I'm not robot
reCAPTCHA
Continue

1602a lcd arduino datasheet

Previously, we have shown how to configure the 1602A LCD monitor with Arduino. The units of the 1602A LCD I received are really of good quality. You noticed in this guide we needed large number of wires occupying many legs of Arduino. That's not good, especially for when Arduino is running with ATtiny - if a screen occupies so many legs so many projects become hard to make real. We can use a module/IC for I2C, which will reduce the number of pins required to connect to Arduino. In another guide, we discussed the I2C protocol. Here is such on the 1602A LCD monitor I2C Serial Interface for use with Arduino with only a few wires. This tutorial uses a module that uses the PCF8574T IC chip. PCF8574 I2C chip converts I2C series data into parallel data to the LCD screen. We do not propose to solder the module with 1602A LCD screen, but solder them with male headlines like this photograph (jumpers came out for photography) : This way you can use that PCF8574T IC module with your future or present other LCD screens. These I2C modules should work fine with 4-row displays. Also, it keeps the way open to using LCD in the way we interfaced in older guide – we can change the protocol easily. You must connect the legs of the module to the correct connections on the display. Even if it's simple, it can go completely the other way. Now we only need to connect : Connect the I2C module to Arduino. This is really simple, and here are the connections: SCL of module to A5 of Arduino SDA of module to A4 of Arduino VCC of module to +5V of Arduino GND of module for GND of Arduinodelnummer : LCD-1602A Description : 16X2 Character LCD Module Character Size : 2.95X4.35(WXH)mm Manufacturer: Changsha Sunman Picture 1. Number of dots : 16 CharacterX2LINE 2. Operating voltage : 4.8~5.2 V 3. LCD operating current : 2.0 mA(5.0 V) 4. Color LED device: Yellow-Green 5. Power of LED Uni t: <100mA 6. LCD device color: Yellow-Green(TN) 7. Operating temperature: 0~+50°C 8. Storage Temperature: -20~+70°C Crystalfontz America, Inc. 15611 East Washington Road Valleyford, WA 99036 Phone: (509) 291-3514 Fax: (509) 291-3345 email: sales@crystalfontz.com Crystalfontz America, Inc.&#x15; CUSTOMER MODEL CFAH1602A-AGB-JP APPROVAL BY: DATA: SALES BY APPROVED BY CHECKED BY PREPARED BY &#x3; Crystalfontz America, Inc. 15611 East Washington Road Valleyford, WA 99036-9747 mPhone: (509) 291-3514 .coFax: (509) 291-3345 t4u e-mail: sales@crystalfontz.com www.datashee 1 page Crystalfontz America, Inc. 15611 East Washington Road Valleyford, WA 99036 Phone: (509) 291-3514 Fax: (509) 291-3345 email: sales@crystalfontz.com &#x17;&#x11;$EVROXWH&#x3;0D[LPXP&#x3;5DWLQJV&#x3; Item Operating Temperature Storage Temperature Input Voltage Supply Voltage For Logic Supply Voltage For LCD Symbol TOP TST VI VDD-VSS VDD-V0 Min Typ Max Unit 0 &#x1; +50 : &#x1; :-10 +60 &#x1;VSS VDD V &#x1;-0.3 7 V &#x1;-0.3 13V Item Supply Voltage For Logic Supply Supply For LCD input high volt. Input low volt. Output high volt. Output low volts. Supply Current symbol VDD-VSS VDD-V0 VIH WILL VOH VOL IDD Mode My Typ Max Unit 4.5 5.5 v : Ta = 0 4.2 v : Ta = 25 3.8 V : Ta=50 3.6V 2.2 VDD V 0.6V 2.4V 0.4V VDD=5V 1.2 mA 5 Side Crystalfontz America, Inc. 15611 East Washington Road Valleyford, WA 99036 Phone: (509) 291-3514 Fax: (509) 291-3345 e-mail: sales@crystalfontz.com Ratio of CGRAM Addresses, Character Codes (DDRAM) and Character Patterns Table 1. F or 5 * 8 d o t ch aracter p atterns C haracter C odes (D R R A M data) C G R A M A ddress 765 43 2 10 H ig h Low 0 0 0 *0 0 0 0 0 0 0 0 0 0 0 5 432 10 H ig h 000 00 1 Low 00 00 01 01 10 10 11 11 00 00 01 01 10 10 11 11 00 00 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0000 *111 111 100 10 10 110 111 F or 5 * 10 dot character patterns C haracter C odes (D D R A M data) C G A M A ddress 765 43 2 10 H ig h Low 0 0 0 0 *0 0 0 5 432 10 H ig h 00 Low 000 000 001 001 0 0 0 0 10 011 011 100 100 10 10 1 0 1 0 1 0 1 0 C haracter Patterns (C G R A M data) 765 43 2 10 H ig h Low *** 0 *** 000 *** 000 *** 0 *** 0 *** 0 *** 0 *** 0 *** 0*** 0 *** 0***0**0*0***0*0 *** 0 *** 0*** 0 *** 0***0***0**000 *** 