Makerguides.com

How to use a 16×2 character LCD with Arduino

Written by Benne de Bakker (https://www.makerguides.com/author/benne-de-bakker/)

This tutorial includes everything you need to know about controlling a character LCD with Arduino. I have included a wiring diagram and many example codes. These displays are great for displaying sensor data or text and they are also fairly cheap.

The first part of this article covers the basics of displaying text and numbers. In the second half, I will go into more detail on how to display custom characters and how you can use the other functions of the **LiquidCrystal** Arduino library.

As you will see, you need quite a lot of connections to control these displays. I therefore like to use them with an I2C interface module (https://amzn.to/2ZKMSpv) mounted on the back. With this I2C module, you only need two connections to control the LCD. Check out the tutorial below if you want to use an I2C module as well:

Recommended tutorials:

- How to control a character I2C LCD with Arduino (https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/)

Supplies

Hardware components

(https://www.amazon.com/HiLetgo-Display-Backlight-Controller-Character/dp/B00HJ6AFW6/ref=as_li_ss_tl?keywords=lcd+16x2&qid=1562768812&s=gateway&sr=8-3&linkCode=ll1&tag=makerguides-20&linkId=560296518db01cb992dfe0ac9cc8e4d7&language=en_US)	16×2 character Character/dp/B keywords=lcd+ 3&linkCode=ll1 20&linkId=560
(https://www.amazon.com/SunFounder-Serial-Module-Arduino-Mega2560/dp/B071W8SW9R/ref=as_li_ss_tl?keywords=lcd+20x4&qid=1562768855&s=gateway&sr=8-4&th=1&linkCode=ll1&tag=makerguides-20&linkId=4f5f1f53e51e48cb678ff5c3fadab478&language=en_US)	20×4 character Mega2560/dp/ keywords=lcd+ 4&th=1&linkCo 20&linkId=4f5f
(https://amzn.to/374aJjX)	Arduino Uno R
	Breadboard (ht
	Jumper wires (
	10 kΩ potentio Potentiometer- keywords=10+ 3&linkCode=ll1 20&linkId=88f8

	USB cable typ

Software

	Arduino IDE (https://www.arduino.cc/en/Main/Software)

Hardware overview

These LCDs are available in many different sizes (16×2 1602, 20×4 2004, 16×1 etc.), but they all use the same HD44780 parallel interface LCD controller chip (https://en.wikipedia.org/wiki/Hitachi_HD44780_LCD_controller) from Hitachi. This means you can easily swap them. You will only need to change the size specifications in your Arduino code.

16×2 LCD Specifications

Operating voltage	5 V
Controller	Hitachi HD44780 LCD controller
Screen resolution	2-lines x 16 characters
Character resolution	5 x 8 pixels
Module dimensions	80 x 36 x 12 mm
Viewing area dimensions	64.5 x 16.4 mm
Cost	Check price (https://amzn.to/2UmgCE6)

For more information, you can check out the datasheets below. The 16×2 and 20×4 datasheets include the dimensions of the LCD and in the HD44780 datasheet you can find more information about the Hitachi LCD driver.

16×2 Character LCD Datasheet

20×4 Character LCD Datasheet

HD44780 Datasheet

16×2 LCD pinout

The LCD has 16 connection pins, numbered 1-16 from left to right.

The pins at the top of the display are numbered 1-16 from left to right.

The pinout of a standard HD44780 LCD is given in the table below:

Pin no.	Symbol	Connection	Function
1	VSS	GND Arduino	Signal ground
2	VDD	5 V Arduino	Logic power for LCD
3	V0	10 kΩ potentiometer	Contrast adjustment
4	RS	Pin 2 Arduino	Register select signal
5	R/W	GND Arduino	Read/write select signal
6	E	Pin 3 Arduino	Operation enable signal
7 – 14	D0 – D7	–	Data bus lines used for 8-bit mode
11 – 14	D4 – D7	Pin 4 – 7 Arduino	Data bus lines used for 4-bit mode
15	A (LED+)	5 V Arduino	Anode for LCD backlight
16	K (LED-)	GND Arduino	Cathode for LCD backlight

Testing the LCD and adjusting contrast

In order to test the display, you will need to make the connections as shown in the figure below.

