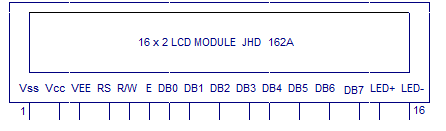
**Interfacing 16x2 LCD to Arduino – Display Text and Characters on LCD Screen using Arduino**

**A 16×2 LCD Module** can display 32 ASCII characters in 2 lines (16 characters in 1 line). Other commonly used LCD displays are 20×4 Character LCD, Nokia 5110 LCD module, 128×64 Graphical LCD Display and 2.4 inch TFT Touch screen LCD display.

JHD162A is the LCD module used here. JHD162A is a 16×2 LCD module based on the **HD44780 driver from Hitachi**. The JHD162A has 16 pins and can be operated in 4-bit mode (using only 4 data lines) or 8-bit mode (using all 8 data lines). we are using the LCD module in 4-bit mode to display a plain text messages on the LCD module using arduino .

**16×2 LCD Module Pin Out Diagram**



The name and functions of each pin of the 16×2 LCD module is given below.

**Pin1(Vss)**:Ground pin of the LCD module.

**Pin2(Vcc)**: Power to LCD module (+5V supply is given to this pin).

**Pin3(VEE)**:Contrast adjustment pin. This is done by connecting the ends of a 10K potentimeter to +5V and ground and then connecting the slider pin to the VEE pin. The voltage at the VEE pin defines the contrast. The normal setting is between 0.4 and 0.9V.

**Pin4(RS)**:Register select pin. The JHD162A has two registers namely **command register** and **data register**. Logic HIGH at RS pin selects data register and logic LOW at RS pin selects command register. If we make the RS pin HIGH and feed an input to the data lines (DB0 to DB7), this input will be treated as data to display on LCD screen. If we make the RS pin LOW and feed an input to the data lines, then this will be treated as a command ( a command to be written to LCD controller – like positioning cursor or clear screen or scroll).

**Pin5(R/W)**: Read/Write modes. This pin is used for selecting between read and write modes. Logic HIGH at this pin activates read mode and logic LOW at this pin activates write mode.

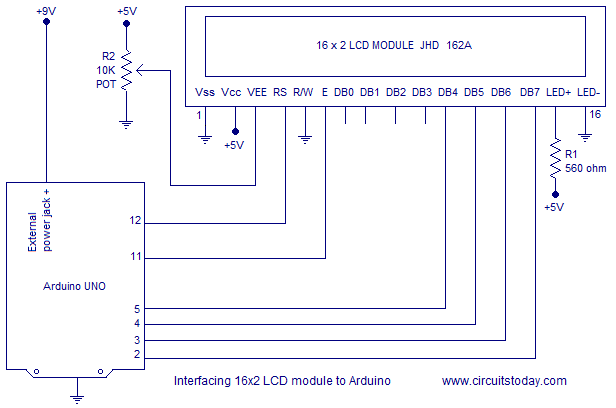
**Pin6(E)**: This pin is meant for enabling the LCD module. A HIGH to LOW signal at this pin will enable the module.

**Pin7(DB0) to Pin14(DB7)**:  These are data pins. The commands and data are fed to the LCD module though these pins.

**Pin15(LED+)**: Anode of the back light LED. When operated on 5V, a 560 ohm resistor should be connected in series to this pin. In arduino based projects the back light LED can be powered from the 3.3V source on the arduino board.

**Pin16(LED-)**: Cathode of the back light LED.

**Circuit Diagram : Arduino to 16x2 LCD:**



**Description:**

RS pin of the LCD module is connected to digital pin 12 of the arduino. R/W pin of the LCD is grounded. Enable pin of the LCD module is connected to digital pin 11 of the arduino. In this project, the **LCD module and arduino are interfaced in the 4-bit mode**. This means only four of the digital input lines( DB4 to DB7)  of the LCD are used. This method is very simple, requires less connections and you can almost utilize the full potential of the LCD module. Digital lines DB4, DB5, DB6 and DB7 are interfaced to digital pins 5, 4, 3 and 2 of the Arduino. The 10K potentiometer is used for adjusting the contrast of the display. 560 ohm resistor R1 limits the current through the back light LED. The arduino can be powered through the external power jack provided on the board. +5V required in some other parts of the circuit can be tapped from the 5V source on the arduino board. The arduino can be also powered from the PC through the USB port.

