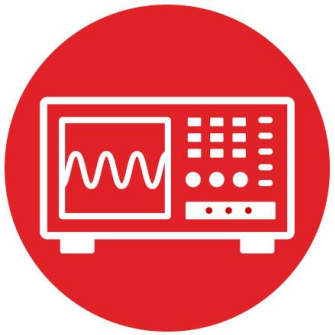


Module 11

Lab 11: Liquid Crystal Display

Lab: Liquid Crystal Display

11.0 Objectives

The purpose of this lab is to interface an LCD display to the microcontroller and develop a set of software routines to assist in debugging subsequent labs on the robot.

1. You will connect an LCD to the microcontroller.
2. You will use the synchronous serial protocol to communicate.
3. You will implement a set of software functions for the LCD.

Good to Know: Even though the LCD doesn't actually contribute to the input-calculate-output loop required to solve the robot challenge, it will be extremely useful when debugging when the robot is running untethered in the arena.

11.1 Getting Started

11.1.1 Software Starter Projects

Look at this project:
Lab11_FSM (starter project for this lab)

11.1.2 Student Resources (in datasheets directory)

Nokia5110.pdf (data sheet for the LCD)

11.1.3 Reading Materials

Volume 1 Sections 4.5, 7.6, 7.7, 8.3, and 8.4
Embedded Systems: Introduction to the MSP432 Microcontroller",
or
Volume 2 Sections 1.5, 3.4, and 7.5
Embedded Systems: Real-Time Interfacing to the MSP432 Microcontroller",

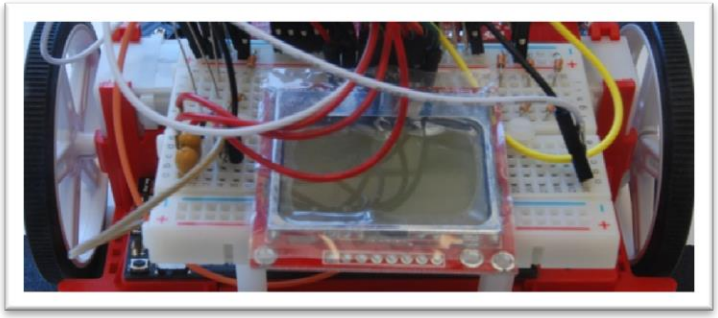


Figure 1. Bump sensors positioned at the front of the robot.

11.1.4 Components needed for this lab

Quantity	Description	Manufacturer	Mfg P/N
1	MSP-EXP432P401R LaunchPad	TI	MSP-EXP432P401R
1	LCD display	Nokia	5110

11.1.5 Lab equipment needed

Oscilloscope (two channels with at least 10MHz sampling)
Or Logic Analyzer (4 channels with at least 10MHz sampling)

11.2 System Design Requirements

The overall goal of this lab is to interface an LCD to the microcontroller and use it to output characters and numbers. The Nokia 5110 is a monochrome display that is 84 pixels wide by 48 pixels high. Each character is defined by a 5 pixels wide by 8 pixels high image. You can see the font table as a constant array called **ASCII** inside **Nokia.c** starter file. For example, the letter '7' is defined in line 132 as

```
{0x01, 0x71, 0x09, 0x05, 0x03}
```

This 40-bit value creates the image shown in Figure 1 on the display. The 0x01 is the first column and 0x03 is the last column, with bit 0 on top.

Each character has bit 7 clear to make a space between lines. The function **Nokia5110_OutChar** will place a blank vertical line in front of and one blank line after each character. This means each character requires a 7 by 8 pixel area to print. Since the display is 84 wide by 48 high, this font size allows for 84/7= 12 characters on each line, and allows 48/8=6 lines.

