LCD Display Screen

1. Introduction to the LCD Screen

Fig 1 1602A QAPASS 16×2 LCD display

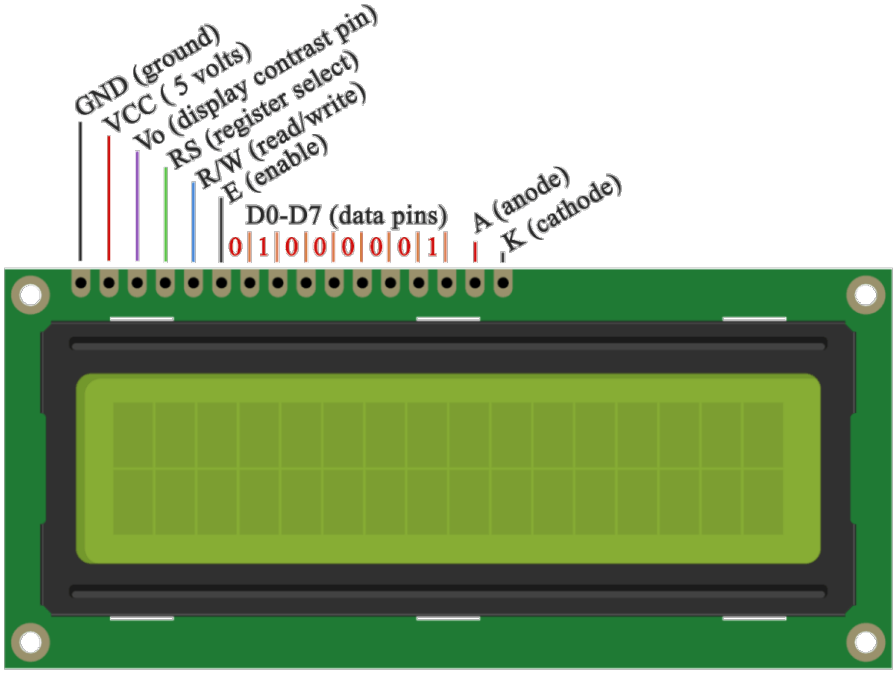


Fig 2 16×2 LCD display Pin Diagram

The LCD screen is made of 2 rows of 16 character spaces each. It is also built with 16 pins (16 channels) to connect to other devices and display the desired messages.

LiquidCrystal Library : This library allows an Arduino board to control Liquid Crystal displays (LCDs) based on the Hitachi HD44780 (or compatible) chipset, which is most popular, through a simple instruction set. A complete description of all the commands is available through the LiquidCrystal Library link shown above. The interface for the 1602A QAPASS consists of the following pins (the labels may be different for other 16x2 LCD displays but the functions will be the same):

- **Pin 1: VSS**, Ground (0V).
- **Pin 2: VDD**, power (+5V).
- **Pin 3: V0**, display contrast control through a potentiometer (0 to 5V).
- **Pin 4: RS**, register select that toggles between instruction and data register (0: instruction mode, and 1: data mode).
- **Pin 5: RW or R/W**, Read/Write (0: Write, 1: Read).
- **Pin 6: E**, Enable, active high to enable operations.
- **Pins 7-14: D0-D7**, in that order representing either data or instruction sequences.
- **Pins 15,16: A, K**, anode and cathode of the backlight LED, respectively.
- ***4-bit and 8-bit Mode of LCD:***

The LCD display can work in a 4-bit mode and an 8-bit mode.
4-bit mode: Only 4 bits of data are processed at a time. If the data is represented by a byte, then the high order nibble and then the low-order nibble will be handled to show the required character, by using shifting of data in the register. This more complicated process is desired only when the number of I/O pins needed from the Arduino Uno is not sufficient to handle an 8-bit sequence simultaneously.
8-bit mode: the entire 8-bit sequence is processed through the 8-bit data lines of the LCD display and 8 lines from the Arduino Uno. If another Arduino board with a lot more I/O ports is used, then the 8-bit mode would lead to simpler programming and faster execution.

Example 1.1 4-bit Mode String Display

Hardware Required

- Arduino Uno Board
- LCD Screen 1602A QAPASS or compatible
- 10k ohm potentiometer to control the contrast
- 10k ohm potentiometer to control the brightness of the backlight LED
- Connecting wires
- Breadboard

Circuit

The following table shows the connections between the LCD display and the Arduino board.

