// Nokia5110.c

// Runs on LM4F120

// Use SSI0 to send an 8-bit code to the Nokia5110 48x84

// pixel LCD to display text, images, or other information.

// Daniel Valvano

// June 17, 2013

// Font table, initialization, and other functions based

// off of Nokia\_5110\_Example from Spark Fun:

// 7-17-2011

// Spark Fun Electronics 2011

// Nathan Seidle

// http://dlnmh9ip6v2uc.cloudfront.net/datasheets/LCD/Monochrome/Nokia\_5110\_Example.pde

/\* This example accompanies the book

"Embedded Systems: Real Time Interfacing to Arm Cortex M Microcontrollers",

ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2013

Copyright 2013 by Jonathan W. Valvano, valvano@mail.utexas.edu

You may use, edit, run or distribute this file

as long as the above copyright notice remains

THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED

OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF

MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.

VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,

OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

For more information about my classes, my research, and my books, see

http://users.ece.utexas.edu/~valvano/

\*/

// Signal (Nokia 5110) LM4F120 pin

// Reset (RST, pin 1) connected to PA7

// SSI0Fss (CE, pin 2) connected to PA3

// Data/Command (DC, pin 3) connected to PA6

// SSI0Tx (Din, pin 4) connected to PA5

// SSI0Clk (Clk, pin 5) connected to PA2

// 3.3V (Vcc, pin 6) power

// not connected (BL, pin 7) back light

// Ground (Gnd, pin 8)

#include "Nokia5110.h"

#define DC (\*((volatile unsigned long \*)0x40004100))

#define DC\_COMMAND 0

#define DC\_DATA 0x40

#define RESET (\*((volatile unsigned long \*)0x40004200))

#define RESET\_LOW 0

#define RESET\_HIGH 0x80

#define GPIO\_PORTA\_DIR\_R (\*((volatile unsigned long \*)0x40004400))

#define GPIO\_PORTA\_AFSEL\_R (\*((volatile unsigned long \*)0x40004420))

#define GPIO\_PORTA\_DEN\_R (\*((volatile unsigned long \*)0x4000451C))

#define GPIO\_PORTA\_AMSEL\_R (\*((volatile unsigned long \*)0x40004528))

#define GPIO\_PORTA\_PCTL\_R (\*((volatile unsigned long \*)0x4000452C))

#define SSI0\_CR0\_R (\*((volatile unsigned long \*)0x40008000))

#define SSI0\_CR1\_R (\*((volatile unsigned long \*)0x40008004))

#define SSI0\_DR\_R (\*((volatile unsigned long \*)0x40008008))

#define SSI0\_SR\_R (\*((volatile unsigned long \*)0x4000800C))

#define SSI0\_CPSR\_R (\*((volatile unsigned long \*)0x40008010))

#define SSI0\_CC\_R (\*((volatile unsigned long \*)0x40008FC8))

#define SSI\_CR0\_SCR\_M 0x0000FF00 // SSI Serial Clock Rate

#define SSI\_CR0\_SPH 0x00000080 // SSI Serial Clock Phase

#define SSI\_CR0\_SPO 0x00000040 // SSI Serial Clock Polarity

#define SSI\_CR0\_FRF\_M 0x00000030 // SSI Frame Format Select

#define SSI\_CR0\_FRF\_MOTO 0x00000000 // Freescale SPI Frame Format

#define SSI\_CR0\_DSS\_M 0x0000000F // SSI Data Size Select

#define SSI\_CR0\_DSS\_8 0x00000007 // 8-bit data

#define SSI\_CR1\_MS 0x00000004 // SSI Master/Slave Select

#define SSI\_CR1\_SSE 0x00000002 // SSI Synchronous Serial Port

// Enable

#define SSI\_SR\_RNE 0x00000004 // SSI Receive FIFO Not Empty

#define SSI\_SR\_TNF 0x00000002 // SSI Transmit FIFO Not Full

#define SSI\_SR\_TFE 0x00000001 // SSI Transmit FIFO Empty

#define SSI\_CPSR\_CPSDVSR\_M 0x000000FF // SSI Clock Prescale Divisor

#define SSI\_CC\_CS\_M 0x0000000F // SSI Baud Clock Source

#define SSI\_CC\_CS\_SYSPLL 0x00000000 // Either the system clock (if the

// PLL bypass is in effect) or the

// PLL output (default)

#define SYSCTL\_RCGC1\_R (\*((volatile unsigned long \*)0x400FE104))

#define SYSCTL\_RCGC2\_R (\*((volatile unsigned long \*)0x400FE108))

#define SYSCTL\_RCGC1\_SSI0 0x00000010 // SSI0 Clock Gating Control

#define SYSCTL\_RCGC2\_GPIOA 0x00000001 // port A Clock Gating Control

enum typeOfWrite{

COMMAND, // the transmission is an LCD command

DATA // the transmission is data

};

// This is a helper function that sends an 8-bit message to the LCD.