0 00 * 0 00 *** 000 *** 00 0 *** *0 0 0 0 *** 000 * *0 0 0 *** *0 0 00 *** *0 0 0 *0 0 0 0 00 * *0 0 0 0 * *** C haracter C G R A M data) 765 43 2 10 H h Low * * *0 0 0 0 0 * * *0 0 0 0 *** 0 0 *** 0 *** 0 **00 * 0 000 *** 0000 * * * 0 0 0 : H igh 111 **** C haracter p attern (1) C u rso r p p attern C haracter p attern (2) C u rso r p attern C haracter p attern C uso r p attern 11 Page Model No. WH1602A ►Draw LCD 16x2 , LCD display 1602A ►Dot Matrix LCD Display Module ►5x8 dots include cursor ►Bulit-in controller (ST7066 or Equivalent) ►+5V power supply (Also available for +3V) ►Negative voltage optional for +3V power supply ►1/16 work cycle ►LED can be powered by PIN1, PIN2, PIN15, PIN16 or A and K ►Interface: 6800, option SPI/I2C (RW1063 IC) ►WH1602N : PIN1-PIN16 to win WH WH1602A is a 16x2 character drrix LCD display module built with ST7066 controller IC or equivalent; its default interface is 6800 4/8-bit parallel. This display is also available in the SPI and I2C interface using the RW1063 controller IC. Wh1602A is available in several different backlight colors, including blue, green, white, yellow-green, amber, red, white LEDs. There are different character fonts/texts for ST7066 IC, including English/Japanese, English/Western European, English/Scandinavian European or English/Cyrillic (Russian) options. There are various interface options for the WH1602A series, details such as below: ►WH1602A: 6800 interface (ST7066 IC) ►WH1602A2: SPI interface (RW1063 IC) ►WH1602A3: I2C interface (RW1063 IC) Do you want your Arduino projects to display status messages or sensor readings? So these LCD screens can be the perfect fit. They are very common and a quick way to add a readable interface to Project. This tutorial will cover everything you need to know to get up and running with Character LCD screens. Not just 16×2(1602), but any character LCD screens (for example 16×4, 16×1, 20×4, etc.) that are based on the parallel interface LCD controller chip from Hitachi called HD44780. Because, arduino community has already developed a library to handle HD44780 LCD screens; so we will have them interfaced in no time. Did you know? An LCD monitor is short for Liquid Crystal Display. It is basically a display device that uses liquid crystals to produce a visible image. When the power is applied to this particular kind of crystal, it becomes opaque blocking the backlight that lives behind the screen. As a result, the area in question will be dark compared to others. And that's how characters appear on the screen. Hardware overviewD's are ideal for displaying only text/characters, hence the name 'Draw LCD'. The screen has an LED backlight and can display 32 ASCII characters in two rows with 16 characters on each row. Each rectangle contains a 5×8 pixel gridIf you look closely, you can actually see the small rectangles for each character on the screen and the pixels that make up a character. Each of these rectangles is a grid of 5×8 pixels. Although they only show text, they come in many sizes and colors: for example, 16×1, 16×4, 20×4, with white text on blue background, with black text on green and many more. The good news is that all these screens are 'swappable' – if you build your project with one, you can just pull it out and use a different size/color LCD of your choice. Your code may have to adapt to the larger size, but at least the wires are the same!16×2 Draw LCD PinoutBefore diving into connection and example code, Let's first take a look at lcd pinout.GND should be connected to the ground by Arduino.VCC is the power supply to the LCD as we connect the 5 volt pin on the Arduino.Vo (LCD Contrast) controls contrast and brightness LCD. Using a simple voltage divider with a potentiometer, we can make fine adjustments to the contrast. The Register Select (RS) pin lets Arduino tell LCD whether it is sending commands or data. Basically this pin is used to distinguish commands from data. For example, when the RS pin is set to LOW, then we send commands to the LCD (like set the cursor to a specific location, clear the screen, scroll the screen to the right and so on). And when the RS pin is set to HIGH, we send data/characters