LCD Display		Arduino Uno
Pin #	Label	Label
1	VSS	GND
2	VDD	5V
3	VO	See notes below
4	RS	Digital Pin 11
5	R/W	GND
6	E	Digital Pin 12
11	DB4	Digital Pin 2
12	DB5	Digital Pin 3
13	DB6	Digital Pin 4
14	DB7	Digital Pin 5
15	Backlight +	See notes below
16	BackLight -	GND

Pin 3 of the LCD display is connected to the wiper of the 10kΩ potentiometer, used to control the contrast of the display.

Pin 15 of the LCD display is connected to the wiper of the 1kΩ potentiometer Used as a variable resistor to dim the backlight LED, also limiting its current.

Look at the sketch for additional connections related to the potentiometers.

You can zoom in the pictures for better reading of the connections.

Note that in a 4-bit mode, we have to use by default DB4-DB7 (high order nibble) of the LCD screen data pins. The string is represented by the ASCII code of the characters to be displayed. Most likely then, a character generator is included in the LCD electronics.

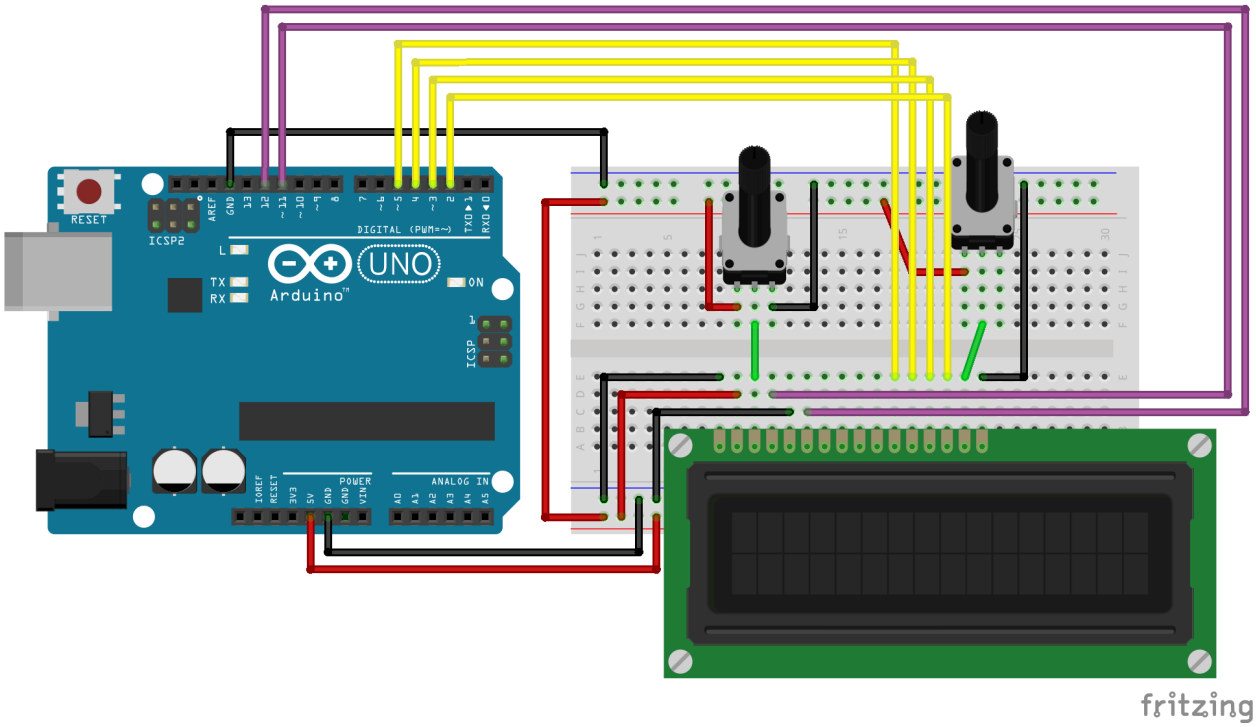


Fig 3 Fritzing Sketch of the Diagram

Schematic

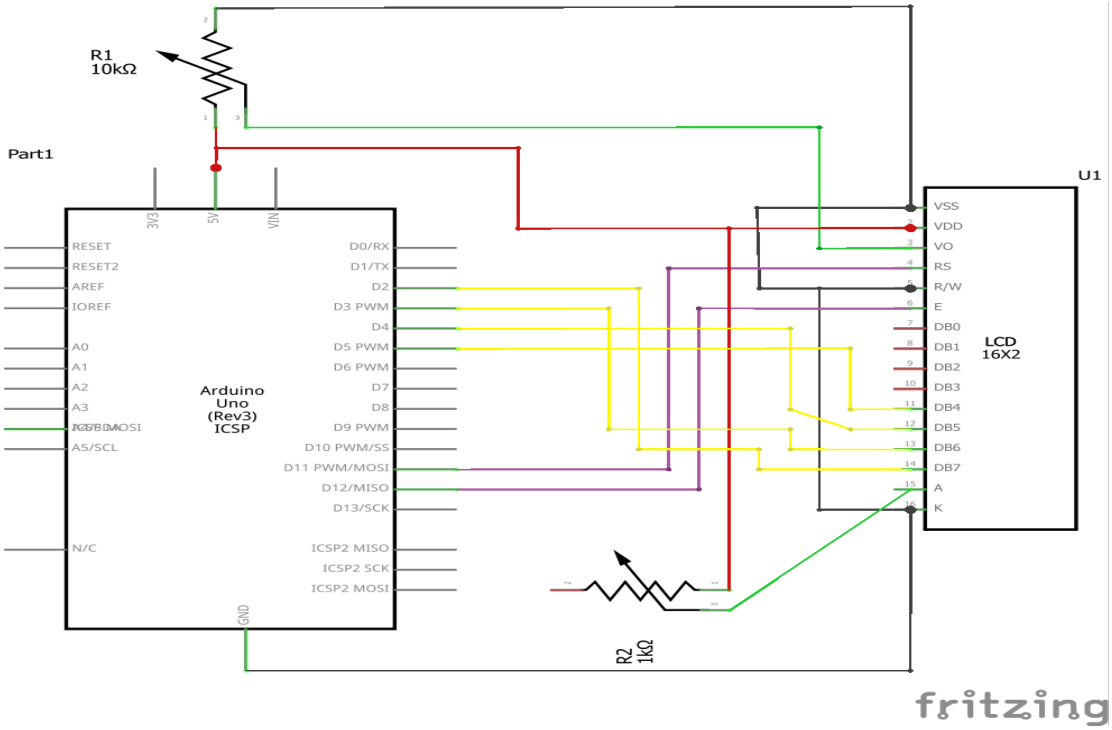


Fig 4 Circuit Diagram for Example 1.1

Code

```
*/
// include the library code:
#include <LiquidCrystal.h>
// initialize the LCD library based on the schematic pin connection
const int rs = 11, en = 12, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
// Specify the LCD's number of columns and rows. Change to (20, 4) for a 20x4 LCD:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  // The cursor is already at the beginning of the first row
  lcd.print("Arduino Tutorial");
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  lcd.print("ECE MF");
}

void loop() {
    // set the cursor to column 13, line 1
  // where column 13 is one space after ECE NCE NJIT
  // where line 1 is the second row, since counting begins with 0:
  lcd.setCursor(10, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

8-bit Mode Implementation

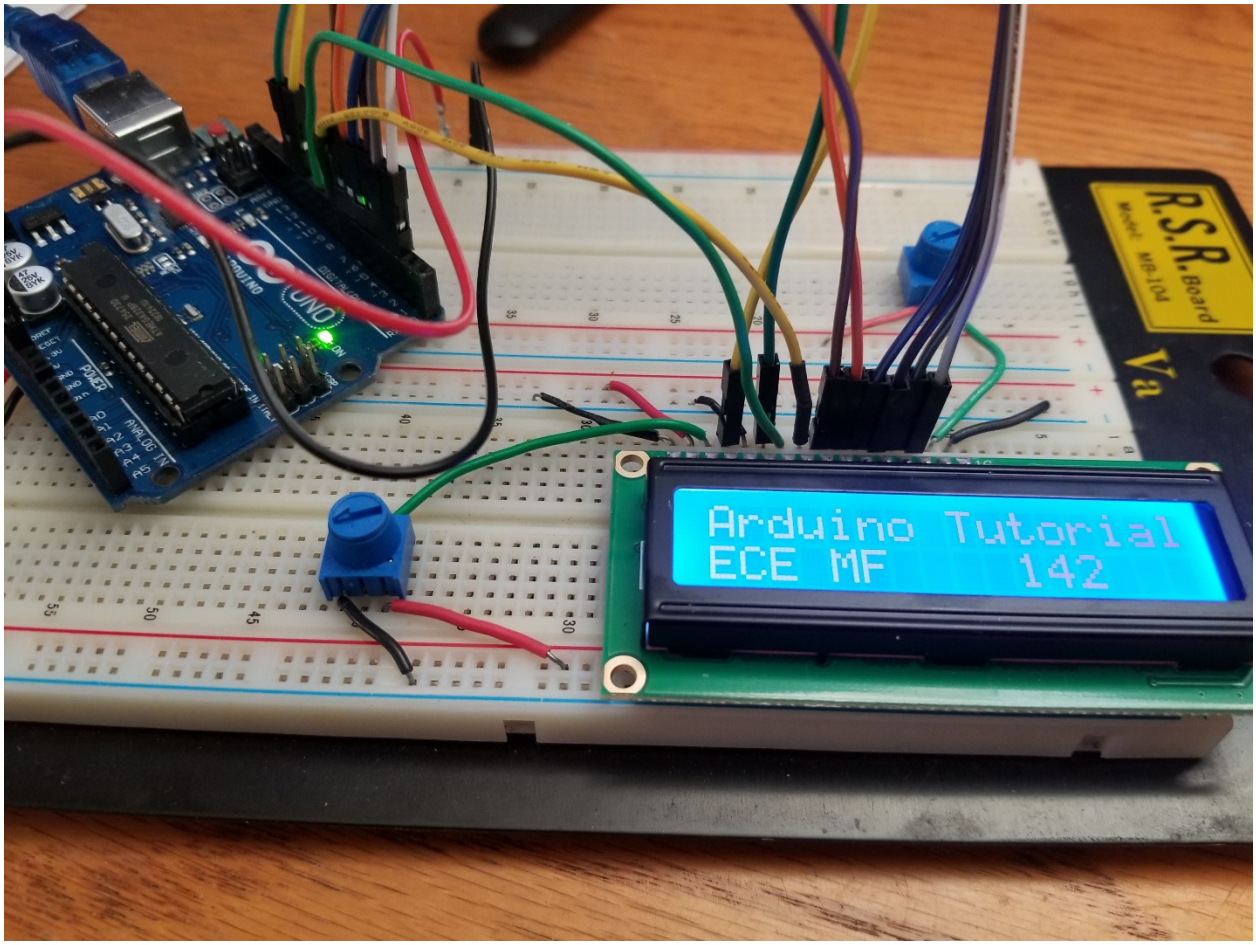


Fig 5 8-bit Mode Implementation for Example 1.1

The picture shown above represents the 8-bit implementation of the same example 1.1 that was presented previously. Four additional connections are required. Connect the display pins DB0 – DB3 to the Arduino digital pins 9, 8, 7, 6, respectively. In addition, change lines 5 and 6 from the code

```
const int rs = 11, en = 12, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

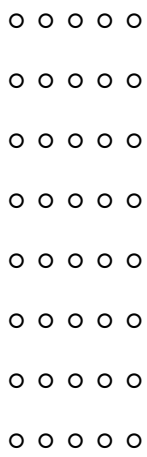
Into

```
const int rs = 11, en = 12, d0 = 9, d1 = 8, d2 = 7, d3 = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal lcd(rs, en, d0, d1, d2, d3, d4, d5, d6, d7);
```

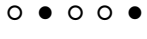
As we can see, whether we use 4 pins or 8 pins from the display, the system sets the mode accordingly. However, though we gain some processing time in dealing with a byte as a single sequence, the loss of utilization of 4 digital pins from the Arduino may not warrant this gain, especially when the application does not require high speed of execution as in this case.

2. Display of Special Characters

In this section, we are going to show how to control the pixels through data to display any character that you can design based on a 5x8 matrix of pixels. A 0 will turn off the pixel, and a 1 will turn it on. Each sequence of 5 bits will control one row of the character, from left to right, and from top to bottom.



If the first datum is B01001 (B stands for binary format), then the top row would look like



where the solid dot means the pixel is ON (2nd and 5th from the left).