// inputs: type COMMAND or DATA

// data 8-bit data to transmit

// outputs: none

// assumes: SSI0 and port A have already been initialized

void static lcdwrite(enum typeOfWrite type, char data){

if(type == COMMAND){

DC = DC\_COMMAND;

} else{

DC = DC\_DATA;

}

while((SSI0\_SR\_R&SSI\_SR\_TFE)==0){}; // wait until transmit FIFO empty

SSI0\_DR\_R = data; // data out

while((SSI0\_SR\_R&SSI\_SR\_RNE)==0){}; // wait until response

data = SSI0\_DR\_R; // remove meaningless response from receive FIFO

}

//\*\*\*\*\*\*\*\*Nokia5110\_Init\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Initialize Nokia 5110 48x84 LCD by sending the proper

// commands to the PCD8544 driver. One new feature of the

// LM4F120 is that its SSIs can get their baud clock from

// either the system clock or from the 16 MHz precision

// internal oscillator. If the system clock is faster than

// 50 MHz, the SSI baud clock will be faster than the 4 MHz

// maximum of the Nokia 5110.

// inputs: none

// outputs: none

// assumes: system clock rate of 50 MHz or less

void Nokia5110\_Init(void){

volatile unsigned long delay;

SYSCTL\_RCGC1\_R |= SYSCTL\_RCGC1\_SSI0; // activate SSI0

SYSCTL\_RCGC2\_R |= SYSCTL\_RCGC2\_GPIOA; // activate port A

delay = SYSCTL\_RCGC2\_R; // allow time to finish activating

GPIO\_PORTA\_DIR\_R |= 0xC0; // make PA6,7 out

GPIO\_PORTA\_AFSEL\_R |= 0x2C; // enable alt funct on PA2,3,5

GPIO\_PORTA\_AFSEL\_R &= ~0xC0; // disable alt funct on PA6,7

GPIO\_PORTA\_DEN\_R |= 0xEC; // enable digital I/O on PA2,3,5,6,7

// configure PA2,3,5 as SSI

GPIO\_PORTA\_PCTL\_R = (GPIO\_PORTA\_PCTL\_R&0xFF0F00FF)+0x00202200;

// configure PA6,7 as GPIO

GPIO\_PORTA\_PCTL\_R = (GPIO\_PORTA\_PCTL\_R&0x00FFFFFF)+0x00000000;

GPIO\_PORTA\_AMSEL\_R &= ~0xEC; // disable analog functionality on PA2,3,5,6,7

SSI0\_CR1\_R &= ~SSI\_CR1\_SSE; // disable SSI

SSI0\_CR1\_R &= ~SSI\_CR1\_MS; // master mode

// configure for system clock/PLL baud clock source

SSI0\_CC\_R = (SSI0\_CC\_R&~SSI\_CC\_CS\_M)+SSI\_CC\_CS\_SYSPLL;

// clock divider for 3.125 MHz SSIClk (50 MHz PIOSC/16)

SSI0\_CPSR\_R = (SSI0\_CPSR\_R&~SSI\_CPSR\_CPSDVSR\_M)+16;

SSI0\_CR0\_R &= ~(SSI\_CR0\_SCR\_M | // SCR = 0 (3.125 Mbps data rate)

SSI\_CR0\_SPH | // SPH = 0

SSI\_CR0\_SPO); // SPO = 0

// FRF = Freescale format

SSI0\_CR0\_R = (SSI0\_CR0\_R&~SSI\_CR0\_FRF\_M)+SSI\_CR0\_FRF\_MOTO;

// DSS = 8-bit data

SSI0\_CR0\_R = (SSI0\_CR0\_R&~SSI\_CR0\_DSS\_M)+SSI\_CR0\_DSS\_8;

SSI0\_CR1\_R |= SSI\_CR1\_SSE; // enable SSI

RESET = RESET\_LOW; // reset the LCD to a known state

for(delay=0; delay<10; delay=delay+1);// delay minimum 100 ns

RESET = RESET\_HIGH; // negative logic

lcdwrite(COMMAND, 0x21); // chip active; horizontal addressing mode (V = 0); use extended instruction set (H = 1)

lcdwrite(COMMAND, 0xB8); // set LCD Vop (contrast): try 0xB1 (good @ 3.3V) or 0xBF if your display is too dark

lcdwrite(COMMAND, 0x04); // set temp coefficient

lcdwrite(COMMAND, 0x14); // LCD bias mode 1:48: try 0x13 or 0x14

lcdwrite(COMMAND, 0x20); // we must send 0x20 before modifying the display control mode

lcdwrite(COMMAND, 0x0C); // set display control to normal mode: 0x0D for inverse

}

//\*\*\*\*\*\*\*\*Nokia5110\_OutChar\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Print a character to the Nokia 5110 48x84 LCD. The

// character will be printed at the current cursor position,

// the cursor will automatically be updated, and it will

// wrap to the next row or back to the top if necessary.