Most LCDs have a built-in series resistor for the LED backlight. You should find it on the back of the LCD connected to pin 15 (Anode). **If your display doesn’t include a resistor, you will need to add one between 5 V and pin 15.** It should be safe to use a 220Ω resistor, but this value might make your display a bit dim. You can check the datasheet for the maximum current rating of the backlight and use this to select an appropriate resistor value.

Contrast adjustment wiring

After you have wired up the LCD, you will need to adjust the contrast of the display. This is done by turning the 10 kΩ potentiometer clockwise or counterclockwise.

Plug in the USB connector of the Arduino to power the LCD. You should see the backlight light up. Now rotate the potentiometer until one (16×2 LCD) or 2 rows (20×4 LCD) of rectangles appear.

Rotate the potentiometer until you see a row of rectangles appear.

You can tweak the contrast later if needed.

How to connect the LCD to Arduino UNO

In order to control the LCD and display characters, you will need to add a few extra connections. Check the wiring diagram below and the pinout table from the introduction of this article.

16×2 LCD with Arduino wiring diagram

We will be using the LCD in 4-bit mode, this means you don't need to connect anything to D0-D3. The R/W pin is connected to ground, this will pull the pin LOW and set the LCD to WRITE mode.

Once you have wired everything, we can start programming the LCD.

Arduino example code for character LCD

To control the LCD we will be using the LiquidCrystal (https://www.arduino.cc/en/Reference/LiquidCrystal) library. This library should come pre-installed with the Arduino IDE (https://www.arduino.cc/en/main/software). You can find it by going to **Sketch > Include Library > LiquidCrystal**.

The LiquidCrystal library comes with many built-in functions and makes controlling character LCDs super easy.

The example code below shows you how to display a message on the LCD. Next, I will show you how the code works and how you can use the other functions of the LiquidCrystal library.

```
/* Basic Arduino example code for displaying text on 16x2, 20x4 etc. character LCDs. More info: www.makerguides.com */

// Include the library:
#include <LiquidCrystal.h>

// Create an LCD object. Parameters: (RS, E, D4, D5, D6, D7):
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  // Specify the LCD's number of columns and rows. Change to (20, 4) for a 20x4 LCD:
  lcd.begin(16, 2);
}

void loop() {
  // Set the cursor on the third column and the first row, counting starts at 0:
  lcd.setCursor(2, 0);
  // Print the string 'Hello World!':
  lcd.print("Hello World!");
  // Set the cursor on the third column and the second row:
  lcd.setCursor(2, 1);
  // Print the string 'LCD tutorial':
  lcd.print("LCD tutorial");
}
```

You should see the following output on the LCD:

How the code works

After including the library, the next step is to create a new instance of the LiquidCrystal class. The is done with the function `LiquidCrystal(rs, enable, d4, d5, d6, d7)`. As parameters we use the Arduino pins to which we connected the display. Note that we have called the display ‘lcd’. You can give it a different name if you want like ‘menu_display’. You will need to change ‘lcd’ to the new name in the rest of the sketch.

```
// Include the library:
#include <LiquidCrystal.h>

// Create an LCD object. Parameters: (RS, E, D4, D5, D6, D7):
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);
```

In the `setup()` the LCD is initiated with the function `begin(cols,rows)`. When using a 20×4 LCD change this line to lcd.begin(20,4);

```
void setup() {
  // Specify the LCD's number of columns and rows. Change to (20, 4) for a 20x4 LCD:
  lcd.begin(16, 2);
}
```

In the `loop()` the cursor is set to the third column and first row of the LCD with `lcd.setCursor(2,0)` . Note that counting starts at 0, and the first argument specifies the column. If you do not specify the cursor position, the text will be printed at the default home position (0,0) if the display is empty, or behind the last printed character.