Custom characters Arduino example code

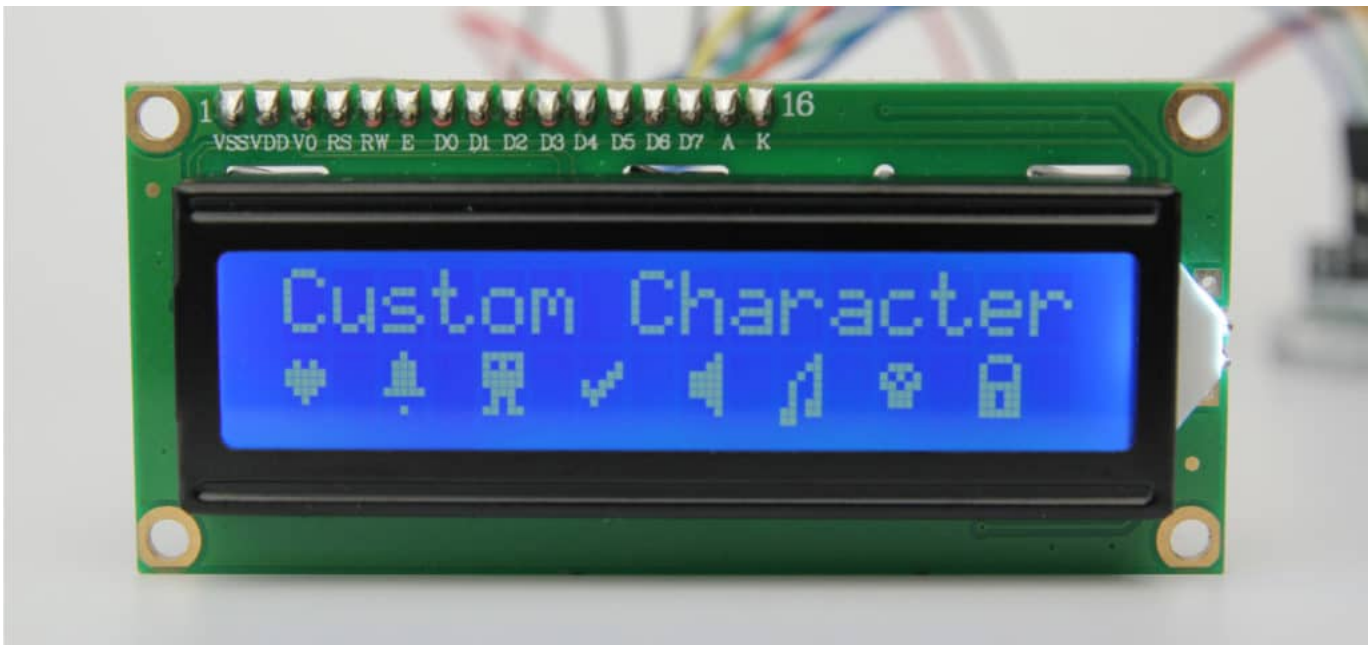
The following example sketch creates and displays eight custom characters (numbered 0 – 7).

```
/* 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 Skull[] = {B00000,B01110,B10101,B11011,B01110,B01110,B00000,B00000};
byte Crescent[] = {B00011,B01110,B11100,B11100,B11100,B11100,B01110,B00011};
// byte Sound[] = {B00001,B00011,B00101,B01001,B01001,B01011,B11011,B11000};
byte Star[] = {B00000,B00000,B00100,B11111,B01110,B01010,B10001,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, Skull);
lcd.createChar(5, Crescent);
// lcd.createChar(6, Sound);
lcd.createChar(6, Star);
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:
for (int i = 0; i <= 7; i++) {
   lcd.setCursor(2*i,1);
   lcd.write(byte(i));
```

```
  }
// 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:


1
16
VSSVDDV0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
Custom Character

Appendix

LiquidCrystal Library Function Set

LiquidCrystal()

Description

Creates a variable of type LiquidCrystal. The display can be controlled using 4 or 8 data lines. If the former, omit the pin numbers for d0 to d3 and leave those lines unconnected. The RW pin can be tied to ground instead of connected to a pin on the Arduino; if so, omit it from this function's parameters.

Syntax

LiquidCrystal(rs, enable, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)

Parameters

rs: the number of the Arduino pin that is connected to the RS pin on the LCD
rw: the number of the Arduino pin that is connected to the RW pin on the LCD (*optional*)
enable: the number of the Arduino pin that is connected to the enable pin on the LCD
d0, d1, d2, d3, d4, d5, d6, d7: the numbers of the Arduino pins that are connected to the corresponding data pins on the LCD. d0, d1, d2, and d3 are optional; if omitted, the LCD will be controlled using only the four data lines (d4, d5, d6, d7).

Example

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(11, 12, 5, 4, 3, 2);
void setup()
{
 lcd.begin(16,2);
 lcd.print("hello, world!");
}
void loop() {}
```

