```c
/*
	CE2812 Lab 3
	LCD API
	Evan Heinrich
	12/16/2021
	Testing method
*/



#include <stdio.h>
#include <stdlib.h>
#include "uart_driver.h"
#include "LCD.h"
#include "delay.h"

#define F_CPU 16000000UL

int main(void){
	init_usart2(57600,F_CPU);

	// Initializing the LCD also initializes the delay API
	LCD_Init();

	char test[12] = "Testing\0";
	int testnum = 12345;

	while(1==1) {
		printf("Testing PrintString, expecting 7\n");
		uint32_t reported = LCD_PrintString(test);
		printf("Function reported %d\n", (int)reported);
		delay_ms(2000);

		printf("Testing LCD_MoveCursor by a newline...\n");
		LCD_MoveCursor(1, 0);
		delay_ms(2000);

		printf("Testing PrintNum, expecting 5\n");
		reported = LCD_PrintNum(testnum);
		printf("Function reported %d\n", (int)reported);
		delay_ms(2000);

		printf("Testing Homing...");
		LCD_Home();
		delay_ms(2000);

		printf("\nClearing display in 3 seconds");
		delay_ms(3000);
		LCD_Clear();
		delay_ms(2000);
	}

	exit(EXIT_SUCCESS);

	return 0;
}
```

```c
/*

	CE2812
	LCD API, the actual API
	Evan Heinrich
	12/16/2021
*/

#include "registers.h"
#include "delay.h"
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#define DECIMAL 10

/*
 * Private function, writes an instruction
 * to the display. Mainly used when initializing
 * the display or moving the cursor
 */
static void LCD_WriteInstruction(uint32_t inst) {
	volatile uint32_t* addr;

	// Clear RS&RW flags, set E
	addr = GPIOC_BSRR;
	*addr = (7<<24)|(1<<10);

	// LCD takes 8-bit instructions, so clear all
	// but the lower 8 bits
	inst &= 0xFF;

	// Data bus starts on PA4 so shift the instruction left 4
	inst = inst << 4;

	// Clear data bus
	addr = GPIOA_BSRR;
	*addr = 0x0FF00000;

	// Write
	*addr = inst;

	// Clear E flag which executes instruction
	addr = GPIOC_BSRR;
	*addr = 1<<26;

	delay_us(1);

	return;
}
```

```c
/*
 * Private function, writes data to the LCD memory.
 * Mainly used when writing the chars from a string
 */
static void LCD_WriteData(uint32_t data) {
	volatile uint32_t* addr;

	// Clear RW, set RS and E
	addr = GPIOC_BSRR;
	*addr = (1<<25)|(5<<8);


	// LCD takes 8-bit data, so clear all
	// but the lower 8 bits
	data &= 0xFF;

	// Data bus starts on PA4 so shift the instruction left 4
	data = data << 4;

	// Write to data bus
	addr = GPIOA_BSRR;
	*addr = 0x0FF00000;
	*addr = data;

	// Clear E flag
	addr = GPIOC_BSRR;
	*addr = 1<<26;

	return;
}

/*
 * Initializes the LCD Display on our MSOE Devboards
 * Also initializes the delay API as delays are needed
 * between instructions
 */
void LCD_Init() {
	volatile uint32_t* addr;

	// Enable GPIOA and GPIOC in RCC
	addr = RCC_AHB1ENR;
	*addr |= (1<<0)|(1<<2);

	// Set PA4-PA11 as outputs
	addr = GPIOA_MODER;
	*addr |= 0x00555500;

	// Set PC8-PC10 as outputs
	addr = GPIOC_MODER;
	*addr |= 0x00550000;

	// Make sure to initialize delay as we need them
	delay_Init();

	// LCD Initialization sequence

	// Function set 2x, 40us delay each
	LCD_WriteInstruction(0x38);
	delay_us(40);
```

```
	LCD_WriteInstruction(0x38);
	delay_us(40);

	// Display on, 40us delay
