7

Dot Matrix LCD Display Module

7.1 Introduction

In this experiment, you are going to implement the program that drives 16x2 dot matrix LCD on the experiment board. Your program is going to use a predefined char array as an input and display the string using LCD.

7.2 Background Information

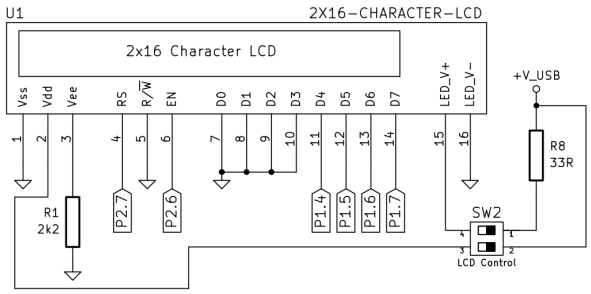


Fig. 7.1. Connection between LCD and MSP430

There are two working modes of LCD display which require 8-bit and 4-bit connection bandwidth respectively. Connection between the LCD and the MSP430 LaunchPad is given in Figure 7.1. Since, the only upper nibble (D4-D7) of data bus are wired to microcontroller, we should use 4-bit working mode to utilize the LCD. Table 7.1 contains list of commands for driving LCD. Please, remember that the LCD works in 8-bit mode by default. There are two things you need to do in this lab: Firstly you should configure the LCD display in order to communicate in 4-bit mode. Secondly you should send 8-bit ASCII characters as nibbles (4 bits) to display using the specific instruction. The flow chart that shows the steps of initialization and configuration of the LCD is given at the end of the experiment. More detail about this flow chart could be found in this link. In addition to this, the datasheet of the LCD display will be available in Ninova.

Instruction	**Code**										**Description**	**Execution time**
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position (address 0).	1.64mS
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position (address 0). Also returns display being shifted to the original position. DDRAM contents remains unchanged.	1.64mS
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction (I/D), specifies to shift the display (S). These operations are performed during data read/write.	40uS
Display On/Off control	0	0	0	0	0	0	1	D	C	B	Sets On/Off of all display (D), cursor On/Off (C) and blink of cursor position character (B).	40uS
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*	Sets cursor-move or display-shift (S/C), shift direction (R/L). DDRAM contents remains unchanged.	40uS
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display line (N) and character font(F).	40uS
Set DDRAM address	0	0	1	DDRAM address							Sets the DDRAM address. DDRAM data is sent or received after this setting.	40uS
Read busy-flag and address counter	0	1	BF	DDRAM address							Reads Busy-flag (BF) indicating internal operation is being performed and reads address counter contents.	0uS
Write to CGRAM or DDRAM	1	0	write data								Writes data to CGRAM or DDRAM.	40uS
Read from CGRAM or DDRAM	1	1	read data								Reads data from CGRAM or DDRAM.	40uS

Table 7.1. Instruction Set of LCD, retrieved from the link.

7.3 Experiment

In the experiment, you are requested to implement a *print* function which uses char arrays as inputs and show them on the LCD display. An example array is given as follows. This array contains "\n" and "\0" characters which represent "break line" and "end of string" respectively. Your function should be able to interpret these characters and create the output given as in Figure 7.2.

```
1           .data
2 string    .byte "ITU - Comp. Eng.",0Dh,"MC Lab. 2017",00h
```

Fig. 7.2. Output on the Display for the given input array

The list of subroutines are given below to assist you while implementing your code. You can construct your program by using these subroutines as your building blocks or you may choose different approach to implement the requested program. In other words, these subroutines are not mandatory to implement, you are free to choose any other approach/subroutine set to implement your program.

1 **initLCD:** The steps of initialization and configuration (are given in the flow chart) can be implemented in this subroutine. However, you should be aware of the initial state of the LCD while writing this function. Initially, the LCD works in 8-bit mode until you configure it otherwise. Thus, you should send your first commands accordingly.
2 **sendCMD:** Works in 4 bit mode, first loads the upper nibble of the command to the output port and calls **trigerEN** function to send the data to LCD, then repeats the same steps for the lower nibble.
3 **sendDATA:** Works in 4 bit mode, differs from sendCMD in terms of **RS** input, similarly it first it loads the upper nibble of the command to the output port and calls **trigerEN** function to send the data to LCD, then repeats the same steps for the lower nibble.
4 **delay:** busy-wait (loop) function to create necessary delays which is mandatory for LCD to function properly. You can create the delays by modifying the 1 second delay block provided in Experiment 5.
5 **trigerEN:** first it changes the value of EN to high (1) then it changes it back to low (0).