**Program to print hello, class:**

#include<LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // sets the interfacing pins

void setup()

{

lcd.begin(16, 2); // initializes the 16x2 LCD

}

void loop()

{

lcd.setCursor(0,0); //sets the cursor at row 0 column 0

lcd.print("16x2 LCD MODULE"); // prints 16x2 LCD MODULE

lcd.setCursor(2,1); //sets the cursor at row 1 column 2

lcd.print("HELLO WORLD"); // prints HELLO WORLD

}

**Liquid Crystal Library functions description:**

Library  “LiquidCrystal.h” is used for easily controlling the LCD module using Arduino board with the help of built in methods defined inside the library For example, data string can be printed on the LCD module by merely calling a method **lcd.print()**.

If you want to print “Hello World” at row 1, starting from column 3; first set the cursor at the desired position using method **lcd.setCursor(1,3)** and then write the command to print the characters as **lcd.print(“Hello World”);**

The LiquidCrystal.h library provides functions/methods for almost all applications like printing a string, setting the cursor, initializing the LCD, scrolling the display, auto scroll, clear LCD, blink cursor etc.

[**lcd.begin()**](https://www.arduino.cc/en/Reference/LiquidCrystalBegin) – is called to initialize the lcd screen and to pass the dimension of lcd screen (columns, rows) as parameters of the invoked method.

**Program for scrolling the LCD screen using Arduino.**

A simple program for scrolling a text message on the LCD screen using arduino is shown here. This is done using the “[**scroll()**](https://www.arduino.cc/en/Reference/LiquidCrystalScrollDisplayLeft)” method defined inside LiquidCrystal.h library.

For example the method “**lcd.scrollDisplayRight()**” will scroll the display to right and the method”**lcd.scrollDisplayLeft()**” will scroll the display to left. A “for” loop is used for selecting the number of positions to scroll at a time. In the program shown below, it is chosen to be 2 because the text to be displayed is comparatively long. For shorter texts more number of positions must be scrolled at a time to get a smooth display.

Program position left/ right :

#include <LiquidCrystal.h>

int pos=0; // variable to hold cursor position

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()

{

lcd.begin(16, 2); //initializes 16x2 LCD

lcd.print("16x2 LCD MODULE & ARDUINO-UNO"); //text to display

}

void loop()

{

for(pos=0; pos<2; pos++)

{

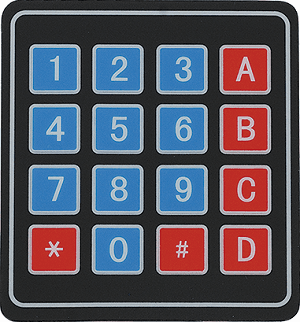
lcd.scrollDisplayLeft(); //scrolls display left by two positions

}

delay(800); //sets the speed at which display moves

}

**4x4 keypad interfacing with Arduino UNO**

****

Keypad is used as an input device to read the key pressed by the user and to process it.

4x4 keypad consists of 4 rows and 4 columns. Switches are placed between the rows and columns.

A key press establishes a connection between the corresponding row and column, between which the switch is placed.

**Functions Used:**

1.  makeKeymap(keys)

* This function is used to initialize the internal keymap to be equal to the user defined key map (in function syntax given above, keys).

2.  Keypad customKeypad = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols)

* This defines an object customKeypad of the class Keypad and initializes it.
* *rowPins* and *colPins* are the pins on Arduino to which the rows and columns of the keypad are connected to.
* *rows* and *cols* are the number of rows and columns the keypad has.

3.  customKeypad.getKey()

* This function is used to identify which key is pressed on the keypad.

**Program:**

#include <Keypad.h>

const byte ROWS = 4; /\* four rows \*/

const byte COLS = 4; /\* four columns \*/

/\* define the symbols on the buttons of the keypads \*/

char hexaKeys[ROWS][COLS] = {

{'0','1','2','3'},

{'4','5','6','7'},

{'8','9','A','B'},

{'C','D','E','F'}

};

byte rowPins[ROWS] = {10, 11, 12, 13}; /\* connect to the row pinouts of the keypad \*/

byte colPins[COLS] = {6, 7, 8, 9}; /\* connect to the column pinouts of the keypad \*/

/\* initialize an instance of class NewKeypad \*/

Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){

Serial.begin(9600);

}

void loop(){

char customKey = customKeypad.getKey();

if (customKey){

Serial.println(customKey);

}

}