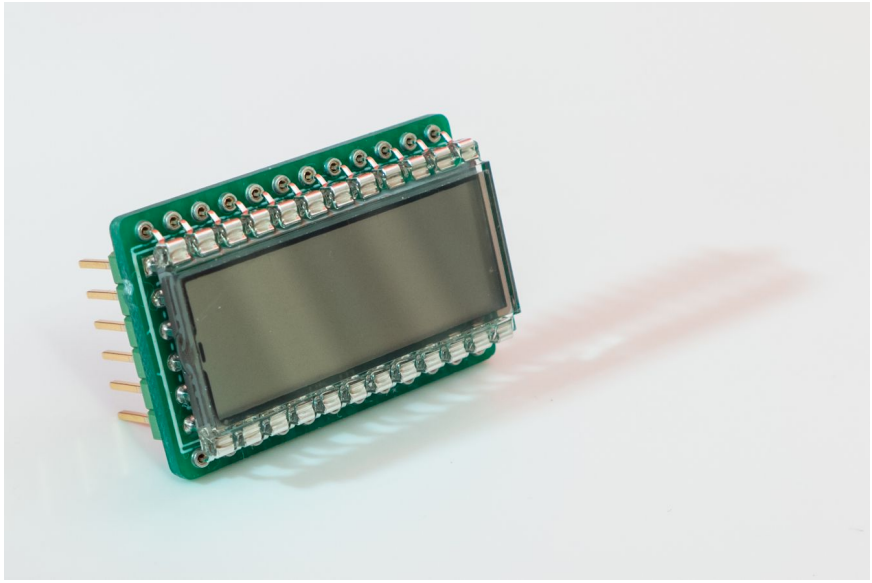
Building a Low Power Compact LCD Display

Version 10

Created by shabaz on Apr 22, 2016 11:41 AM. Last modified by summerella on Jul 21, 2016 4:27 PM.

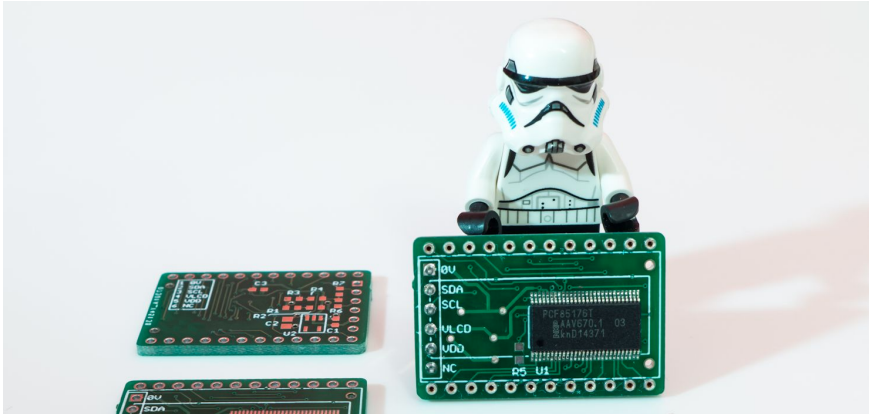


Introduction

Sometimes a really simple, low power display is needed. A lot of information can be shown on even a 7-segment display!

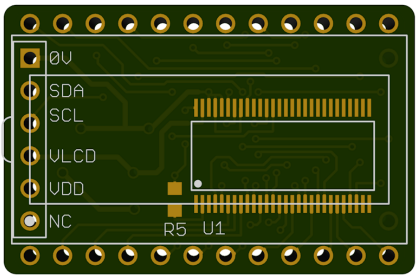
This LCD board project could be useful for such purposes. The three 7-segment digits (and two decimal places total, between the digits) can show percentages, hexadecimal values, voltages and current readings. Plenty of characters can be represented too.

Up to four of these displays can be connected to a single I2C bus so (for example) a power supply project could use one display for voltage and another for current, for two channels. To save costs for smaller projects a single display could be used to show more than one item of information using a button press or periodic cycling of data on the display.