begin()

Description

Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display. `begin()` needs to be called before any other LCD library commands.

Syntax

lcd.begin(cols, rows)

Parameters

lcd: a variable of type LiquidCrystal
cols: the number of columns that the display has
rows: the number of rows that the display has

clear()

Description

Clears the LCD screen and positions the cursor in the upper-left corner.

Syntax

lcd.clear()

Parameters

lcd: a variable of type LiquidCrystal

home()

Description

Positions the cursor in the upper-left of the LCD. That is, use that location in outputting subsequent text to the display. To also clear the display, use the clear() function instead.

Syntax

lcd.home()

Parameters

lcd: a variable of type LiquidCrystal

setCursor()

Description

Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.

Syntax

lcd.setCursor(col, row)

Parameters

lcd: a variable of type LiquidCrystal
col: the column at which to position the cursor (with 0 being the first column)
row: the row at which to position the cursor (with 0 being the first row)

write()

Description

Write a character to the LCD. The example, given below, shows how a word typed through the keyboard, shown through the serial monitor, will end up being displayed on the LCD screen.

Syntax

lcd.write(data)

Parameters

lcd: a variable of type LiquidCrystal
data: the character to write to the display

Returns

byte
write() will return the number of bytes written, though reading that number is optional

Example

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(11, 12, 5, 4, 3, 2);
void setup()
```

```
{
 Serial.begin(9600);
}
void loop()
{
 if (Serial.available()) {
   lcd.write(Serial.read());
 }
}
```

print()

Description

Prints text to the LCD.

Syntax

lcd.print(data)
lcd.print(data, BASE)

Parameters

lcd: a variable of type LiquidCrystal
data: the data to print (char, byte, int, long, or string)
BASE (optional): the base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

Returns

byte
print() will return the number of bytes written, though reading that number is optional

Example

```
#include <LiquidCrystal.h>
// Creates an LCD object named print_clear just to show that the example is
// using the print and clear functions from the LiquidCrystal libray.
// Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal print_clear= LiquidCrystal(11, 12, 5, 4, 3, 2);

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

void loop() {
print_clear.clear();
print_clear.print("Friday");
delay(2000);
print_clear.clear();
print_clear.print("January 8, 2021");
delay(2000);
print_clear.clear();
print_clear.print("14:15");
```

```
delay(2000);
}
```

cursor()

Description

Display the LCD cursor: an underscore (line) at the position to which the next character will be written.

Syntax

lcd.cursor()

Parameters

lcd: a variable of type LiquidCrystal

noCursor()

Description

Hides the LCD cursor.

Syntax

lcd.noCursor()

Parameters

lcd: a variable of type LiquidCrystal

Example

cursor() and noCursor()

```
#include <LiquidCrystal.h>

// Creates an LCD object and then displays “cursor()” followed by

// a cursor appearing for one second every other second.

// Parameters: (RS, E, D4, D5, D6, D7)

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


void setup() {

curs_nocurs.begin(16, 2);

curs_nocurs.print ("cursor()");

}


void loop() {

curs_nocurs.cursor();

delay(1000);
```

```
curs_nocurs.noCursor();

delay(1000);

}
```

blink()

Description

Display the blinking LCD cursor. If used in combination with the cursor() function, the result will depend on the particular display.

Syntax

lcd.blink()

Parameters

lcd: a variable of type LiquidCrystal

Example

- blink() and noBlink()

noBlink()

Description

Turns off the blinking LCD cursor.

Syntax

lcd.noBlink()

Parameters

lcd: a variable of type LiquidCrystal

Example

- blink() and noBlink()

```
#include <LiquidCrystal.h>

// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)

