){

nFixedPointStr[i] = n % 10 + '0';

n /= 10;

}

}

ST7735\_OutString(nFixedPointStr); // Output to LCD

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: ST7735\_uBinOut8

// The following function prints a fixed point unsigned binary representation of the unsigned 32-bit input number

// Input: unsigned 32-bit number

// Output: None

void ST7735\_uBinOut8(uint32\_t n){

if(n >= 256000){

char nErrorStr[] = "\*\*\*.\*\*"; // Error string

ST7735\_OutString(nErrorStr); // Output to LCD

}else{

uint32\_t uBinFixPtNum = (100 \* n >> 8) ; // resolution = .01

//n = (double)(n / 2.56) + .5; // 2^8 = 256 resolution; the .5 is to round a value up because the program truncates the decimal

char uBinFixPtStr[] = " . ";

for(int i = OUTPUT\_CMD\_LENGTH - 1; i >= 0; i--){

// At index less than 2, 0's should not be outputted if n is already 0

if(i < 2 && uBinFixPtNum == 0){

break;

}

// At index 3 of nBinFixedStr, there is a decimal

if(i != 3){

uBinFixPtStr[i] = uBinFixPtNum % 10 + '0';

uBinFixPtNum /= 10;

}

}

ST7735\_OutString(uBinFixPtStr); // Output to LCD

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: ST7735\_XYplotInit

// The following function creates initializes the LCD ST7735 screen to output a graph

// Input: char pointer to title, signed 32-bit min and max of x and y

// Output: None

void ST7735\_XYplotInit(char \*title, int32\_t minX, int32\_t maxX, int32\_t minY, int32\_t maxY){

ST7735\_FillScreen(0); // Reset screen to black

ST7735\_FillRect(0, 32, ST7735\_TFTWIDTH, ST7735\_TFTWIDTH, ST7735\_Color565(228,228,228)); // Space for the graph (light gray)

ST7735\_SetCursor(0, 0); // Reset cursor

ST7735\_OutString(title); // Output title onto LCD

// Set x limits

XMinLim = minX;

XMaxLim = maxX;

// Set y limits

YMinLim = minY;

YMaxLim = maxY;

// Set distance between each pixel

int32\_t xTotalDist = XMaxLim - XMinLim;

int32\_t yTotalDist = YMaxLim - YMinLim;

XScale = (double)xTotalDist / ST7735\_TFTWIDTH + .5;

YScale = (double)yTotalDist / ST7735\_TFTWIDTH + .5;

// Set point of origin of graph

// The origin is the number pixels from the left depending on minX

XOrigin = ((double)absValue(minX) / xTotalDist) \* (ST7735\_TFTWIDTH - 1);

// The origin is the number pixels from the bottom depending on minY

YOrigin = ST7735\_TFTHEIGHT - ((double)absValue(minY) / yTotalDist) \* (ST7735\_TFTWIDTH - 1);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: ST7735\_XYplot

// The following function graphs the buffers of the x and y coordinates

// Input: number of coordinates, signed 32 -bit x and y buffers

// Output: None

void ST7735\_XYplot(uint32\_t num, int32\_t bufX[], int32\_t bufY[]){

for(int i = 0; i < num; i++){

plotXYpoint(bufX[i], bufY[i]);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: plotXYpoint

// The following private function plots a (x,y) point on the graph of the LCD

// Input: signed 32-bit x and y coordinates

// Output: None

// Note: Four pixels (2 by 2) are drawn for better visualization on graph. The actual point will on the upper left corner

void plotXYpoint(int32\_t x, int32\_t y){

// If a point is beyond the scope of the graph limits, then nothing is graphed

if(!(x > XMaxLim || x < XMinLim || y > YMaxLim || y < YMinLim)){

int32\_t xOffset = (double)x / XScale + .5; // The offset from the origin to draw the pixel

int32\_t yOffset = (double)y / YScale + .5; // The .5 is meant to round the number of if above \_\_.5

// NOTE: positive y direction is up on LCD but the address decreases as y increases on graph so thats why YOrigin - yOffset

drawPixels(XOrigin + xOffset, YOrigin - yOffset, GRAPH\_POINTS\_PIXEL\_SIZE, 0);

// ST7735\_DrawPixel(XOrigin + xOffset, YOrigin - yOffset, 0); // Use this line if only 1 pixel to be drawn on LCD

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: drawPixels

// The following pixel fills in n by n pixel with the reference at the upper left corner

// Input: unsigned 32-bit x and y addresses of ST7735, size of the point (size by size), color of coordinate

// Output: None

void drawPixels(uint32\_t x, uint32\_t y, uint32\_t size, uint16\_t color){

for(int i = 0; i < size; i++){

for(int j = 0; j < size; j++){

ST7735\_DrawPixel(x + i, y + j, color);