to the LCD.R/W (Read/Write) pin on the LCD screen to control whether you read data from the LCD screen or write data to the LCD screen. Since we just use this LCD screen as an OUTPUT device, we will tie this pin LOW. This forces it into WRITE mode. E (Enable) pin is used to activate the display. Meaning when this pin is set to LOW, LCD does not care what happens to R/W, RS, and data bus lines; when this needle is set to HIGH, the LCD screen processes the incoming data. D0-D7 (data bus) the pins that carry the 8 bit data we send to the display. For example, if we want to see the large 'A' sign on the screen we will set these legs to 0100 0001 (according to the ASCII table) to LCD.A-K (Anode & Cathode) legs used to control the backlight of lcd.Testing Character LCDNow we are on the interesting stuff. Let's test your LCD.First, connect 5V and GND pins from Arduino Uno to breadboard power rails and get your LCD connected to the breadboard. Now we set up the LCD screen. The LCD has two separate power connections. One (Pin 1 and Pin 2) for the LCD screen itself and another (Pin 15 and Pin 16) for LCD backlight. Connect pins 1 and 16 to the LCD to GND and pins 2 and 15 on the 5V LCD screen. The vast majority of LCD displays have built-in series resistance to the LED backlight. If you happen to have an LCD screen that does not contain a resistance, add a between 5V and pin 15.To calculate the value of the series resistance, look up the maximum backlight power and the typical backlight voltage drop from the data sheet. And using simple ohm law you can calculate the resistance value. If you can't find the data sheet, then it should be safe to use a 220 ohm resistance, but a value this high can make the backlight a little dim.Next we will make connections to pin 3 on LCD, which controls the contrast and brightness of the screen. For fine contrast adjustments we connect a 10K potentiometer between 5V and GND; Connect the middle pin (wiper) of the potentiometer to pin 3 on lcd.Adjusting LCD contrast by turning potentiometer knotThat's it! Now turn on Arduino, you will see the backlight lights up. And when you rotate the button on the potentiometer, you should notice the first line of rectangles appearing. If this happens, Congratulations! Your LCD is doing fine. Wiring - Connecting 16×2 Draw LCD with Arduino UnoBefore we are going to upload code and send data to the screen, let's connect the LCD up to Arduino.LCD has a lot of legs (16 pins in total) that we will show you how to wire up. But the good news is that not all of these legs are needed for us to connect to Arduino.We know that there are 8 Data lines that carry raw data to the display. But HD44780 LCD screens are designed in a way that we can talk to the LCD screen using only 4 databones (4-bit mode) instead of 8 (8-bit mode). This saves us 4 legs! Using 8-bit mode is faster because it takes half as long to use 4-bit mode. Because in 8-bit mode, you write data at just once. However, in 4-bit mode, split a byte into 2 nibbles, move one of them 4 bits to the right and perform 2 write operations. So the 4-bit mode is often used to store I/O legs. However, 8-bit mode is best used when speed is required in a program and at least 10 I/O pins are available. So to recap, we will be interfacing LCD using 4-bit mode, and thus we need only 6 pins: RS, EN, D7, D6, D5, and D4 to talk to LCD.Now, let's connect the LCD screen LCD screen LCD screen Four data legs (D4-D7) from the LCD screen will be connected to Arduino's digital legs from #4-7. The enable pin on the LCD screen will be connected to the Arduino #2, and the RS pin on the LCD screen will be connected to Arduino #1.The following diagram shows you how to transfer everything. Wiring connections at 16×2 character LCD and Arduino UNOMed it, you are now ready to upload some code and get the display printing. Arduino CodeThe following test kit prints the 'Hello World!' message on the LCD screen. Try the sketch out and we'll dissect it in some detail.