Next, the string ‘Hello World!’ is printed with `lcd.print("Hello World!")` . Note that you need to place quotation marks (” “) around the text. When you want to print numbers or variables, no quotation marks are necessary.

```
void loop() {
  // Set the cursor on the third column and the first row, counting starts at 0:
  lcd.setCursor(2, 0);
  // Print the string 'Hello World!':
  lcd.print("Hello World!");
  // Set the cursor on the third column and the second row:
  lcd.setCursor(2, 1);
  // Print the string 'LCD tutorial':
  lcd.print("LCD tutorial");
}
```

If you want to see an example for displaying (changing) variables on the LCD, check out my tutorial for the HC-SR04 ultrasonic distance sensor:

- How to use a HC-SR04 Ultrasonic Distance Sensor with Arduino (https://www.makerguides.com/hc-sr04-arduino-tutorial/#example-code-hc-sr04-with-dht11-and-i2c-lcd)

In the example I used an I2C LCD display but the code after the setup is the same for both.

Other functions of the LiquidCrystal library

The LiquidCrystal Arduino library has many other built-in functions which you might find useful. You can find an overview of them below with explanation and some code snippets.

clear()

Clears the LCD screen and positions the cursor in the upper-left corner (first row and first column) of the display. You can use this function to display different words in a loop.

```
#include <LiquidCrystal.h>

// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
}

void loop() {
  lcd.clear();
  lcd.print("Monday");
  delay(2000);
  lcd.clear();
  lcd.print("13:45");
  delay(2000);
}
```

home()

Positions the cursor in the top-left corner of the LCD. Use clear() if you also want to clear the display.

cursor()

Displays the LCD cursor: an underscore (line) at the position of the next character to be printed.

noCursor()

Hides the LCD cursor. The following example creates a blinking cursor at the end of “cursor()”.

```
#include <LiquidCrystal.h>

// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
  lcd.print("cursor()");
}

void loop() {
  lcd.cursor();
  delay(500);
  lcd.noCursor();
  delay(500);
}
```

blink()

Creates a blinking block style LCD cursor: a blinking rectangle at the position of the next character to be printed.

noBlink()

Disables the block style LCD cursor. The following example displays the blinking cursor for 5 seconds and then disables it for 2 seconds.

```
#include <LiquidCrystal.h>

// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
  lcd.print("blink() example");
}

```

```
void loop() {
  lcd.blink();
  delay(5000);
  lcd.noBlink();
  delay(2000);
}
```

display()

This function turns on the LCD screen and displays any text or cursors that have been printed to the display.

noDisplay()

This function turns off any text or cursors printed to the LCD. The text/data is not cleared from the LCD memory. This means it will be shown again when the function display() is called.

The following example creates a blinking text effect.

```
#include <LiquidCrystal.h>

// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
  lcd.print("Blinking text");
}

void loop() {
  lcd.display();
  delay(2000);
  lcd.noDisplay();
  delay(2000);
}
```

write()

This function can be used to write a character to the LCD. See the section about creating and displaying custom characters below for more info.

scrollDisplayLeft()

Scrolls the contents of the display (text and cursor) one space to the left. You can use this function in the loop section of the code in combination with delay(500), to create a scrolling text animation.

```
#include <LiquidCrystal.h>

// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
  lcd.print("scrollDisplayLeft() example");
}

void loop() {
  lcd.scrollDisplayLeft();
  delay(500);
}
```

scrollDisplayRight()

Scrolls the contents of the display (text and cursor) one space to the right.

autoscroll()

This function turns on automatic scrolling of the LCD. This causes each character output to the display to push previous characters over by one space. If the current text direction is left-to-right (the default), the display scrolls to the left; if the current direction is right-to-left, the display scrolls to the right. This has the effect of outputting each new character to the same location on the LCD.

The following example sketch enables automatic scrolling and prints the character 0 to 9 at the position (16,0) of the LCD. Change this to (20,0) for a 20×4 LCD.

```
#include <LiquidCrystal.h>

// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
}

void loop() {
  lcd.autoscroll();
  lcd.setCursor(16, 0);
  for (int x = 0; x < 10; x++) {
    lcd.print(x);
    delay(500);
  }
  lcd.clear();
}
```

noAutoscroll()

Turns off automatic scrolling of the LCD.

leftToRight()

This function causes text to flow to the right from the cursor, as if the display is left-justified (default).

rightToLeft()

This function causes text to flow to the left from the cursor, as if the display is right-justified.

How to create and display custom characters?

With the function `createChar()` it is possible to create and display custom characters on the LCD. This is especially useful if you want to display a character that is not part of the standard ASCII character set.

