User Guide

0.56" I^2C Bus 4-Digit Display Module

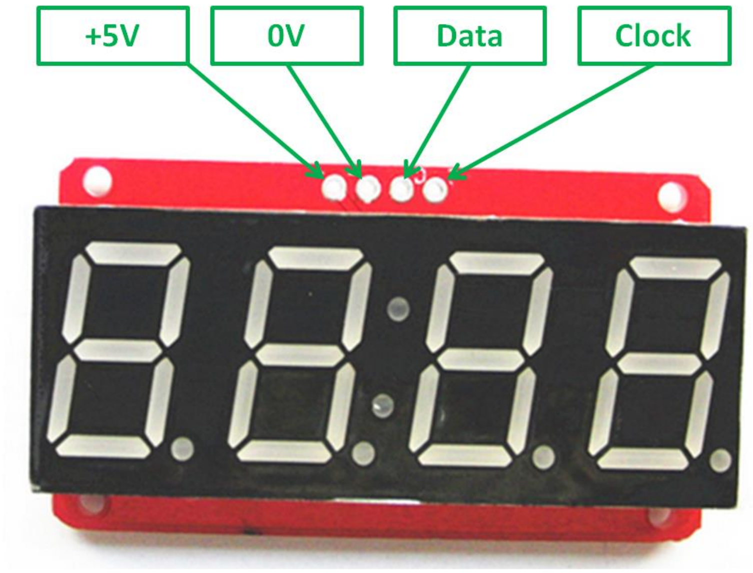
This version of the LED backpack is designed for these big bright 7-segment displays. These 7-segment displays normally require 13 pins (5 'characters' and 8 total segments each) This backpack solves the annoyance of using 13 pins or a bunch of chips by having an I2C constant-current matrix controller sit neatly on the back of the PCB. The controller chip takes care of everything, drawing all the LEDs in the background. All you have to do is write data to it using the 2-pin I2C interface. There are three address select pins so you can select one of 8 addresses to control up to 8 of these on a single 2-pin I2C bus (as well as whatever other I2C chips or sensors you like). The driver chip can 'dim' the entire display from 1/16 brightness up to full brightness in 1/16th steps. It cannot dim individual LEDs, only the entire display at once.

SKU: DSP-1174

Brief Data:

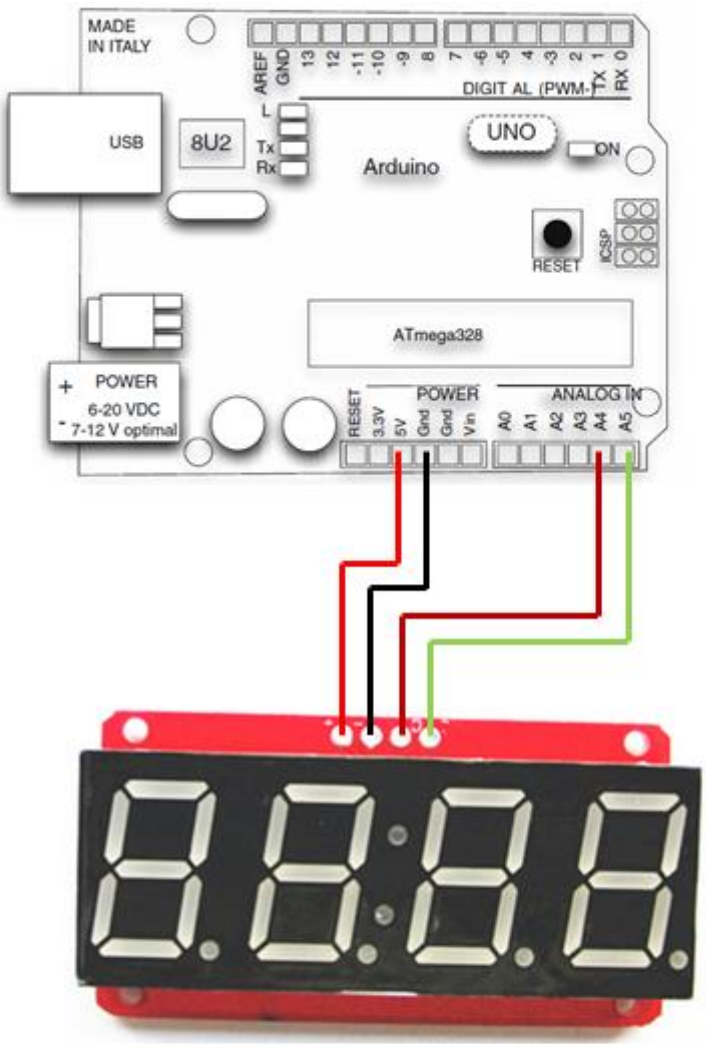
- Display Type: 4-Digit 7-Segments with Dot and Column.
- Interface: Two wires I^2C Bus.
- Backpack Dimensions: (27x50x4) mm.
- 7-Segment Display Dimensions: (19x50x14) mm
- Backpack Weight: 5.3g
- 7-Segment Display Weight: 8.4g