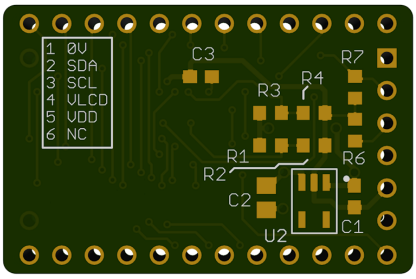


The reflective LCD that is used has no backlight but has very good contrast and in tests had excellent visibility even in dim light. The power consumption is extremely low and this makes it ideal for portable projects which might need to run for years on batteries without charging. The entire bill-of-materials is less than £5GBP (or lower if multiple displays are purchased).
The schematic and plot (gerber) files and source code is attached to this

post. I had 20 PCBs made for less than the cost of a pizza.
As a quick test I tried it using a Raspberry Pi just counting up from 0.0 to 99.9. With very little effort it would be possible to use it to display Pi stats – for example temperature, or the IP address (one byte at a time) upon start-up. It is easy to get any script to write to the display from the command line.



Display Side



Rear Side

LCD 3-Digit Project
I2C Controlled 3-Digit LCD

Features Summary

- 3.3V or 5V capable
- Ultra-low current consumption (24uA at 3.3V)
- Extremely compact; 32x21mm
- Just a 4 wire connection; I2C Interface, 0V and VDD (3.3V or 5V)
- Individually addressable segments for alphanumeric and other symbol use
- Quick to assemble; less than a dozen parts
- Low cost; about £5GBP in quantities of 1

Parts List

Ref	Code	Mnfr Part #	Description	Qty	GBP Each
LCD1	1838930	LCD-S301C31TR	LCD 3-digit	1	1.85
U1	2400460	PCF85176T	LCD Driver	1	1.18
C3	1759122	MC0603B104K500CT	100N 0603 x7r	1	0.01
R1, R3	2073349	MCMR06X1002FTL	10K Res 0603	2	0.01
R2, R4, R5			D.N.F.	0	0.00
R6, R7	9331972	MC0063W060352K7	2.7k Res 0603	2	0.01

Ref	Code	Mnfr Part #	Description	Qty	GBP Each
J1	1593415	2211S-06G	SIL header 6-way	1	0.26
Optional	177848	D01-99012 01	PCB pin sockets 12-way	2	1.18
Optional	2299856	0552-2-15-01-21-27-10-0	PCB pin sockets (alternative)	24	0.07

Additional components for 5V operation

Ref	Code	Mnfr Part #	Description	Qty	GBP Each
U2	2314372	TPS70925DBVT	LDO 2.5V Voltage Regulator	1	0.82
C1	1907343	C1608X7R1C105K080AC	1u 0603 X7R	1	0.04
C2	1735542	GRM21BR71E225KA73L	2.2u 0805 X7R	1	0.24