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

void setup() {

lcd.begin(16, 2);

lcd.print("blink() example");

lcd.setCursor(0,1);

lcd.print("4s out of 7s");

}

void loop() {

lcd.blink();
```

```
delay(4000);

lcd.noBlink();

delay(3000);

}
```

display()

Description

Turns on the LCD display, after it's been turned off with noDisplay(). This will restore the text (and cursor) that was on the display.

Syntax

lcd.display()

Parameters

lcd: a variable of type LiquidCrystal

Example

noDisplay()

Description

Turns off the LCD display, without losing the text currently shown on it.

Syntax

lcd.noDisplay()

Parameters

lcd: a variable of type LiquidCrystal

Example

display() and noDisplay()

```
#include <LiquidCrystal.h>

// Creates an LCD object named disp_nodisp.

// The text will disappear every other second

// Parameters: (RS, E, D4, D5, D6, D7)

LiquidCrystal disp_nodisp = LiquidCrystal(11, 12, 5, 4, 3, 2);


void setup() {

disp_nodisp.begin(16, 2);

disp_nodisp.print("Blinking message");

}
```

```
void loop() {
disp_nodisp.display();
delay(1000);
disp_nodisp.noDisplay();
delay(1000);
}
```

scrollDisplayLeft()

Description

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

Syntax

lcd.scrollDisplayLeft()

Parameters

lcd: a variable of type LiquidCrystal

scrollDisplayRight()

Description

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

Syntax

lcd.scrollDisplayRight()

Parameters

lcd: a variable of type LiquidCrystal

Example

- scrollDisplayLeft() and scrollDisplayRight()

```
#include <LiquidCrystal.h>
// Creates an LCD object.
// The message will rotate left every 0.3s.
// The length of the message affects the display
// Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(11, 12, 5, 4, 3, 2);


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

```
}


void loop() {

lcd.scrollDisplayLeft();

delay(300);

// lcd.scrollDisplayRight();

// delay(300);

}
```

autoscroll()

Description

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.

Syntax

lcd.autoscroll()

Parameters

lcd: a variable of type LiquidCrystal

Example

```
#include <LiquidCrystal.h>

// Creates an LCD object.

// shows 0 to 15 while scrolling the display to the left

// prints the new character in the same location (right justified)

// beware of the length of the sequence

// Parameters: (RS, E, D4, D5, D6, D7)

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


void setup() {

lcd.begin(16, 2);

}
```

```
void loop() {

lcd.autoscroll();

lcd.setCursor(16, 0);

for (int x = 0; x < 16; x++) {

lcd.print(x);

delay(500);

}

lcd.clear();

}
```

noAutoscroll()

Description

Turns off automatic scrolling of the LCD.

Syntax

lcd.noAutoscroll()

Parameters

lcd: a variable of type LiquidCrystal

leftToRight()

Description

Set the direction for text written to the LCD to left-to-right, the default. This means that subsequent characters written to the display will go from left to right, but does not affect previously-output text.

Syntax

lcd.leftToRight()

Parameters

lcd: a variable of type LiquidCrystal

rightToLeft()

Description

Set the direction for text written to the LCD to right-to-left (the default is left-to-right). This means that subsequent characters written to the display will go from right to left, but does not affect previously-output text.

Syntax

lcd.rightToLeft()

Parameters

lcd: a variable of type LiquidCrystal

createChar()

Description

Create a custom character (glyph) for use on the LCD. Up to eight characters of 5x8 pixels are supported (numbered 0 to 7). The appearance of each custom character is specified by an array of eight bytes, one for each row. The five least significant bits of each byte determine the pixels in that row. To display a custom character on the screen, write() its number.

NB : When referencing custom character "0", if it is not in a variable, you need to cast it as a byte, otherwise the compiler throws an error. See the example below.

Syntax

lcd.createChar(num, data)

Parameters

lcd: a variable of type LiquidCrystal
num: which character to create (0 to 7)
data: the character's pixel data

Example

```
#include <LiquidCrystal.h>

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

byte smiley[8] = {
  B00000,
  B00000,
  B10001,
  B00000,
  B00000,
  B10001,
  B01110,
  B00000,
};
void setup() {
  lcd.createChar(0, smiley);
  lcd.begin(16, 2);
  lcd.write(byte(0));
}
void loop() {}
```