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name: absValue

// The following private function returns the absolute value of a number

// Input: signed 32-bit number

// Output: positive value of signed 32-bit input

int32\_t absValue(int32\_t n){

if(n < 0){

n = -n;

}

return n;

}

**fixed.h**

// filename \*\*\*\*\*\*\*\* fixed.h \*\*\*\*\*\*\*\*\*\*\*\*\*\*

// possible header file for Lab 1 Spring 2018

// feel free to change the specific syntax of your system

// Sijin Woo

// 1/19/2018

#ifndef FIXED\_H

#define FIXED\_H

#include <stdint.h>

#define GRAPH\_POINTS\_PIXEL\_SIZE 2

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ST7735\_sDecOut2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

converts fixed point number to LCD

format signed 32-bit with resolution 0.001

range -9.999 to +9.999

Inputs: signed 32-bit integer part of fixed-point number

Outputs: none

send exactly 6 characters to the LCD

Parameter LCD display

12345 " \*.\*\*\*"

2345 " 2.345"

-8100 "-8.100"

-102 "-0.102"

31 " 0.031"

-12345 " \*.\*\*\*"

\*/

void ST7735\_sDecOut3(int32\_t n);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*ST7735\_uBinOut8\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned 32-bit binary fixed-point with a resolution of 1/256.

The full-scale range is from 0 to 999.99.

If the integer part is larger than 256000, it signifies an error.

The ST7735\_uBinOut6 function takes an unsigned 32-bit integer part

of the binary fixed-point number and outputs the fixed-point value on the LCD

Inputs: unsigned 32-bit integer part of binary fixed-point number

Outputs: none

send exactly 6 characters to the LCD

Parameter LCD display

0 " 0.00"

1 " 0.01"

16 " 0.25"

25 " 0.39"

125 " 1.95"

128 " 2.00"

1250 " 19.53"

7500 "117.19"

63999 "999.99"

64000 "\*\*\*.\*\*"

\*/

void ST7735\_uBinOut8(uint32\_t n);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*ST7735\_XYplotInit\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Specify the X and Y axes for an x-y scatter plot

Draw the title and clear the plot area

Inputs: title ASCII string to label the plot, null-termination

minX smallest X data value allowed, resolution= 0.001

maxX largest X data value allowed, resolution= 0.001

minY smallest Y data value allowed, resolution= 0.001

maxY largest Y data value allowed, resolution= 0.001

Outputs: none

assumes minX < maxX, and minY < maxY, and maxX - minX != 0, and maxY - minY != 0

\*/

void ST7735\_XYplotInit(char \*title, int32\_t minX, int32\_t maxX, int32\_t minY, int32\_t maxY);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*ST7735\_XYplot\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Plot an array of (x,y) data

Inputs: num number of data points in the two arrays

bufX array of 32-bit fixed-point data, resolution= 0.001

bufY array of 32-bit fixed-point data, resolution= 0.001

Outputs: none

assumes ST7735\_XYplotInit has been previously called

neglect any points outside the minX maxY minY maxY bounds

\*/

void ST7735\_XYplot(uint32\_t num, int32\_t bufX[], int32\_t bufY[]);

#endif

**Lab1.c**

// Lab1.c

// Runs on TM4C123

// Uses ST7735.c LCD.

// Jonathan Valvano

// January 17, 2018

// Possible main program to test the lab

// Feel free to edit this to match your specifications

// Backlight (pin 10) connected to +3.3 V

// MISO (pin 9) unconnected

// SCK (pin 8) connected to PA2 (SSI0Clk)

// MOSI (pin 7) connected to PA5 (SSI0Tx)

// TFT\_CS (pin 6) connected to PA3 (SSI0Fss)

// CARD\_CS (pin 5) unconnected

// Data/Command (pin 4) connected to PA6 (GPIO)

// RESET (pin 3) connected to PA7 (GPIO)

// VCC (pin 2) connected to +3.3 V

// Gnd (pin 1) connected to ground

#include <stdio.h>

#include <stdint.h>

#include "string.h"

#include "ST7735.h"

#include "PLL.h"

#include "fixed.h"

#include "../inc/tm4c123gh6pm.h"

void DelayWait10ms(uint32\_t n);

void PortF\_Init(void);