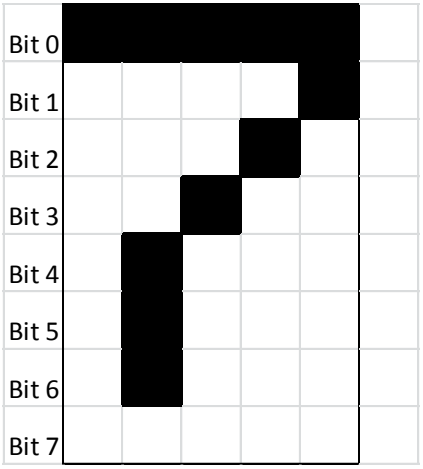


Figure 1. Pixel image to create the number 7, showing a blank vertical line before and after the character. Bit 7 is clear for all characters.

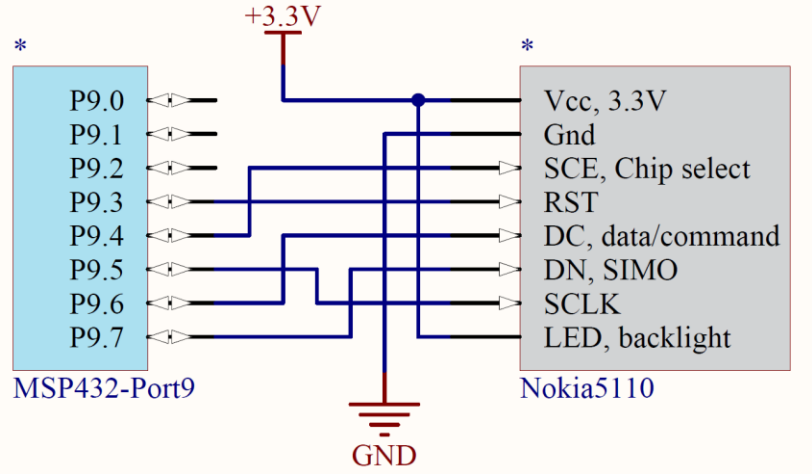
On the low level, you are required to write one routine that sends an 8-bit command to the LCD (**lcdcommandwrite**) and a second routine that sends 8-bit data to the LCD (**lcddatawrite**). These routines use busy-wait synchronization with the SPI synchronous serial interface. These are private functions, so prototypes are not available in the **Nokia5110.h** header file.

On the high level, you are required to write one routine that outputs a string (**Nokia5110_OutString**) and a second routine to output an unsigned 16-bit decimal number (**Nokia5110_OutUDec**).

You will find the **Nokia5110.h** and **Nokia5110.c** files in the **inc** folder. This means after you complete this lab, you can use these functions in the remaining labs.

11.3 Experiment set-up

You will implement this lab using the MSP432 LaunchPad and the LCD. Figure 2 shows Port 9 can be used to interface the LCD. However you may use any pins that support the synchronous serial port (SPI protocol). Pins P9.7, P9.5, and P9.4 are configured for SPI mode. Pins P9.3 and P9.6 are set to be regular digital outputs. DC=1 means data, and DC=0 means command. The reset pin is used to initialize the LCD hardware (RESET=0, at least 100ns wait, RESET=1).



11.4 System Development Plan

11.4.1 lcdcommandwrite

To send one 8-bit command, your **lcdcommandwrite** function should perform the following four steps, even though 1) and 4) are the same.

1) Wait for the SPI to be idle (let previous frame finish), **UCBUSY**
2) Set DC for command (0)
3) Write 8-bit command to the SPI data register (**TXBUF**), starts SPI
4) Wait for the SPI to be idle (let this transmission finish), **UCBUSY**

Look up in the Nokia5110 and MSP432 data sheets what the expected waveforms should look like when **lcdcommandwrite** is called in the program. Use an oscilloscope or logic analyzer to verify the waveforms are as expected. You can use a simple program like the following to test this low-level function.

```
void Testlcdcommandwrite(void){
  while(1){
    lcdcommandwrite(0x21);
  }
}
```

11.4.2 lcddatawrite

To send one 8-bit data, your **lcddatawrite** function should perform the following three steps. Skipping step 4) when outputting data makes it run much faster.