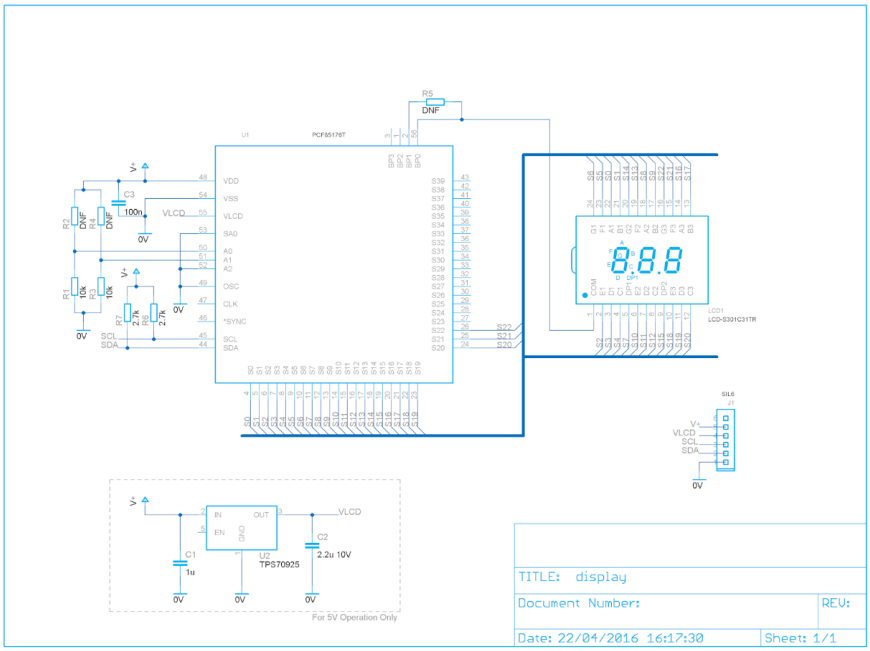
How it Works

There are different types of LCDs, but in general they require an alternating current (AC) signal to turn on segments. Some use a square wave, others require multiple voltage levels in order to achieve high segment density with low pin count through multiplexing.
The particular LCD chosen requires a square wave, and has the benefit of providing high contrast digits. Reflective displays offer high contrast but have the disadvantage that they are opaque from the underside and cannot be lit from behind so require either ambient light to be visible or a side light.

Some microcontrollers feature LCD driving capability but there is a classic integrated circuit from NXP that is useful to offload the task from microcontrollers and allow the use of an I2C bus for interfacing. That IC is the PCF8576 which has been around for more than two decades. NXP have a more recent IC called PCF85176 (it appears to be backwards-compatible) and that is the one used for this project because it has very low power consumption requirements and a higher speed I2C bus capability than its predecessor.
Nothing else is needed apart from a few resistors and capacitors.

The LCD needs a few volts peak voltage in order to function. For 5V operation a linear regulator provides the lower voltage to run the LCD.

Building It



Order the boards from a PCB firm (the files are attached to this post). Once the PCBs arrive, the first step is to solder the PCF85176 chip. It is quite small (0.5mm spaced pins) but hand-soldering is still possible. A hot plate and solder paste would be another option. I just used a normal soldering iron (2mm tip) and no-clean flux (cleaned up afterwards) and thin solder.

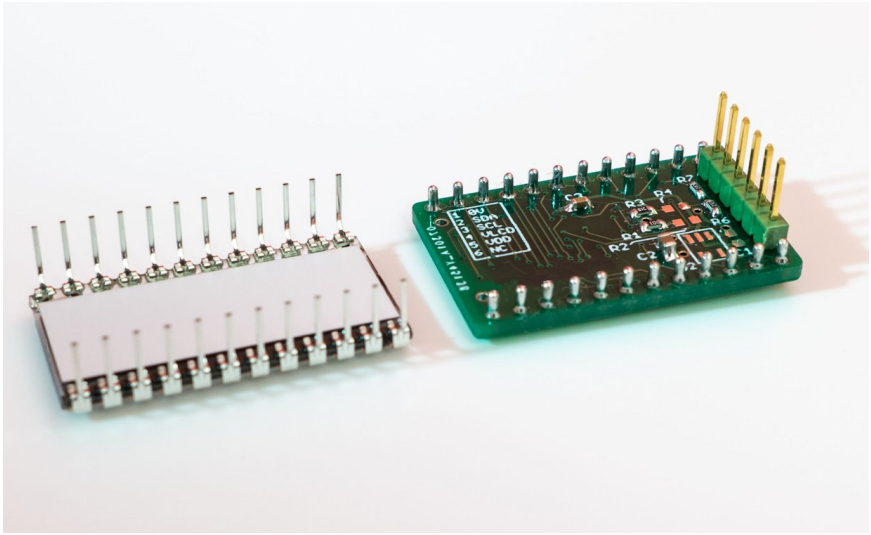
The hardest part is perfect alignment. The way I do it is to tape the IC onto the PCB first. Then, inspect the pins using a magnifying lens. If it looks incorrect, peel the tape from one end but keep the IC attached to the tape. Move the tape and re-stick it onto the PCB. It becomes quite easy to align the IC and secure it in this manner, just moving the tape from one side at a time.