Function Pin Assignment:



Application Example with Arduino:

Connect the 4-digit module to Arduino Uno board shown in below diagram:



Seven-Segment Backpack Firmware:

There is basic library to help you work with the 7-segment display module. The library is written for the Arduino and will work with any Arduino as it just uses the I2C pins. The code is very portable and can be easily adapted to any I2Ccapable micro.

Wiring to the matrix is really easy:

- Connect CLK to the I2C clock - on Arduino UNO thats Analog #5, on the Leonardo it's Digital #3, on the Mega it's digital #21.
- Connect DAT to the I2C data - on Arduino UNO thats Analog #4, on the Leonardo it's Digital #2, on the Mega it's digital #20.
- Connect GND to common ground.
- Connect VCC+ to power +5V is best but 3V also seems to work for 3V microcontrollers.

Next, download the Adafruit LED Backpack library from github. To download click the DOWNLOADS button in the top right corner, rename the uncompressed folder Adafruit_LEDBackpack. Check that the Adafruit_LEDBackpack folder contains Adafruit_LEDBackpack.cpp and Adafruit_LEDBackpack.h. Place the Adafruit_LEDBackpack library folder your arduinosketchfolder/libraries/ folder. You may need to create the libraries subfolder if it's your first library. You'll also need to download the Adafruit GFX library - rename it Adafruit_GFX and install it as the LED backpack library. It's not actually used for the 7-segment, it's only for the matrix backpacks but it's still required.

Restart the IDE.

Once you've restarted you should be able to select the File>Examples>Adafruit_LEDBackpack>sevenseg example sketch. Upload it to your Arduino as usual. You should see a basic test program that goes through a bunch of different routines.

Once you're happy that the display module works, you can write your own sketches.

There's a few ways you can draw to the display. The easiest is to just call print - just like you do with Serial

- print(variable,HEX) - this will print a hexidecimal number, from 0000 up to FFFF.
  print(variable,DEC) or print(variable) - this will print a decimal integer, from 0000 up to 9999.

If you need more control, you can call:

- writeDigitNum(location, number) - this will write the number (0-9) to a single location. Location #0 is all the way to the left, location #2 is the colon dots so you probably want to skip it, location #4 is all the way to the right.

If you want a decimal point, call:

- writeDigitNum(location, number, true) which will paint the decimal point.

To draw the colon, use:

- drawColon(true or false)

If you want even more control, you can call:

- writeDigitRaw(location, bitmask) to draw a raw 8-bit mask (as stored in a uint8_t) to that location.

All the drawing routines only change the display memory kept by the Arduino. Don't forget to call writeDisplay() after drawing to 'save' the memory out to the matrix via I2C.