// One blank column of pixels will be printed on either side

// of the character for readability. Since characters are 8

// pixels tall and 5 pixels wide, 12 characters fit per row,

// and there are six rows.

// inputs: data character to print

// outputs: none

// assumes: LCD is in default horizontal addressing mode (V = 0)

void Nokia5110\_OutChar(unsigned char data){

int i;

lcdwrite(DATA, 0x00); // blank vertical line padding

for(i=0; i<5; i=i+1){

lcdwrite(DATA, ASCII[data - 0x20][i]);

}

lcdwrite(DATA, 0x00); // blank vertical line padding

}

//\*\*\*\*\*\*\*\*Nokia5110\_OutString\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Print a string of characters to the Nokia 5110 48x84 LCD.

// The string will automatically wrap, so padding spaces may

// be needed to make the output look optimal.

// inputs: ptr pointer to NULL-terminated ASCII string

// outputs: none

// assumes: LCD is in default horizontal addressing mode (V = 0)

void Nokia5110\_OutString(char \*ptr){

while(\*ptr){

Nokia5110\_OutChar((unsigned char)\*ptr);

ptr = ptr + 1;

}

}

//\*\*\*\*\*\*\*\*Nokia5110\_OutUDec\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Output a 16-bit number in unsigned decimal format with a

// fixed size of five right-justified digits of output.

// Inputs: n 16-bit unsigned number

// Outputs: none

// assumes: LCD is in default horizontal addressing mode (V = 0)

void Nokia5110\_OutUDec(unsigned short n){

if(n < 10){

Nokia5110\_OutString(" ");

Nokia5110\_OutChar(n+'0'); /\* n is between 0 and 9 \*/

} else if(n<100){

Nokia5110\_OutString(" ");

Nokia5110\_OutChar(n/10+'0'); /\* tens digit \*/

Nokia5110\_OutChar(n%10+'0'); /\* ones digit \*/

} else if(n<1000){

Nokia5110\_OutString(" ");

Nokia5110\_OutChar(n/100+'0'); /\* hundreds digit \*/

n = n%100;

Nokia5110\_OutChar(n/10+'0'); /\* tens digit \*/

Nokia5110\_OutChar(n%10+'0'); /\* ones digit \*/

}

else if(n<10000){

Nokia5110\_OutChar(' ');

Nokia5110\_OutChar(n/1000+'0'); /\* thousands digit \*/

n = n%1000;

Nokia5110\_OutChar(n/100+'0'); /\* hundreds digit \*/

n = n%100;

Nokia5110\_OutChar(n/10+'0'); /\* tens digit \*/

Nokia5110\_OutChar(n%10+'0'); /\* ones digit \*/

}

else {

Nokia5110\_OutChar(n/10000+'0'); /\* ten-thousands digit \*/

n = n%10000;

Nokia5110\_OutChar(n/1000+'0'); /\* thousands digit \*/

n = n%1000;

Nokia5110\_OutChar(n/100+'0'); /\* hundreds digit \*/

n = n%100;

Nokia5110\_OutChar(n/10+'0'); /\* tens digit \*/

Nokia5110\_OutChar(n%10+'0'); /\* ones digit \*/

}

}

//\*\*\*\*\*\*\*\*Nokia5110\_SetCursor\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Move the cursor to the desired X- and Y-position. The

// next character will be printed here. X=0 is the leftmost

// column. Y=0 is the top row.

// inputs: newX new X-position of the cursor (0<=newX<=11)

// newY new Y-position of the cursor (0<=newY<=5)

// outputs: none

void Nokia5110\_SetCursor(unsigned char newX, unsigned char newY){

if((newX > 11) || (newY > 5)){ // bad input

return; // do nothing

}

// multiply newX by 7 because each character is 7 columns wide

lcdwrite(COMMAND, 0x80|(newX\*7)); // setting bit 7 updates X-position

lcdwrite(COMMAND, 0x40|newY); // setting bit 6 updates Y-position

}

//\*\*\*\*\*\*\*\*Nokia5110\_Clear\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Clear the LCD by writing zeros to the entire screen and

// reset the cursor to (0,0) (top left corner of screen).

// inputs: none

// outputs: none

void Nokia5110\_Clear(void){

int i;

for(i=0; i<(MAX\_X\*MAX\_Y/8); i=i+1){

lcdwrite(DATA, 0x00);

}

Nokia5110\_SetCursor(0, 0);

}

//\*\*\*\*\*\*\*\*Nokia5110\_DrawFullImage\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Fill the whole screen by drawing a 48x84 bitmap image.

// inputs: ptr pointer to 504 byte bitmap

// outputs: none

// assumes: LCD is in default horizontal addressing mode (V = 0)

void Nokia5110\_DrawFullImage(const char \*ptr){

int i;

Nokia5110\_SetCursor(0, 0);

for(i=0; i<(MAX\_X\*MAX\_Y/8); i=i+1){

lcdwrite(DATA, ptr[i]);

}

}