Next, apply a small amount of no-clean flux and start soldering. Any bridges will fall away if you re-heat with the iron, but if the bridge is persistent then use de-soldering braid. The soldering iron needs a bit of solder applied to the tip in order to get the solder-wicking effect when using the braid.

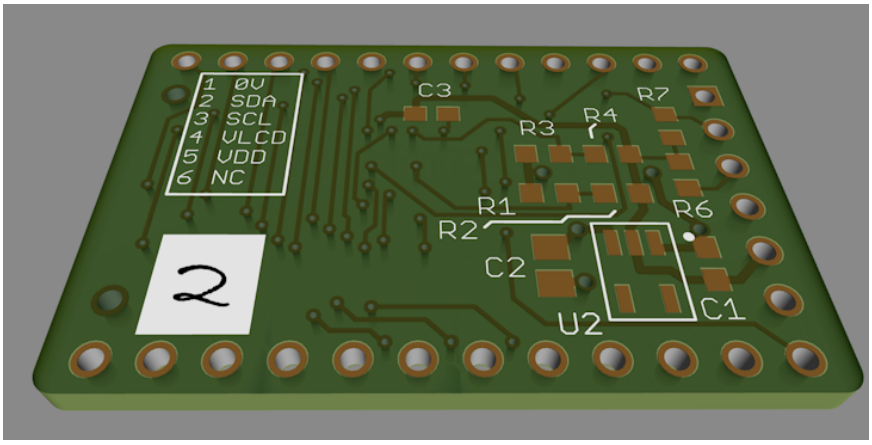
Optionally use pin sockets (or SIL sockets) so that the LCD does not need to be soldered. It is highly recommended that sockets are used to protect the LCD and also because it would be difficult to unsolder the LCD if there was a problem with the PCF85176 that needed examining.
After all the parts have been soldered clean the board using an IPA solution and then examine each pin on the PCF85176 under a lens (push each pin gently with the side of a scalpel blade) to verify all is well.

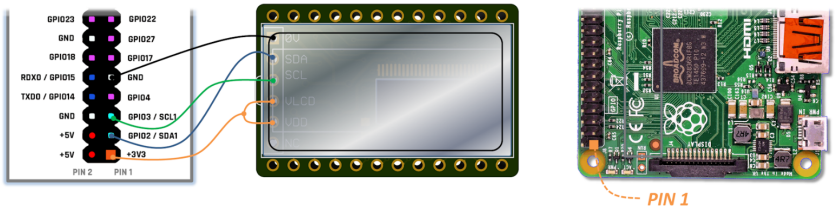
If the display only needs to operate from 3.3V then you don't need to solder the linear regulator IC and its associated capacitors. If multiple displays on the same I2C bus are needed then the addresses are programmed using two 10k resistors that can be soldered in different positions (see the source code for a table which indicates this).

The attached files are slightly improved to the boards that I used, in that there is a white space for labelling them if multiple displays are needed for a single project.



After checks for any short circuits, the LCD can now be connected up to the Pi.





Software

As a quick test some C code was written for the 'Pi. It was tested using a Pi 2 but should work on others, there is nothing special to the code.
To enable I2C on the 'Pi, add the following line to /boot/config.txt (you need to be root user to do this):

+ expand source view plain

```
01. dtparam i2c_arm=on
```

Reboot the Pi and then type (as root user):

+ expand source view plain

```
01. modprobe i2c_dev
```

To make it persistent across reboots you can add the following (as root user) to the /etc/modules file:

+ expand source view plain

```
01. i2c-dev
```

Grab the code from github . To compile the code type the following:

+ expand source view plain

```
01. gcc -c i2cfunc.c -o i2cfunc.o
02. ar rcs libi2cfunc.a i2cfunc.o
03. gcc lcd3-test.c -L. -li2cfunc -o lcd3-test
```

To run it, type:

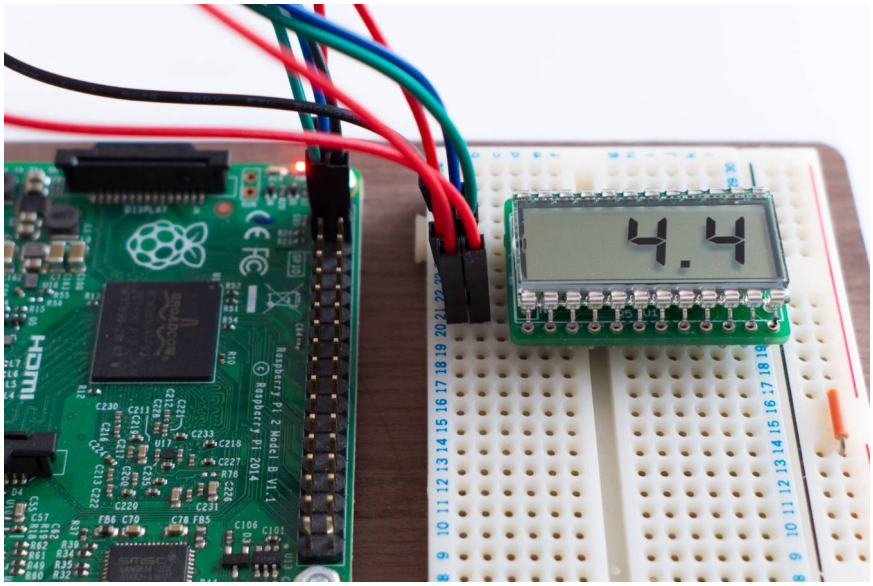
+ expand source view plain

```
01. ./lcd-test
```

Currently the code can display decimal points, digits 0-9 and space.
To use it, call the print_line() function. Here are some examples:

+ expand source view plain

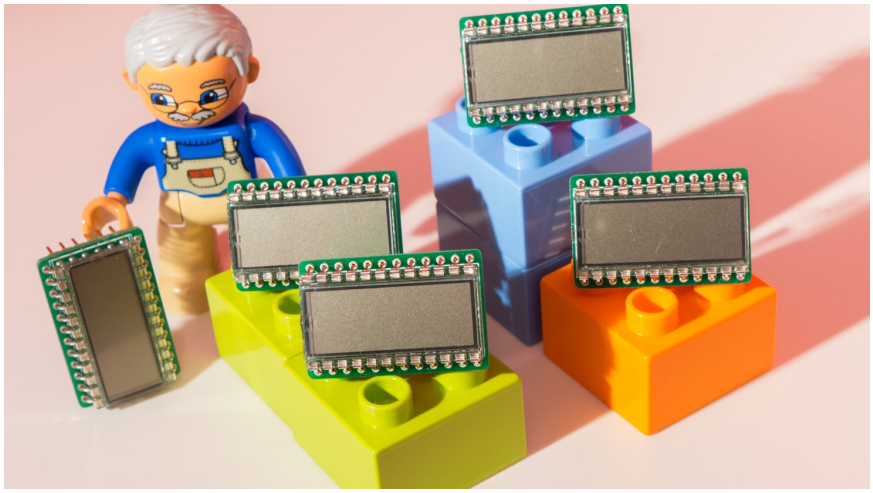
```
01. print_line(" 0.0");
02. print_line("1.23");
03. print_line("    "); // clear the screen
```



The I2C interface is a standard, and it would be very easy to port the code across to any microcontroller/Arduino/etc.