Remember to use the trigerEN subroutine while sending the configuration commands in the flowchart.

7.4 Report

Prepare your report by using the guidelines and the report template which are posted on Ninova e-Learning System.

During the experiment, please do not forget to take notes about the critical points of the implementations in order to write a proper report for the experiment. Additionally, if there were any complications which affect your performance during the experiment, please also indicate these difficulties in your report.

Character Mode Liquid Crystal Display Module
Initialization by Instruction - 4-bit data interface

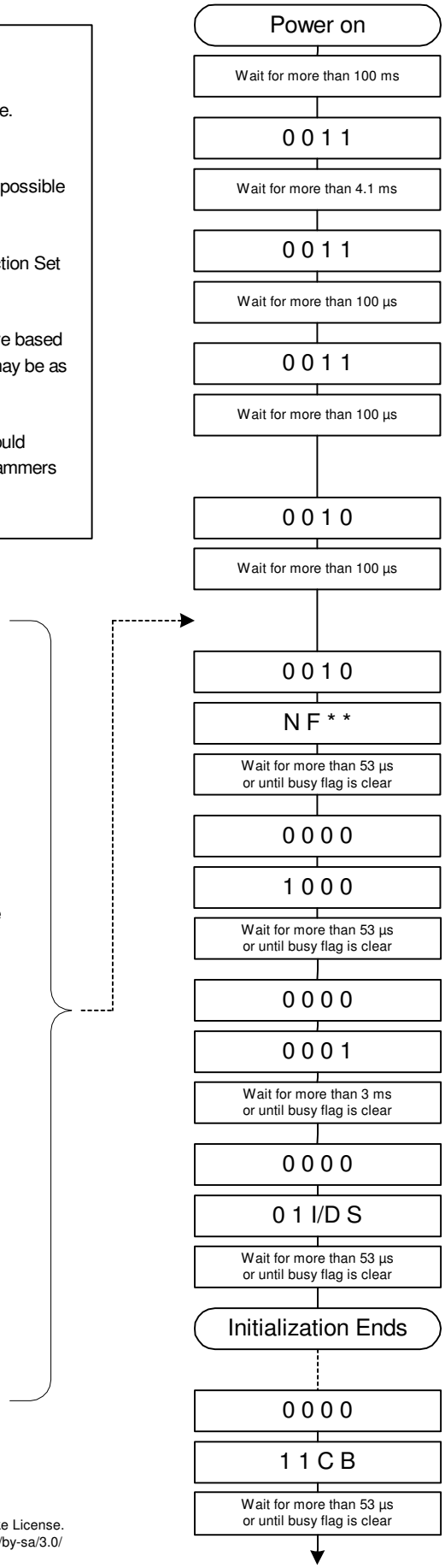
Notes:
RS = 0 to select the Instruction register.
R/W = 0 so that data is written to the LCD module.

The second and third 100 µs time delays are not documented, this figure is speculation, it may be possible to check the busy flag here.

N and F must be set in the first non-special Function Set instruction and cannot be changed subsequently

All time delays specified after the Function Set are based on worst case instruction execution time (clock may be as low as 190 kHz).

The first Display ON/OFF Control instruction should probably be performed as specified (some programmers set D, C, and B here).



The device is in 8-bit mode when powered-up, and it remains in that mode until this point.

Up to this point the device reads all eight data pins each time the enable pin is pulsed.

The four bits shown in the flowchart are the relevant ones and they should be placed on the upper four data lines.

The lower four inputs are supposed to be grounded but they will be ignored in any case.

At this point the device switches to the 4-bit mode.

Beyond this point the device reads only the upper four data pins each time the enable pin is pulsed.

The device will temporarily store the first group of four data bits that it receives. After it receives the second group of four data bits it will reassemble them and execute the resulting instruction.

No time delay is required between the sending of the two groups of bits.