ELEC 291 (20C)

ECE Design Studio I

Lab 3: Keypad and LCD Display (Make Arduino Display)

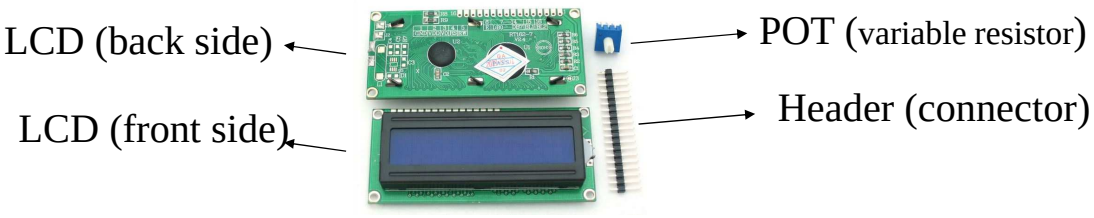
ECE – UBC

2015 W2

January 2016
Electrical and Computer Engineering
University of British Columbia

Objective

- This lab is to use a keypad, and an LCD to equip the Arduino with a standalone display. Any alphanumerical info or data can be displayed by the Arduino on the LCD.
- We will use a standard 16x2 LCD.



- Soldering: be careful, and follow the safety instructions
- Another new component: numeric keypad

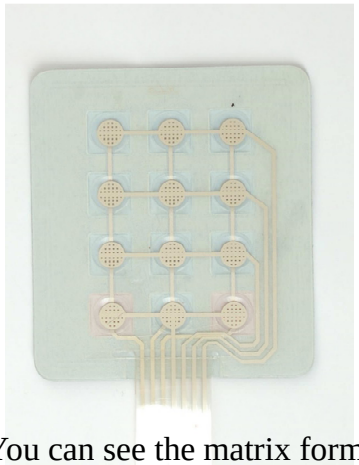
Keypad

- Keypads are basic devices for user input. This numeric matrix keypad has 12 buttons, arranged in a telephone-line 3x4 grid. The keys are connected into a matrix, so you only need 7 pins (3-columns and 4-rows) to use the keypad.

Check the connectivity using a mustimeter

ROWs 0 to 3

COLs 0 to 2

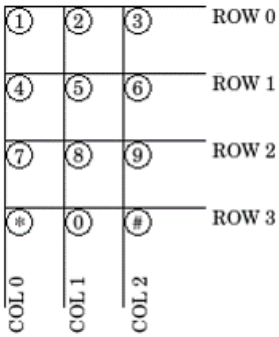


You can see the matrix format from the back of the keypad

Keypad - Matrix Format

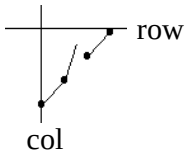
- The buttons are setup in a matrix format as shown below. The matrix has 4 rows and 3 columns here.
- Each square with a number or letter on it is a push to make switch, which connects the a corresponding horizontal wire (row) with the corresponding vertical wire (column).
- By using 7 pins, the embedded system can detect which of the 12 buttons are pressed.

An example matrix configuration



Keypad - programming

- The program can scan the keypad to detect which button is pressed.



- Consider one button at a specific *row* and *col*.
  - Assume *row* is attached to IO pin A.
  - For the scan, one can set this pin A to INPUT_PULLUP. This mode will use the internal pullup resistor. Read documentation for proper use.
  - Also assume that *col* is attached to IO pin B. We set pin B to be OUTPUT, and put LOW on it for the scan.
  - Now if we read from pin A, if the switch is not pressed we will read HIGH (due to the pull-up resistor), but if the switch is pressed, we will read LOW (since when pressed, this *row* and *col* are connected).
- You will need to scan for all buttons as above. Consider timing and blocking effect in the design of your code.
- What to do if two buttons are pressed together? You can either pick one of them (the first one detected) or reject multiple key pressed. The former is the usual practice.

Notes on Arduino IO pins

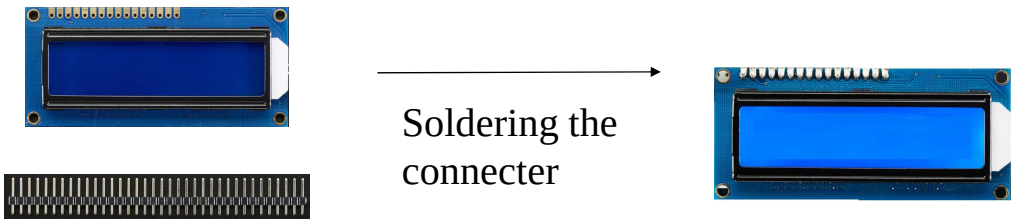
- Pins 0 to 13: digital IO
  - Pins 0 and 1 are also used for Rx/Tx. So they cannot be used as IO pins if/while you are using these commands or serial monitor.
  - Pins 3, 5, 6, 9, 10, 11 can be set to be analog output (with PWM).
  - Pins 13 is also connected to the built-in LED.
- Pins A0 to A5: analog input
  - When used as analog input: a 6 channel 10-bit analog to digital converter
  - These pins can also be used as digital IO. See documentations for more info …