Summary

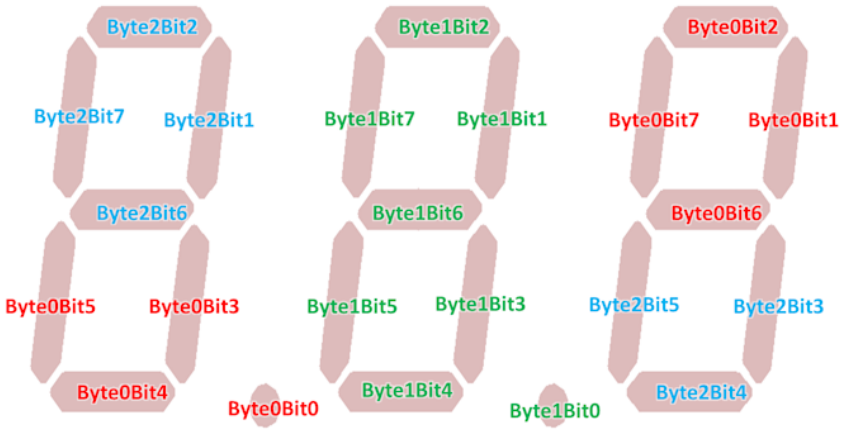
NXP's I2C LCD chip continues to be useful after two decades. The cost, size and low power consumption are attractive for some use-cases. Although this project uses surface mount components they are cheap enough that it could be a useful board to practice with. The very low power consumption means that a couple of AA batteries should last for five years or more with the display always on.



Advanced Use: Extending the Character Set

Many characters are possible with 7-segment displays.

Three bytes represent which segments are to be activated. Refer to the diagram to see the bit mappings.



To extend the character set to display hyphens or alphabet characters, the following array needs to be extended:

```
+ expand source  view plain
const char bit_table[]=
{
  0x38, 0x00, 0x86,    0x00, 0xbe, 0x00,    0x86, 0x00, 0x38,    /* 0 */
  0x08, 0x00, 0x02,    0x00, 0x0a, 0x00,    0x02, 0x00, 0x08,    /* 1 */
  0x30, 0x00, 0x46,    0x00, 0x76, 0x00,    0x46, 0x00, 0x30,    /* 2 */
  0x18, 0x00, 0x46,    0x00, 0x5e, 0x00,    0x46, 0x00, 0x18,    /* 3 */
  0x08, 0x00, 0xc2,    0x00, 0xca, 0x00,    0xc2, 0x00, 0x08,    /* 4 */
  0x18, 0x00, 0xc4,    0x00, 0xdc, 0x00,    0xc4, 0x00, 0x18,    /* 5 */
  0x38, 0x00, 0xc4,    0x00, 0xfc, 0x00,    0xc4, 0x00, 0x38,    /* 6 */
  0x08, 0x00, 0x06,    0x00, 0x0e, 0x00,    0x06, 0x00, 0x08,    /* 7 */
  0x38, 0x00, 0xc6,    0x00, 0xfe, 0x00,    0xc6, 0x00, 0x38,    /* 8 */
  0x18, 0x00, 0xc6,    0x00, 0xde, 0x00,    0xc6, 0x00, 0x18     /* 9 */
};
```

The first group of three bytes indicates the segments that need to be turned on for the digit in the leftmost position, the second group of three refer to the center digit and the last three refer to the rightmost digit. If ASCII sequence is not followed then modify the char_prog function accordingly.

To enable the extra characters, the print_line function case statement checks can be extended.

export-lcd3.zip
9.5 K

export-lcd3-rev2.zip
12.8 K

3269 Views Categories:
Tags: low_power, lcd, display, eagle, nxp, raspberry_pi, eagle_cad, rpiintermediate, eagle-cad, lcd display, feature_tutorial

Average User Rating

(4 ratings)

5 **HELPFUL** **MOST LIKED**

19 Comments

Login or **Register** to comment

DAB Apr 22, 2016 1:53 PM

Great post Shabaz.

You provide good information on how to use different LCD screens and interface them to a MCU.

DAB

1 of 1 people found this helpful

Actions Like (3)

shabaz Apr 22, 2016 2:29 PM (in response to DAB)

Hi DAB,

Thanks!