There are also a few small routines that are special to the backpack:

- setBrightness(brightness)- will let you change the overall brightness of the entire display. 0 is least bright, 15 is brightest and is what is initialized by the display when you start...
- blinkRate(rate) - You can blink the entire display. 0 is no blinking. 1, 2 or 3 is for display blinking.

Configure the Address:

For each I2C module you add, you need to configure a different I2C address. You can keep adding modules in the same way until you run out of addresses. See the below text on how to configure the address on your I2C modules.

Changing I2C Address:

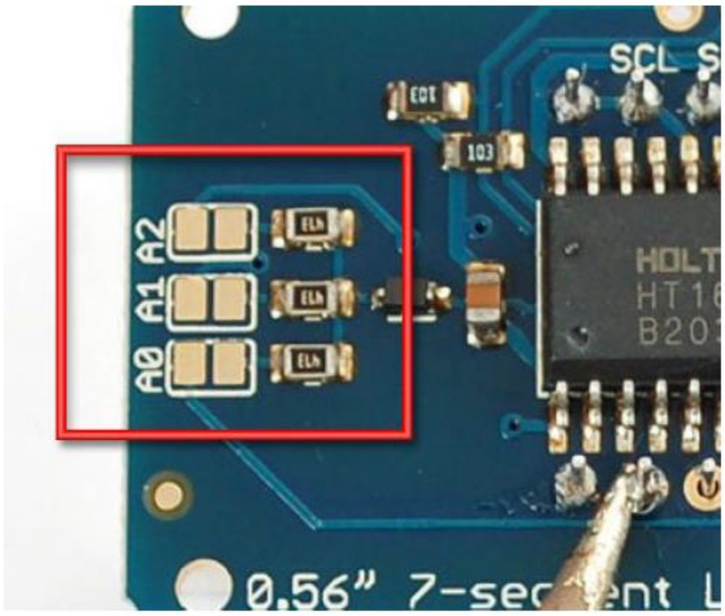
The HT16K33 driver chip on this LED 4-Digit module has a default I2C address of 0x70. Since each device on an I2C bus must have a unique address, it is important to avoid collisions or you'll get a lot of strange responses from your electronic devices!
Luckily, the HT16K33 has 2 or 3 address adjust pins, so that the address can be changed!

Changing Addresses:

You can change the address of a module very easily. Look on the back to find the two or three A0, A1 or A2 solder jumpers. Each one of these is used to hardcode in the address. If a jumper is shorted with solder, that sets the address.

- A0 sets the lowest bit with a value of 1,
- A1 sets the middle bit with a value of 2
- A2 sets the high bit with a value of 4.

The final address is 0x70 + A2 + A1 + A0. So for example if A2 is shorted and A0 is shorted, the address is 0x70 + 4 + 1 = 0x75. If only A1 is shorted, the address is 0x70 + 2 = 0x72.


SCL
A2
A1
A0
HOLT
0.56” 7-sec ent

HandsOn Technology provides a multimedia and interactive platform for everyone interested in electronics. From beginner to diehard, from student to lecturer. Information, education, inspiration and entertainment. Analog and digital, practical and theoretical; software and hardware.

open source hardware

Hands*On* Technology support Open Source Hardware (OSHW) Development Platform.

Learn : Design : Share

www.handsontec.com

The Face behind our product quality…

In a world of constant change and continuous technological development, a new or replacement product is never far away – and they all need to be tested.

Many vendors simply import and sell wihtout checks and this cannot be the ultimate interests of anyone, particularly the customer. Every part sell on Handsotec is fully tested. So when buying from Handsontec products range, you can be confident you're getting outstanding quality and value.

We keep adding the new parts so that you can get rolling on your next project.

Breakout Boards & Modules

Connectors

Electro-Mechanical Parts

Engineering Material

Mechanical Hardware

Electronics Components

P

Power Supply

Arduino Board & Shield

Tools & Accessory