LCD

- Our LCD (liquid crystal display) is a standard 16x2 character LCD.
  - 16 characters wide, 2 rows
  - White text on blue background
  - Pins are documented on the back of the LCD to assist in wiring it up
  - Single LED backlight included can be dimmed easily with a resistor or PWM and uses much less power than LCD with EL (electroluminescent) backlights
  - Can be controlled with as few as 6 digital lines
  - Built in character set supports English text, see the datasheet for the full character set
  - Use the included 10k potentiometer and pin header

LCD - Soldering

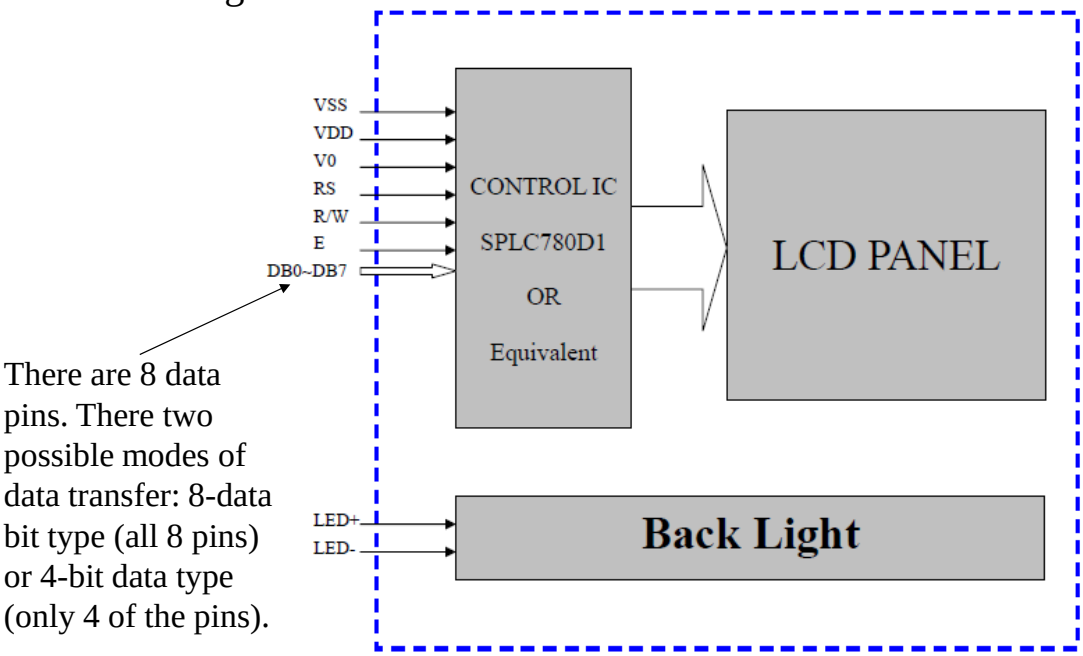
- You will need to solder the header pin connector to the LCD.



- To be safe, it is your responsibility to make sure you and everybody around you are wearing safety glasses. → We will use MCLD 303/309 for soldering (please follow instructions).
- Be careful to solder on the correct side of LCD. Practice first.
- Do not keep the iron for long on a pin, it may damage the LCD.
- Keep the soldering iron clean, keep it in place when not used, and turn it off when you are done.

LCD Block diagram

- Block diagram:



There are 8 data pins. There two possible modes of data transfer: 8-data bit type (all 8 pins) or 4-bit data type (only 4 of the pins).

LCD Connection to Arduino

- This is a sample Fritzing breadboard diagram (4-bit operation type/mode). Of course, you may use different IO pins. For the 8-bit operation mode, you will need to use 4 additional IO pins.