1 of 1 people found this helpful

Actions Like (0)

clem57 Apr 22, 2016 2:27 PM

shabaz
Curious about the part list:

R2, R4, R5		D.N.F.	0	0.00

I get it is not used, but when and why are they here in the first place/
Clem

Actions Like (1)

shabaz Apr 22, 2016 2:55 PM (in response to clem57)

Hi Clem!
R2 and R4 become relevant if more than one display is connected to an I2C bus.
To send information to the display, the I2C address of the PCF85176 is 0x38, but within the data portion of the I2C stream of data is a separate module address beginning 0x60.
R2 and R4 (together with R1 and R3) are used to select that address, so that it is possible to fit up to four of these modules onto a single I2C bus, and have different content sent to each.
By default only R1 and R3 need to be fitted to select the first address (0x60).
There is a table in the source code that indicates which resistors to fit, depending on the desired address.

+ expand source view plain

```
01. R1    R2    R3    R4    DEVICE_ADDRESS
02. Fit         Fit         0x60
03.       Fit   Fit         0x61
04. Fit               Fit   0x62
05.       Fit         Fit   0x63
```

As for R5, that can be fitted with a zero-ohm link to enable higher current drive to the LCD, but it turns out that the display has very high contrast even without it, so it is not needed. LCD datasheets are often very sparsely written with much information lacking, so I had to wait to test before I could determine if high current drive would be needed or not. Since I was rapid prototyping (the schematic and board layout were complete within hours, in one evening) I decided to leave space for R5 and then not populate it if it was not needed.

1 of 1 people found this helpful

Actions Like (0)

clem57 Apr 22, 2016 4:43 PM (in response to shabaz)

After I saw this, it hit me that you can choose the I2C address using current/no current.R5 I would not have guessed. Thanks
Clem

Actions Like (1)

jw0752 Apr 23, 2016 12:19 AM

Hi Shabaz,
Thank you for sharing this great project. Displays are always needed and this is a good one. As usual your blog is very complete and clear.
John

Actions Like (1)

clem57 Apr 23, 2016 1:09 AM (in response to jw0752)

Right John Wiltrout
Completely clear or clearly complete. Take your pick.
Clem

Actions Like (2)

tonyboubady Apr 23, 2016 1:40 AM

Very informative...this goes to my reference library...

Actions Like (1)

dragonstyne Apr 23, 2016 6:09 AM

Good work Shabaz,

[][][]Steve

Actions Like (1)

shabaz Apr 23, 2016 11:34 AM (in response to dragonstyne)

Hi Steve, tonyboubady and jw0752,

Thanks!

Actions Like (0)

ntewinkel Apr 23, 2016 6:35 PM

Wow! Excellent how-to, and the little board looks great, Shabaz!

Actions Like (1)

ntewinkel Apr 23, 2016 6:36 PM (in response to ntewinkel)

ps, do you do this kind of thing (designing electronics) for a living? Looks totally professional

Actions Like (1)

shabaz Apr 23, 2016 7:56 PM (in response to ntewinkel)

Hi Nico,

Thanks!
In the past I've done circuit design in an R&D lab, and indirectly now, it still helps for work.
Also used self-designed stuff for demo'ing/exercising products.
Sometimes fun stuff, like a while back we used an Atmel chip and LED displays for a competition,
see who can make the product fall over with the least amount of ball throws!
With software it is a habit for me, for example last week I needed to use some virtualization technologies for work, but I did it on an SBC (a Pi) first, to prove things out and start getting my documentation sorted.
By using the actual platform and a different but 'near-enough' platform I sometimes end up learning a lot more about the process and learn subtleties that may otherwise be missed.

3 of 3 people found this helpful

Actions Like (1)

Kalle Pokki Jul 20, 2016 7:46 AM

The schematic under the heading 'Building It' doesn't seem to match the actual segment mapping. That should make the characters appear in separate bytes instead of the real mapping shown in the end of the article.