1) Wait for the SPI to be idle (let previous frame finish), **UCBUSY**
2) Set DC for data (1)
3) Write 8-bit data to the SPI data register (**TXBUF**), starts SPI

11.4.3 Nokia5110_OutString

There is a midlevel function given to you called **Nokia5110_OutChar** that outputs one character to the LCD. You will use this function to implement a function called **Nokia5110_OutString** that outputs a string to the LCD.

11.4.4 Nokia5110_OutUDec

Similarly, you will use **Nokia5110_OutChar** to implement a function called **Nokia5110_OutUDec** that outputs a 16-bit unsigned number to the LCD. One of the specifications for this function is that the image is created right justified that fills exactly 5 characters. This functionality allows you to output numbers on the LCD that are easy to read.

More specifically, for numbers 0 to 9, you will output 4 spaces and the one digit. For numbers 10 to 99, you will output 3 spaces followed by two digits. For numbers 100 to 999, you will output 2 spaces followed by three digits. For numbers 1000 to 9999, you will output 1 space followed by four digits. For numbers 10000 to 65535, you will output all five digits.

To illustrate how this function could be used, consider the example where the LCD contains debug data continuously updated during a robot run, You could execute this code once at the beginning

```
Nokia5110_SetCursor(0,2);  // left, third row
Nokia5110_OutString("D= ");
Nokia5110_OutUDec(0);
Nokia5110_OutString(" mm");
```

Then, when you wish to update the LCD with a new distance value, you can just output the 5 characters of the new value. This method will make a pretty display that will not flicker as the numbers change.

```
Nokia5110_SetCursor(3,2);  // spot for number
Nokia5110_OutUDec(myDistance);
```

Use a debugging profile to measure how long it takes to execute **Nokia5110_OutUDec.** Knowing that the 12 MHz SPI clock is 12 MHz, explain why this measurement is reasonable.

11.5 Troubleshooting

The LCD does not display characters:

- Check all the connections between LaunchPad and LCD.
- Make sure RESET is high.
- Run a simple main program that calls **lcdcommandwrite** over and over with the same command. Use a logic analyzer or scope to verify all five signals from LaunchPad to LCD are proper. Section 12.1 of the data sheet describes the expected SPI protocol.

Back light does not operate:

- Check the ground.
- Verify you have pin 1 properly identified and haven't wired it backwards.

11.6 Things to think about

In this section, we list thought questions to consider after completing this lab. These questions are meant to test your understanding of the concepts in this lab.

- What does it mean that this interface is serial? Why is serial important?
- What does it mean that this interface is synchronous? Why is synchronous important?
- What is the purpose of each of these three files: **Nokia5110.h**, **Nokia5110.c**, and **Lab11_LCDmain.c**?
- How long does it take to execute **Nokia5110_OutUDec**? Is it appropriate to call this function during an ISR, or is it better to always perform LCD output in the main program?

Lab: Liquid Crystal Display

11.7 Additional challenges

In this section, we list additional activities you could do to further explore the concepts of this module. You could extend the system or propose something completely different. For example,

- Add an output function for 16-bit signed numbers.
- Add an output function for signed 32-bit numbers
- Interface a different LCD, such as the ST7735R
- Create a set of functions that plots data versus time on the LCD

11.8 Which modules are next?

Modules 1-11 have introduced the basics of the microcontroller. The next set of modules allow for more complex functionality for the robot.

Module 12) interface the motors to the robot.
Module 13) write software to adjust power to the motors.
Module 14) use interrupts to detect collisions in real time
Module 15) interface IR distance sensors to measure distance to the wall.
Module 16) interface tachometers to measure wheel speed.

11.9 Things you should have learned

In this section, we review the important concepts you should have learned in this module:

- Understand busy-wait and know why busy-wait was used for this interface and not interrupts
- Synchronous serial protocol: how it works and why it is important
- The concept of minimally intrusive debugging

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265