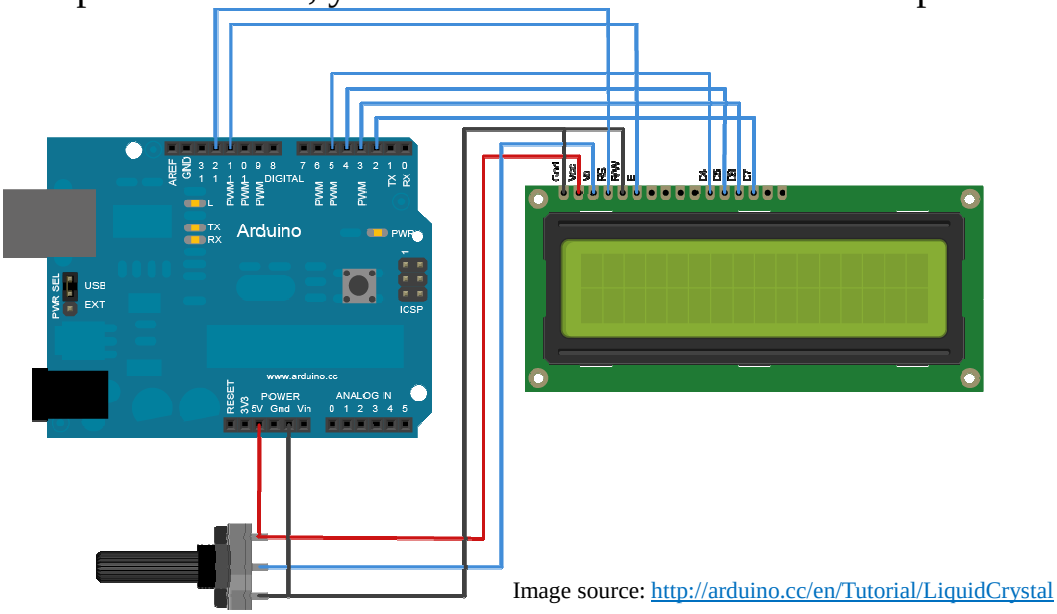
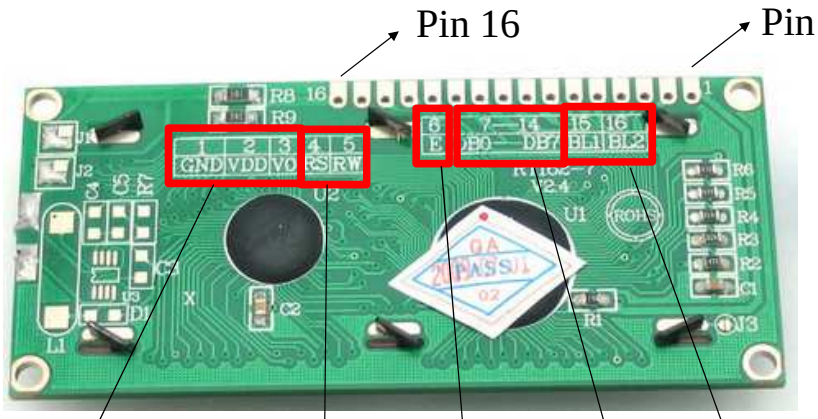


Image source: http://arduino.cc/en/Tutorial/LiquidCrystal

LCD Pins

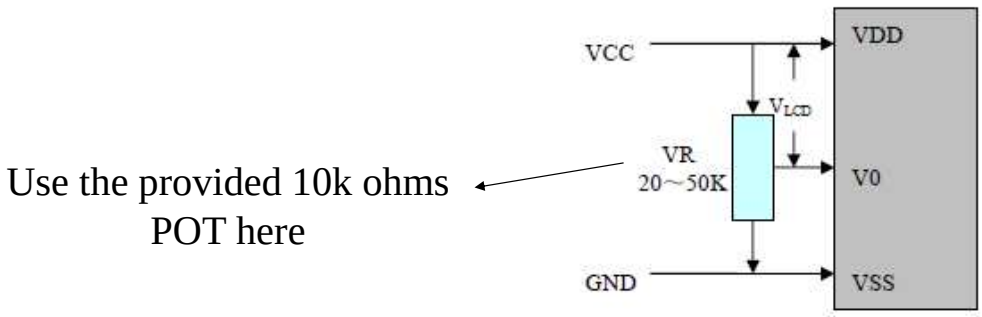
- Info on the back of LCD

PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
NAME	VSS	VDD	V0	RS	R/W	E	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	LED+	LED−



LCD Pins: 1, 2, 3

- Power (VCC: 5V and GND) and a typical V0 connections for display contrast:



- Using the POT, adjust V0 for optimal display appearance.
  - You may see nothing if this is not set correctly.
  - See the data sheet for more info

LCD Pins: 4, 5, 6

PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
NAME	VSS	VDD	V0	RS	R/W	E	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	LED+	LED−

- Pin 4 (RS): It is used to signal the LCD whether data is going to be sent or an instruction
  - RS = H (1): Data Register (for read and write)
  - RS = L (0): Instruction Register (for write), Busy flag-Address Counter (for read).
- Pin 5 (R/W): It is used to signal the LCD whether you want to read from or write to it
  - R/W = H : Read mode.
  - R/W = L: Write mode.
- Pin 6 (E): An enable signal (to tell LCD whether it should read/write the data line)

As you see in slide 10, if there is no need for read, we can connect this to ground.