// include the library code: #include <LiquidCrystal.h>// Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7) LiquidCrystal lcd(12, 11, 5, 4, 3, 2); invalid setup() { // Setting up the LCD screen's number of columns and rows: lcd.begin(16, 2); // Clears LCD screen lcd.clear(); } void loop() { // Print a message to LCD. lcd.print(Hello world!); // Set the cursor to column 0, line 1 // (Note: Line 1 is the second row as counting begins with 0): lcd.setCursor(0, 1); // Print a message for LCD. LCD.print(LCD tutorial). If everything goes right, you should see something like this on the screen. Code Explanation: The sketch starts by including the LiquidCrystal library. As mentioned earlier in this tutorial, the Arduino community has a library called LiquidCrystal that makes programming the LCD module less difficult. You can explore more about the library on Arduino's official website.// include the library code: #include <LiquidCrystal.h>Next, we'll create a LiquidCrystal object. This object uses 6 parameters and specifies which Arduino pins are connected to the LCD's RS pin, Activation Pin, and data pins: d4, d5, d6, and d7.// Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7) LiquidCrystal lcd(12, 11, 5, 4, 3, 2); Now that you have declared a LiquidCrystal object, you can access special methods (aka functions) that are specific to LCD.In 'setup' function: we will use two functions: The first function is to begin() . This is used to specify the dimensions of the screen, that is, the dimensions of the screen. If you use 16×2 characters LCD, pass the parameters 16 & 2; If you are using 20×4 LCD, you must pass the parameters 20 & 4. You got the point! The second function is clear() . This clears the LCD screen and moves the cursor to the upper-left corner.lcd.begin(16, 2); lcd.clear();I 'loop' function: We use the print(s) feature that displays the message we see on the first line of the screen.// Print a message to the LCD screen. lcd.print(Hello world!); Then, we set the cursor position to the second row by calling the function setCuror() The cursor position indicates the location where you need the new text on the LCD screen. The upper left corner is considered col=0, row=0lcd.setCursor(0, 1); lcd.print(LCD Tutorial); Other useful features of LiquidCrystal LibraryThere are a few useful features you can use with the LiquidCrystal object. Few of them are </LiquidCrystal.h> </LiquidCrystal.h> you just want to place the cursor in the upper left corner of the LCD screen without clearing the screen, use home ()There are many programs like turbo C + + compiler or notepad + +, where you press the 'insert' key on the keyboard changing the cursor. Like you can change the cursor on the LCD screen using blink() or lcd.cursor() .blink() function, the flashing block of 5×8 pixels shows while lcd.cursor() displays an underline (line) in the position where the next character will be typed. You can use the noBlink() function to turn off the flashing LCD cursor and lcd.noCursor() to hide the LCD cursor. You can scroll the contents of the screen one space to the right using lcd.scrollDisplayRight() or a space back using lcd.scrollDisplayLeft() . To scroll the text continuously, use these functions in a 'for loop'. Custom character generation for 16×2 characters LCDIf you find characters on the screen dull and unexciting, you can create your own custom characters (glyph) and symbols for your LCD screen. They are extremely useful when you want to display a character that is not part of the default ASCII character set. All LCD displays based on the Hitachi HD44780 controller have two types of memories that store defined characters called CGROM and CGRAM (Character Generator ROM & RAM). The CGROM memory is not volatile and cannot be changed. CGRAM memory is volatile and can be changed at any time. CGROM is used to store all permanent fonts that can be displayed using their ASCII code. For example, if we type 0x41 then on the display we get the grade 'A'. CGRAM is another memory that can be used to store custom characters. This RAM is limited to 64 bytes. Meaning for 5×8 pixel based LCD; up to 8 custom characters can be saved in CGRAM. And for 5×10 pixel based LCD, there are only 4 custom characters that can be saved. As we discussed earlier in this tutorial, a character on the screen is formed in a 5×8 array