// const will place these structures in ROM

struct outTestCase1{ // used to test routines

int32\_t InNumber; // test input number

char OutBuffer[12]; // Output String

};

typedef const struct outTestCase1 outTestCaseType1;

outTestCaseType1 outTests1[13]={

{ 0, " = 0.000?\r" }, // 0/1000 = 0.000

{ 4, " = 0.004?\r" }, // 4/1000 = 0.004

{ -5, " = -0.005?\r" }, // -5/1000 = -0.005

{ 78, " = 0.078?\r" }, // 78/1000 = 0.078

{ -254, " = -0.254?\r" }, // -254/1000 = -0.254

{ 999, " = 0.999?\r" }, // 999/1000 = 0.999

{ -1000, " = -1.000?\r" }, // -1000/1000 = -1.000

{ 1234, " = 1.234?\r" }, // 1234/1000 = 1.234

{ -5678, " = -5.678?\r" }, // -5678/1000 = -5.678

{ -9999, " = -9.999?\r" }, // -9999/1000 = -9.999

{ 9999, " = 9.999?\r" }, // 9999/1000 = 9.999

{ 10000, " = \*.\*\*\*?\r" }, // positive error

{-10000, " = \*.\*\*\*?\r" } // negative error

};

// const will place these structures in ROM

struct outTestCase2{ // used to test routines

uint32\_t InNumber; // test input number

char OutBuffer[12]; // Output String

};

typedef const struct outTestCase2 outTestCaseType2;

outTestCaseType2 outTests2[14]={

{ 0, " = 0.00?\r" }, // 0/256 = 0.00

{ 2, " = 0.01?\r" }, // 2/256 = 0.01

{ 64, " = 0.25?\r" }, // 64/256 = 0.25

{ 100, " = 0.39?\r" }, // 100/256 = 0.39

{ 500, " = 1.95?\r" }, // 500/256 = 1.95

{ 512, " = 2.00?\r" }, // 512/256 = 2.00

{ 1536, " = 6.00?\r" }, // 1536/256 = 6.00

{ 5000, " = 19.53?\r" }, // 5000/256 = 19.53

{ 26000, " = 101.56?\r" }, // 26000/256 = 101.56

{ 30000, " = 117.19?\r" }, // 30000/256 = 117.19

{ 32767, " = 128.00?\r" }, // 32767/256 = 128.00

{152500, " = 595.70?\r" }, // 152500/256 = 595.70

{255997, " = 999.99?\r" }, // 255997/256 = 999.99

{256000, " = \*\*\*.\*\*?\r" }, // error

};

#define PF2 (\*((volatile uint32\_t \*)0x40025010))

#define PF3 (\*((volatile uint32\_t \*)0x40025020))

#define PF4 (\*((volatile uint32\_t \*)0x40025040))

void Pause(void){

while(PF4==0x00){

DelayWait10ms(10);

}

while(PF4==0x10){

DelayWait10ms(10);

}

}

// 180 points on a circle of radius 2.000

const int32\_t CircleXbuf[180] = { 2000, 1999, 1995, 1989, 1981, 1970, 1956, 1941, 1923, 1902, 1879, 1854, 1827, 1798, 1766, 1732, 1696, 1658, 1618, 1576, 1532, 1486, 1439, 1389, 1338, 1286, 1231, 1176, 1118, 1060, 1000, 939, 877, 813, 749, 684, 618, 551, 484, 416, 347, 278, 209, 140, 70, 0, -70, -140, -209, -278, -347, -416, -484, -551, -618, -684, -749, -813, -877, -939, -1000, -1060, -1118, -1176, -1231, -1286, -1338, -1389, -1439, -1486, -1532, -1576, -1618, -1658, -1696, -1732, -1766, -1798, -1827, -1854, -1879, -1902, -1923, -1941, -1956, -1970, -1981, -1989, -1995, -1999, -2000, -1999, -1995, -1989, -1981, -1970, -1956, -1941, -1923, -1902, -1879, -1854, -1827, -1798, -1766, -1732, -1696, -1658, -1618, -1576, -1532, -1486, -1439, -1389, -1338, -1286, -1231, -1176, -1118, -1060, -1000, -939, -877, -813, -749, -684, -618, -551, -484, -416, -347, -278, -209, -140, -70, 0, 70, 140, 209, 278, 347, 416, 484, 551, 618, 684, 749, 813, 877, 939, 1000, 1060, 1118, 1176, 1231, 1286, 1338, 1389, 1439, 1486, 1532, 1576, 1618, 1658, 1696, 1732, 1766, 1798, 1827, 1854, 1879, 1902, 1923, 1941, 1956, 1970, 1981, 1989, 1995, 1999