LCD Pins: data and backlight

PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
NAME	VSS	VDD	V0	RS	R/W	E	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	LED+	LED−

- There are two types of data operations:
  - 4-bit and 8-bit operations.
- Using 4-bit MPU, the interfacing 4-bit data is transferred by 4-busline (DB4～DB7).
  - thus, DB0 to DB3 bus lines are not used.
- You must implement the 4-bit type for the demo. You should also know the 8-bit operation type.
- Pins 15 and 16 are used for backlight LED of the LCD.

LCD Programming

- For the start, you may use the LiquidCrystal Library of Arduino to program and display on the LCD. This library provides a full list of functions that you may use to interface with an LCD.
  - Reference: http://arduino.cc/en/Reference/LiquidCrystal
  - Sample code: http://arduino.cc/en/Tutorial/LiquidCrystal
- However, in this lab eventually you must write the code yourself (that is without using LiquidCrystal.h) in C to get mark for the LCD.
  - That is you need to write your own functions that initializes the LCD, clears the LCD, sets the cursor position, writes data or an instruction to it, …

LCD Initialization

- You will need to first follow the initialization procedure provided in the datasheet.
  - Implement the initialization steps following the diagram.
  - Include delays as specified.
  - Note that in the 4-bit interface initialization (see datasheet; a snapshot is shown below), the interface is assumed 8-bit long initially.

A typical initialization consist of

- A **Function Set** command,
- preferably followed by an **Entry Mode Set**, **Display Control** and a **Clear Display**.

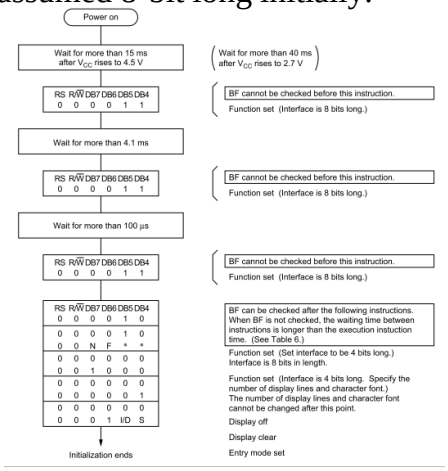


Figure 24 4-Bit Interface

LCD Command Table

See page 11 of the datasheet for the “command table”

- For example, to clear the display, the instruction code must be:

Instruction	Instruction Code										Description	Execution time (Temp = 25℃)		
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		Fosc= 190KHz	Fosc= 270KHz	Fosc= 350KHz
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM and set DDRAM address to "00H" from AC	2.16ms	1.52ms	1.18ms

- Note that a proper delay must be use when sending a command to allow the LCD to get and process it.
- See the timing diagrams in the datasheet

LCD writing Sequence

- In order to write to the LCD, follow this sequence:
  - May need to set **RS:** dependent on data or instruction
  - May need to set **R/W:** dependent on read or write
  - Write the data/instruction to the data pins
  - Set **E** to high to begin write cycle
  - Delay to allow LCD to complete reading
  - Set **E** to low to finish the write cycle

  } enable

  - You will need to set RS and R/W depending on whether this is a data or an instruction and whether this is read or write, respectively.

LCD Programming Example

- LCD must be initialized after power-on. So as an example, you need to write a function that performs this initialization.

- Assuming that you have written a **commandLCD**() function to send commands to LCD, and **writeLCD**() to write data to it, then an example Function Set command would be: (see page 11 of datasheet)

```
commandLCD(0x28);  // Function Set Command:
                   // 4-bit interface, 2 display lines, 5x8 font
```


Displaying Data

See pages 11, 14 and 15 of the datasheet

- See page 11 for the format to "write data to RAM"
- The 80-bit RAM is normally used for storing display data. Those RAM not used for display data can be used as general data RAM. Its address is configured in the Address Counter.

2 LINES X 16 CHARACTERS PER LINE

Char.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

- Page 15 of the datasheet shows the character generator ROM

LCD Precautions

See page 17 of the datasheet

- The display panel is made of glass. Do not subject it to a mechanical shock by dropping it from a high place, etc.
- If the display panel is damaged and the liquid crystal substance inside it leaks out, … If the substance come into contact with your skin or clothes promptly wash it off using soap and water.
- The polarizer covering the display surface of the LCD module is soft and easily scratched. Handle this polarize carefully.
- To prevent destruction of the elements by static electricity, be careful to maintain an optimum work environment.
- …

References

- Arduino Reference: http://arduino.cc/en/Reference/HomePage
- Arduino: http://arduino.cc/
- See Arduino tutorials on LCD and Keypad
- Datasheets:
  - See the datasheets posted on Connect