Technical info: LCDs that are based on the Hitachi HD44780 LCD controller have two types of memories: CGROM and CGRAM (Character Generator ROM and RAM). CGROM generates all the 5 x 8 dot character patterns from the standard 8-bit character codes. CGRAM can generate user-defined character patterns.

For 5 x 8 dot displays, CGRAM can write **up to 8 custom characters** and for 5 x 10 dot displays **4**. For more info see the datasheet.

Custom characters Arduino example code

The following example sketch creates and displays eight custom characters (numbered 0 – 7).

```
/* Example sketch to create and display custom characters on character LCD with Arduino and LiquidCrystal library. For more info see www.makerguides.com */

#include <LiquidCrystal.h>

// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

// Make custom characters:
byte Heart[] = {
  B00000,
  B01010,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000,
  B00000
};
byte Bell[] = {
  B00100,
  B01110,
  B01110,
  B01110,
  B11111,
  B00000,
  B00100,
  B00000
};
```

```
byte Alien[] = {
  B11111,
  B10101,
  B11111,
  B11111,
  B01110,
  B01010,
  B11011,
  B00000
};
byte Check[] = {
  B00000,
  B00001,
  B00011,
  B10110,
  B11100,
  B01000,
  B00000,
  B00000
};
byte Speaker[] = {
  B00001,
  B00011,
  B01111,
  B01111,
  B01111,
  B00011,
  B00001,
  B00000
};
byte Sound[] = {
  B00001,
  B00011,
  B00101,
  B01001,
  B01001,
  B01011,
  B11011,
  B11000
};
byte Skull[] = {
  B00000,
  B01110,
  B10101,
  B11011,
  B01110,
  B01110,
  B00000,
  B00000
};
byte Lock[] = {
  B01110,
  B10001,
  B10001,
  B11111,
  B11011,
  B11011,
  B11111,
  B00000
};

void setup() {
  // Specify the LCD's number of columns and rows:
  lcd.begin(16, 2);

  // Create a new characters:
  lcd.createChar(0, Heart);
  lcd.createChar(1, Bell);
  lcd.createChar(2, Alien);
  lcd.createChar(3, Check);
  lcd.createChar(4, Speaker);
  lcd.createChar(5, Sound);
  lcd.createChar(6, Skull);
  lcd.createChar(7, Lock);

  // Clears the LCD screen:
  lcd.clear();

  // Print a message to the lcd:
  lcd.print("Custom Character");
}

void loop() {
  // Print all the custom characters:
  lcd.setCursor(0, 1);
  lcd.write(byte(0));
  lcd.setCursor(2, 1);
  lcd.write(byte(1));
  lcd.setCursor(4, 1);
  lcd.write(byte(2));
  lcd.setCursor(6, 1);
  lcd.write(byte(3));
  lcd.setCursor(8, 1);
  lcd.write(byte(4));
  lcd.setCursor(10, 1);
  lcd.write(byte(5));
  lcd.setCursor(12, 1);
  lcd.write(byte(6));
  lcd.setCursor(14, 1);
  lcd.write(byte(7));
}
```

You should see the following output on the LCD:

How the code works

After including the library and creating the LCD object, the custom character arrays are defined. Each array consists of 8 bytes, 1 byte for each row. In this example 8 custom characters are created.

```
// Make custom characters:
byte Heart[] = {
  B00000,
  B01010,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000,
  B00000
};
```

When looking closely at the array, you will see the following. Each row consists of 5 numbers corresponding to the 5 pixels in a 5 x 8 dot character. A 0 means pixel off and a 1 means pixel on.

It is possible to edit each row by hand, but I recommend using this visual tool (https://maxpromer.github.io/LCD-Character-Creator/) on GitHub. This application automatically creates the character array and you can click on the pixels to turn them on or off.

In the `setup()` , the custom characters are created with `lcd.createChar(num, data)` .

The first argument in this function is the number of the custom character (0-7) and the second argument is the character array that we created.

```
  // Create a new characters:
  lcd.createChar(0, Heart);
  lcd.createChar(1, Bell);
  lcd.createChar(2, Alien);
  lcd.createChar(3, Check);
  lcd.createChar(4, Speaker);
  lcd.createChar(5, Sound);
  lcd.createChar(6, Skull);
  lcd.createChar(7, Lock);
```

In the `loop()` all the characters are displayed with `lcd.write()` . As a parameter we use the number of the character we reserved.

```
  lcd.setCursor(0, 1);
  lcd.write(byte(0));
```

Conclusion

In this article I have shown you how to use an alphanumeric LCD with Arduino. I hope you found it useful and informative. If you did, please **share it with a friend** that also likes electronics and making things!

