RTC

Source:https://docs.google.com/document/d/1Fz_3EFQYDWudJpF82D8prcQUcPtW_7voxKk2pW1IyA8/edit?tab=t.0#heading=h.othge5f6t2iz

Library Overview

The LCD Library provides a simple I2C character LCD driver for the RFID Security Access System. It performs a hardware reset, sets up controller options, and exposes a small API for clearing the display, positioning the cursor, and printing characters or strings. The driver uses a control byte protocol (0x00 for commands, 0x40 for data) and assumes a DDRAM layout with 0x20 bytes per row.

Key Features

- **I2C Text Display:** Command/data writes over I2C (7-bit address 0x3C)
- **Hardware Reset:** Dedicated reset line on RB6 (active-low)
- **Cursor Control:** Positioning via DDRAM address calculation
- **Simple API:** Clear, set cursor, print char, print string
- **Timing Safe:** Built-in initialization and clear delays

Use Cases

- Displaying prompts for PIN entry and RFID status
- Showing current time/date and access results
- General UI messaging and debugging output

Function Summary

Function Name	Description

lcd_init(void)	Initializes the LCD controller and clears the display. Performs a hardware reset and sends configuration commands.
	Input: none
	Return: none
	Usage:
	int main(void) {
	lcd_init(); // Call once during system initialization
	
	while(1) {
	// Main program loop
	}
	}

lcd_setCursor(char x, char y)	Sets the cursor position on the LCD Input: x - column index (0-based) y - row index (0-based) Return: none Usage: lcd_setCursor(0, 0); // Set cursor to top left.
lcd_printChar(char c)	Prints a single character at the current cursor position Input: c - ASCII character to write Return: none Usage: lcd_printChar('H');
lcd_printStr(const char *s)	Prints a null-terminated string starting at current cursor Input: s - string to print. Return: none. Usage: lcd_printStr("Hello, EE!");
lcd_clear(void)	Clears the LCD display Input: none Return: none Usage: lcd_clear();

Wiring Diagram

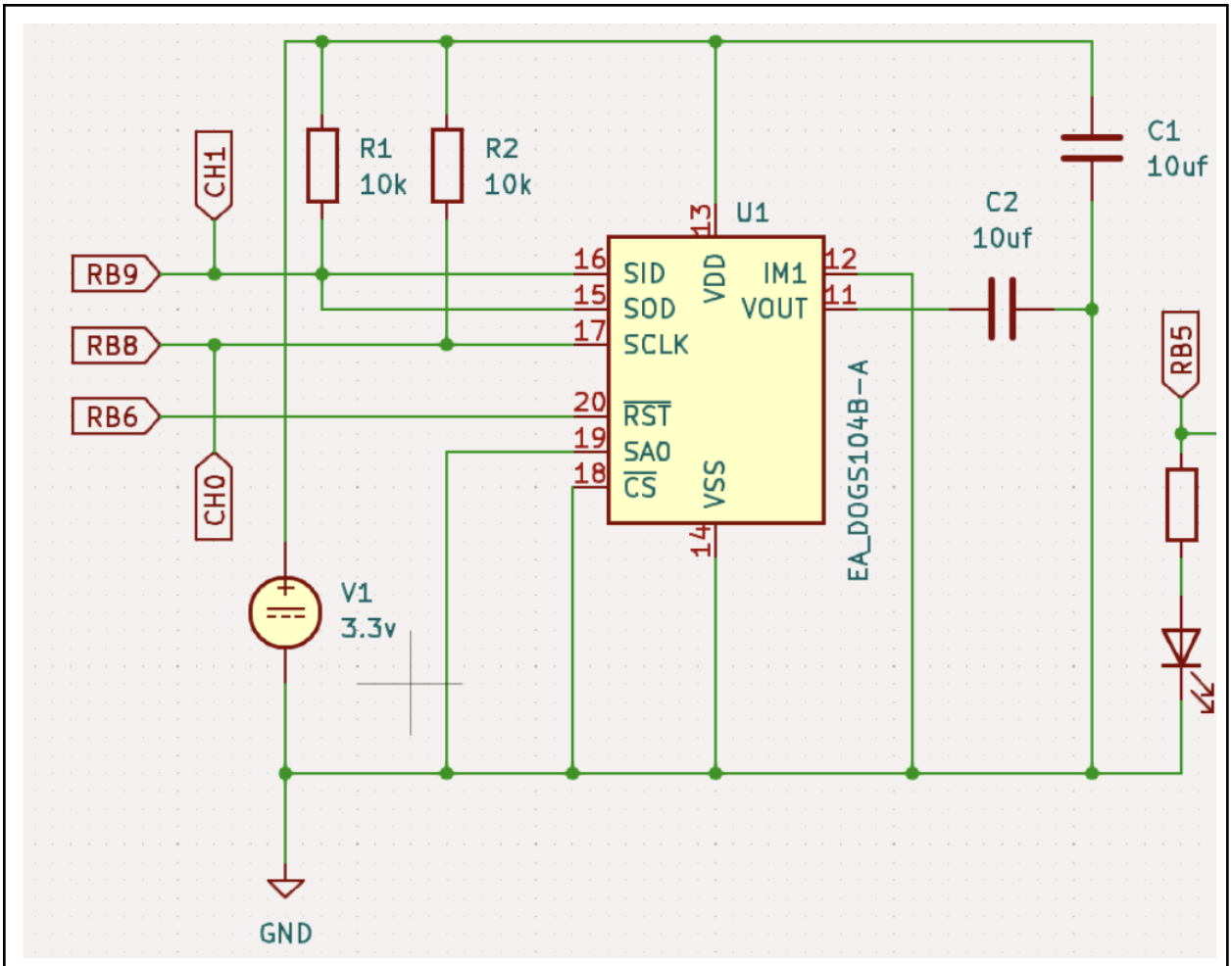

Figure 1: Configuration for the LCD diagram

Notes: Pullup resistors are shared with the RTC chip and connected to the SDA/SCL lines and VCC, and the SDA/SCL lines themselves are connected to pin 18(RB9) and 17(RB8) respectively. Finally, VCC and GND are connected to the PIC24's appropriate pins.