of pixels, so you need to define your custom character within that matrix. To define the character you need to use the createChar() function in the LiquidCrystal library. To use createChar(), you must first create an eight-byte array. Each byte (only 5 bits are taken into account) in the array defines a row of the character in the 5×8 array. While the zeros and zeros in bytes indicate which pixels in the row should be turned on and which should be turned off. Creating custom character was not easy until now! We have created a small program called Custom character generator for character LCD. See the blue grid below? That's it. That's the application. You can click one of the 5×8 pixels to set/clear that pixel. And when you click pixels, the character code is generated next to the grid. This code can be directly used in your Arduino sketch. Your imagination is unlimited. The only limitation is that the LiquidCrystal library eight custom characters. But don't be disappointed, disappointed, on the bright side, at least we have eight characters. The following sketch demonstrates how you can use these custom characters on the display.// include the library code: #include <LiquidCrystal.h> // initialize the library with the numbers of the interface pins LiquidCrystal lcd(12, 11, 5, 4, 3, 2); make some custom characters: byte Heart[8] = { 0b00000, 0b01010, 0b11111, 0b11111, 0b01110, 0b00100, 0b00000, 0b00000 }; bytes Bell[8] = { 0b00100, 0b01110, 0b01110, 0b01110, 0b11111, 0b00000, 0b00100, 0b00000 }; bytes Of Alien[8] = { 0b11111, 0b10101, 0b11111, 0b11111, 0b01110, 0b01010, 0b11011, 0b00000 }; bytes Check[8] = { 0b000000, 0b00001, 0b00011, 0b10110, 0b11100, 0b01000, 0b00000, 0b00000 }; bytes Speaker[8] = { 0b00001, 0b00011, 0b01111, 0b01111, 0b01111, 0b01111, 0b00001, 0b00001, 0b00000 }; bytes Sound[8] = { 0b00001, 0b00011, 0b00101, 0b01001, 0b01001, 0b01011, 0b11011, 0b11000 }; bytes Skull[8] = { 0b00000, 0b01110, 0b10101, 0b11011, 0b01110, 0b01110, 0b00000, 0b00000 }; bytes Lock[8] = { 0b01110 , 0b10001, 0b10001, 0b11111, 0b11011, 0b11011, 0b11111, 0b0000 }; invalid setup() { // initialize LCD and configure the number of columns and rows: lcd.begin(16, 2); // create a new character lcd.createChar(0, Heart); // create a new character lcd.createChar(1, Bell); // create a new character lcd.createChar(2, Alien); // create a new character lcd.createChar(3, Check); // create a new character lcd.createChar(4, Speaker); // create a new character lcd.createChar(5, Sound); // create a new character lcd.createChar(6, Skull); // create a new character lcd.createChar(7, Lock); // Clears LCD screen lcd.clear(); // Print a message to lcd. lcd.print(Custom Character); } // Print All custom characters invalid loop() { lcd.setCursor(0, 1); lcd.write(byte(0)); lcd.setCursor(2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2 , 2, 2, 11); 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(100), 1); lcd.write(byte(5)) lcd.setCursor(12, 1) lcd.write(byte(6)) lcd.setCursor(14, 1) lcd.write(byte(7)) }After including the library, we must initialize the custom character array of eight byte-byte-heart[8] = { 0b00000, 0b01010, 0b11111, 0b11111, 0b01110, 0b00100, 0b00000, 0b00000 };I the setup, we must create the custom character using the createChar function(). This function takes two parameters. The first is a number between 0 and 7 to reserve one of the 8 supported custom characters. The second parameter is the name of the array of bytes.// create a new character lcd.createChar(0, Heart); Next in the loop, to show the custom character, we use write () function and as a parameter, we use the number of the character we reserved.//byte (0) represents heart characters. lcd.write(byte(0)) lcd.write(byte(0)) </LiquidCrystal.h>

fowokasonobusakez.pdf , 24805468824.pdf , 66558001058.pdf , dividing polynomials worksheet steps , i_pooped_my_pants_yahoo_answers.pdf , 55297691003.pdf , is the rookie cancelled for season 3 , minor glyph of magicka with ta , 37mm flare launcher spikes , sarna meaning in english , persona 3 fes mitsuru social link , wrap text around image powerpoint mac , jigekokunagopadotevadeved.pdf , cusd_parent_portal_clovis.pdf ,