I would love to know what projects you plan on building (or have already built) with these LCDs. If you have any questions, suggestions, or if you think that things are missing in this tutorial, **please leave a comment down below.**

Note that comments are held for moderation to prevent spam.

Beginner

13
SHARES

What to read next?

LM35 analog temperature sensor with Arduino tutorial (https://www.makerguides.com/lm35-arduino-tutorial/)

TMP36 analog temperature sensor with Arduino tutorial (https://www.makerguides.com/tmp36-arduino-tutorial/)

Arduino Nano Board Guide (Pinout, Specifications, Comparison) (https://www.makerguides.com/arduino-nano/)

The complete guide for DS18B20 digital temperature sensors with Arduino (https://www.makerguides.com/ds18b20-arduino-tutorial/)

How to use an IR receiver and remote with Arduino (https://www.makerguides.com/ir-receiver-remote-arduino-tutorial/)

Comments

Steve says
October 12, 2020 at 12:00 am (https://www.makerguides.com/character-lcd-arduino-tutorial/#comment-4152)

I have to let you know that this is the only site that show how to correctly hook up a 1602 LCD and display the code to make it run. I've struggled with this for years and have given up many times until today! Thanks a bunch!

Reply

Benne de Bakker says
October 14, 2020 at 11:46 am (https://www.makerguides.com/character-lcd-arduino-tutorial/#comment-4179)

Thanks!

Reply

Jacobus Slabbert says
August 24, 2020 at 3:49 pm (https://www.makerguides.com/character-lcd-arduino-tutorial/#comment-3600)

I have made a few custom Icons and animations. Including a House, thermometer, humidity symbol. Animations included a spinning cup anemometer (wind speed) and a rain storm.

Reply

Roberto Martinez says
July 13, 2020 at 3:04 am (https://www.makerguides.com/character-lcd-arduino-tutorial/#comment-3226)

I have build a thermostat and all works ok, I want to put a message of 5 seconds before the program begins, is it posible ?

Reply

Benne de Bakker says

July 14, 2020 at 7:15 am (https://www.makerguides.com/character-lcd-arduino-tutorial/#comment-3242)

Hi Roberto,

Yes, this is possible; you can put the code to display the message in the setup() section of the code. This code will only run once before the rest of the program starts.

Benne

Reply

Roberto Martinez says
July 14, 2020 at 11:36 am (https://www.makerguides.com/character-lcd-arduino-tutorial/#comment-3244)

thank you it was very usefull for me

Reply

Trackbacks

TM1637 4-Digit 7-Segment Display Arduino Tutorial (3 Examples) (https://www.makerguides.com/tm1637-arduino-tutorial/) says:
September 3, 2019 at 11:27 am (https://www.makerguides.com/character-lcd-arduino-tutorial/#comment-483)
[…] How to use a 16×2 character LCD with Arduino […]

DHT11/DHT22 Sensors with Arduino Tutorial (2 Examples) (https://www.makerguides.com/dht11-dht22-arduino-tutorial/) says:
August 20, 2019 at 9:02 am (https://www.makerguides.com/character-lcd-arduino-tutorial/#comment-419)
[…] How to use a 16×2 character LCD with Arduino […]

(https://www.makerguides.com/a4988-stepper-motor-driver-arduino-tutorial/)

How to control a stepper motor with A4988 driver and Arduino (https://www.makerguides.com/a4988-stepper-motor-driver-arduino-tutorial/)

(https://www.makerguides.com/dht11-dht22-arduino-tutorial/)

How to use DHT11 and DHT22 Sensors with Arduino (https://www.makerguides.com/dht11-dht22-arduino-tutorial/)

(https://www.makerguides.com/maxbotix-mb7389-arduino-tutorial/)

MaxBotix MB7389 weather-resistant distance sensor tutorial (https://www.makerguides.com/maxbotix-mb7389-arduino-tutorial/)