Code

Source file

```
/*
 * File: lcd.c
 * Author: Yusuf Mahamud
 * Project: RFID Security Access System
 * Description: Implementation of the LCD driver over I2C
```

```
 */

#include <string.h>                // Include string library for string ops
#define FCY 16000000UL             // Define instruction cycle frequency
#include <libpic30.h>              // Include for __delay_ms() function
#include "pic24.h"                 // Include for pic24_write_i2c() function
#include "xc.h"
#include "lcd.h"

#define LCD_ADDR_I2C 0x3C          // 7-bit I2C address of the LCD module
#define LCD_RST LATBbits.LATB6     // LCD reset pin (active-low) on RB6

// Function: lcd_cmd
// Description: Sends a single command byte to the LCD over I2C
// Input: cmd - instruction byte per LCD controller datasheet
// Return: none
static void lcd_cmd(uint8_t cmd)
{
    uint8_t buf[2] = {0x00, cmd};    // Control=0x00 (Co=0, RS=0), then command
    pic24_write_i2c(LCD_ADDR_I2C, buf, 2);
}

// Function: lcd_init
// Description: Initializes the LCD controller and clears the display
// Performs a hardware reset and sends configuration commands
// Return: none
void lcd_init(void)
{
    // Reset the LCD display (active-low pulse sequence on RB6)
    __delay_ms(2);
    LCD_RST = 1;                   // set reset line HIGH (inactive)
    __delay_ms(2);
    LCD_RST = 0;                   // pull reset line LOW (assert reset)
    __delay_ms(2);
    LCD_RST = 1;                   // release reset line HIGH
    __delay_ms(40);                // wait for controller to stabilize

    // Extended function set / bias / entry mode / power settings
    lcd_cmd(0x3A);                 // Function set (extended)
    lcd_cmd(0x09);                 // Bias/oscillator setting (module specific)
    lcd_cmd(0x06);                 // Entry mode set (cursor move, shift off)
    lcd_cmd(0x1E);                 // Internal power/booster settings

    __delay_ms(40);                // allow settings to take effect

    // Extended function set again + contrast settings (3-part sequence)
```

```
    lcd_cmd(0x39);              // Function set (extended)
    lcd_cmd(0x1B);              // Internal follower control
    lcd_cmd(0x6E);              // Contrast set (part 1)
    lcd_cmd(0x56);              // Contrast set (part 2)
    lcd_cmd(0x7A);              // Contrast set (part 3)

    __delay_ms(40);             // wait for power/contrast stabilization

    // Return to normal instruction set; enable display with cursor + blink
    lcd_cmd(0x38);              // Function set (normal instruction set)
    lcd_cmd(0x0F);              // Display ON, cursor ON, blink ON

    __delay_ms(40);             // allow display to turn on

    // Final configuration tweaks and clear display
    lcd_cmd(0x3A);              // Extended function set
    lcd_cmd(0x09);              // Bias setting
    lcd_cmd(0x1A);              // Power control tweak
    lcd_cmd(0x3C);              // Additional function set
    lcd_cmd(0x01);              // Clear display

    __delay_ms(5);              // clear command requires >1.5ms
}

// Function: lcd_clear
// Description: Clears the display and homes the cursor
// Return: none
void lcd_clear()
{
    lcd_cmd(0x01);
    __delay_ms(5);
}

// Function: lcd_setCursor
// Description: Sets the cursor position on the LCD
// Input: x - column index (0-based)
//        y - row index (0-based)
// Notes: DDRAM address spacing uses 0x20 per line for this controller
// Return: none
void lcd_setCursor(char x, char y)
{
    uint8_t addr = (uint8_t)(0x20 * y + x);
    uint8_t buf[2] = {0x00, (uint8_t)(0x80 | addr)};
    pic24_write_i2c(LCD_ADDR_I2C, buf, 2);
}
```

```
// Function: lcd_printChar
// Description: Prints a single character at the current cursor position
// Input: c - ASCII character to write
// Return: none
void lcd_printChar(char c)
{
    uint8_t buf[2] = {0x40, (uint8_t)c};
    pic24_write_i2c(LCD_ADDR_I2C, buf, 2);
}

// Function: lcd_printStr
// Description: Prints a null-terminated string starting at current cursor
// Input: s - pointer to C string to print
// Notes: No automatic line wrapping is performed
// Return: none
void lcd_printStr(const char *s)
{
    while (*s != '\0')
    {
        lcd_printChar(*s++);
    }
}
```

Header file

```
/*
 * File: lcd.h
 * Author: Yusuf Mahamud
 * Project: RFID Security Access System
 * Description: Header file for the LCD display module
 */

#ifndef LCD_H
#define LCD_H

void lcd_init(void);
void lcd_setCursor(char x, char y);
void lcd_printChar(char c);
void lcd_printStr(const char *s);
void lcd_clear(void);

#endif /* LCD_H */
```

```
void rtc_init(void);
void rtc_set_time(rtc_time_t *time);
void rtc_get_time(rtc_time_t *time);
float rtc_get_temperature(void);

#endif /* RTC_H */
```