	LCD_WriteInstruction(0x0F);
	delay_us(40);

	// Display clear, 2ms delay
	LCD_WriteInstruction(0x01);
	delay_ms(2);

	// Entry mode, 40us delay
	LCD_WriteInstruction(0x06);
	delay_us(40);

	return;
}

/*
 * Clears the LCD Display on our MSOE Devboards
 */
void LCD_Clear() {
	// Write clear display instruction
	LCD_WriteInstruction(0x01);
	delay_ms(2);

	return;
}

/*
 * Returns the cursor to the home location
 * on our MSOE Devboards
 */
void LCD_Home() {
	LCD_WriteInstruction(0x02);
	delay_ms(2);
	return;
}

/*
 * Moves the LCD cursor to a zero-indexed row and
 * column on our MSOE Devboards
 */
void LCD_MoveCursor(uint32_t row, uint32_t col) {
	uint32_t DDRAM = 0;
	// Our displays can do up to 40 chars/line
	// so if desired column is >39, don't do anything
	// Likewise with rows, but we have a two row display
	if(!(row > 0x1 || col > 0x27)) {
		if(row == 0) {
				DDRAM = col;
			} else {
				DDRAM = 0x40 + col;
			}
		LCD_WriteInstruction(DDRAM|(1<<7));
	}

	return;
}
```

```c
/*
 * Prints a null-terminated string on our
 * MSOE devboards
 * Returns the number of characters printed
 */
uint32_t LCD_PrintString(char* string) {
	uint32_t reported = strlen(string);
	uint32_t actual = 0;

	for(int i = 0; i < reported; i++) {
		if(string[i] != '\0') {
			LCD_WriteData((uint32_t)(string[i]));
			delay_us(40);
			actual++;
		} else {
			break;
		}
	}

	return actual;
}

/*
 * Prints a base-10 signed integer on our MOSE
 * Devboards
 * Returns the number of digits printed
 */
uint32_t LCD_PrintNum(signed int num) {
	char* str[12];
	sprintf((char*)str, "%d", num);

	return LCD_PrintString(str);
}
```

```c
/*
	CE2812
	Header for LCD API
	Evan Heinrich
	12/16/2021
*/

#ifndef LCD_IS_ALIVE
#define LCD_IS_ALIVE 1

#include <stdint.h>

void LCD_Init();
void LCD_Clear();
void LCD_Home();
void LCD_MoveCursor(uint32_t row, uint32_t col);
uint32_t LCD_PrintString(char* string);
uint32_t LCD_PrintNum(signed int num);

#endif
```

```c
/*
	CE2812
	Register addresses for our NODE F446RE boards
	Evan Heinrich
	NOTE: Running list of all registers used, updated as new
	peripherals are used
*/

#ifndef REG_LIST_ALIVE
#define REG_LIST_ALIVE 1
#include <stdint.h>

#define RCC_APB1ENR (volatile uint32_t*) 0x40023840
#define RCC_AHB1ENR (volatile uint32_t*) 0x40023830
#define TIM2_CR1    (volatile uint32_t*) 0x40000000
#define TIM2_PSC    (volatile uint32_t*) 0x40000028
#define TIM2_CNT    (volatile uint32_t*) 0x40000024
#define TIM2_EGR    (volatile uint32_t*) 0x40000014
#define TIM2_SR     (volatile uint32_t*) 0x40000010
#define TIM2_ARR    (volatile uint32_t*) 0x4000002C
#define GPIOB_MODER (volatile uint32_t*) 0x40020400
#define GPIOB_ODR   (volatile uint32_t*) 0x40020414
#define GPIOB_BSRR  (volatile uint32_t*) 0x40020418
#define GPIOA_MODER (volatile uint32_t*) 0x40020000
#define GPIOA_ODR   (volatile uint32_t*) 0x40020014
#define GPIOA_BSRR  (volatile uint32_t*) 0x40020018
#define GPIOC_MODER (volatile uint32_t*) 0x40020800
#define GPIOC_ODR   (volatile uint32_t*) 0x40020814
#define GPIOC_BSRR  (volatile uint32_t*) 0x40020818

#endif
```