};

const int32\_t CircleYbuf[180] = {0, 70, 140, 209, 278, 347, 416, 484, 551, 618, 684, 749, 813, 877, 939, 1000, 1060, 1118, 1176, 1231, 1286, 1338, 1389, 1439, 1486, 1532, 1576, 1618, 1658, 1696, 1732, 1766, 1798, 1827, 1854, 1879, 1902, 1923, 1941, 1956, 1970, 1981, 1989, 1995, 1999, 2000, 1999, 1995, 1989, 1981, 1970, 1956, 1941, 1923, 1902, 1879, 1854, 1827, 1798, 1766, 1732, 1696, 1658, 1618, 1576, 1532, 1486, 1439, 1389, 1338, 1286, 1231, 1176, 1118, 1060, 1000, 939, 877, 813, 749, 684, 618, 551, 484, 416, 347, 278, 209, 140, 70, 0, -70, -140, -209, -278, -347, -416, -484, -551, -618, -684, -749, -813, -877, -939, -1000, -1060, -1118, -1176, -1231, -1286, -1338, -1389, -1439, -1486, -1532, -1576, -1618, -1658, -1696, -1732, -1766, -1798, -1827, -1854, -1879, -1902, -1923, -1941, -1956, -1970, -1981, -1989, -1995, -1999, -2000, -1999, -1995, -1989, -1981, -1970, -1956, -1941, -1923, -1902, -1879, -1854, -1827, -1798, -1766, -1732, -1696, -1658, -1618, -1576, -1532, -1486, -1439, -1389, -1338, -1286, -1231, -1176, -1118, -1060, -1000, -939, -877, -813, -749, -684, -618, -551, -484, -416, -347, -278, -209, -140, -70

};

// 50 points of a start

const int32\_t StarXbuf[50] = {0, -6, -12, -18, -24, -30, -35, -41, -47, -53, 59, 53, 47, 41, 35, 30, 24, 18, 12, 6, 95, 76, 57, 38, 19, 0, -19, -38, -57, -76, -59, -44, -28, -13, 3, 18, 33, 49, 64, 80, -95, -80, -64, -49, -33, -18, -3, 13, 28, 44

};

const int32\_t StarYbuf[50] = {190, 172, 154, 136, 118, 100, 81, 63, 45, 27, 9, 27, 45, 63, 81, 100, 118, 136, 154, 172, 121, 121, 121, 121, 121, 121, 121, 121, 121, 121, 9, 20, 31, 43, 54, 65, 76, 87, 99, 110, 121, 110, 99, 87, 76, 65, 54, 43, 31, 20

};

void SystemInit(){

}

int main(void){uint32\_t i;

PLL\_Init(Bus80MHz);

PortF\_Init();

ST7735\_InitR(INITR\_REDTAB);

//ST7735\_FillScreen(ST7735\_CYAN);

while(1){

ST7735\_FillScreen(ST7735\_BLACK);

ST7735\_SetCursor(0,0);

printf("Lab 1\rST7735\_sDecOut3\r");

for(i=0; i<13; i++){

ST7735\_sDecOut3(outTests1[i].InNumber); // your solution

ST7735\_OutString((char\*)outTests1[i].OutBuffer); // expected solution

}

Pause();

ST7735\_FillScreen(0); // set screen to black

ST7735\_SetCursor(0,0);

printf("ST7735\_uBinOut8\r");

for(i=0; i<14; i++){

ST7735\_uBinOut8(outTests2[i].InNumber); // your solution

ST7735\_OutString((char\*)outTests2[i].OutBuffer); // expected solution

}

Pause();

ST7735\_XYplotInit("Circle",-2500, 2500, -2500, 2500);

ST7735\_XYplot(180,(int32\_t \*)CircleXbuf,(int32\_t \*)CircleYbuf);

Pause();

ST7735\_XYplotInit("Star- upper right",-450, 150, -400, 200);

int bufX[5] = {0, 0, 0, 150, -450};

int bufY[5] = {0, 200, -400, 0, 0};

//ST7735\_XYplot(5, (int32\_t \*)bufX, (int32\_t \*)bufY); // Debug

ST7735\_XYplot(50,(int32\_t \*)StarXbuf,(int32\_t \*)StarYbuf);

Pause();

}

}

// PF4 is input

// Make PF2 an output, enable digital I/O, ensure alt. functions off

void PortF\_Init(void){

SYSCTL\_RCGCGPIO\_R |= 0x20; // 1) activate clock for Port F

while((SYSCTL\_PRGPIO\_R&0x20)==0){}; // allow time for clock to start

// 2) no need to unlock PF2, PF4

GPIO\_PORTF\_PCTL\_R &= ~0x000F0F00; // 3) regular GPIO

GPIO\_PORTF\_AMSEL\_R &= ~0x14; // 4) disable analog function on PF2, PF4

GPIO\_PORTF\_PUR\_R |= 0x10; // 5) pullup for PF4

GPIO\_PORTF\_DIR\_R |= 0x04; // 5) set direction to output

GPIO\_PORTF\_AFSEL\_R &= ~0x14; // 6) regular port function

GPIO\_PORTF\_DEN\_R |= 0x14; // 7) enable digital port

}

// Subroutine to wait 10 msec

// Inputs: None

// Outputs: None

// Notes: ...

void DelayWait10ms(uint32\_t n){uint32\_t volatile time;

while(n){

time = 727240\*2/91; // 10msec

while(time){

time--;

}

n--;

}

}