Actions Like (0)

shabaz Jul 20, 2016 8:50 AM (in response to Kalle Pokki)

Hi Kalle,

Excellent point.. I agree as you say that it should, but either there is a mistake in the LCD datasheet, or we have both misinterpreted it in the same way.
(In general many LCD datasheets are very sparse, some are incorrect, something gets lost in translation!).

As a result, it was found that despite it theoretically mapping to separate bytes with the schematic under 'Building It', in reality it didn't, and so the mapping in the code was used.
A later schematic/PCB revision could correct the mapping to reflect reality, so that the mapping in the code can be

simplified. However it is only three digits to map, and can be done with little code
and at high speed, so I felt it was not worth the effort to re-spin the design, since the software mapping had a low overhead.

In summary the schematic shown, CAD files and graphic mapping diagram and code mappings are all sync'd, and will work together.

Actions Like (1)

Kalle Pokki Jul 21, 2016 2:12 AM (in response to shabaz)

Hi Shabaz,

It's not the LCD datasheet, but the PCB isn't routed according to the schematic. Take e.g. NXP pin 26 near the corner. It should go to LCD pin 16 so that it would light up LCD segment G3. In reality, it goes to LCD pin 21, which indeed lights up LCD segment B1, as discovered in the end of the article.

So it's the classic case of fixing a hardware bug in software .

Actions Like (2)

shabaz Jul 21, 2016 6:48 AM (in response to Kalle Pokki)

Hi Kalle,

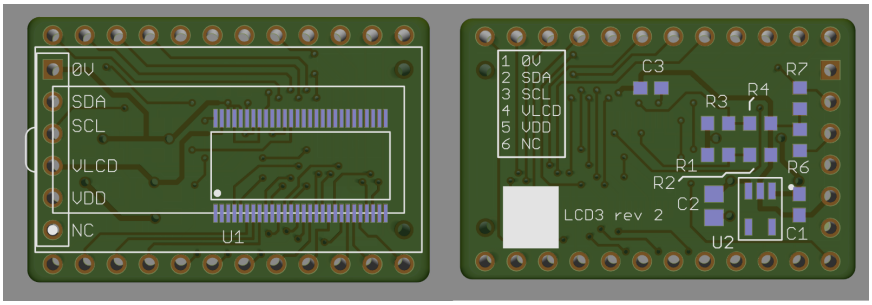
Ahh,, thank you for identifying the root cause! I see it now. You're right, I've just checked, and I'd created the CAD part incorrectly, which leads to the inconsistency, i.e. the datasheet was correct.
It was a silly mistake, I'd numbered the pins 13..24 back-to-front : (Probably did a mirror command of 1..12 which I would normally do, but then this time forgot to manually edit the numbering : (
For SMD I have my own automated scripts so it wouldn't have made this error, but through-hole I still do manually.

Actions Like (1)

shabaz Jul 21, 2016 5:16 PM (in response to Kalle Pokki)

I've edited the board, labelled it revision 2.
Now the mapping should be correct, and doesn't need manipulation in the software.

I won't update the main post, since it isn't tested for real yet, but if you do get a chance to try it (or if I build some more, which I probably will at some stage) then I'll update the main article.
The changes are low risk, since the rev 1 has been proven to work; this rev 2 has been DRC checked.
The CAM files are attached to the main post (since I cannot add them to the comment here) filename **export-lcd3-rev2.zip**
Just to summarise, the original file export-lcd3.zip contains the older mapping that is corrected in software, and the new file export-lcd3-rev2.zip has a corrected mapping which will simplify the code.
The renders below show the export-lcd3-rev2.zip output. (I took the opportunity to tidy the silkscreen slightly too, and remove the resistor R5 which was unused). The inner wide rectangle on the top side show where a cutout/window can be made if it is fitted inside an enclosure. The white filled square on the underside is for labelling with a pen, if several of the boards are to be connected to the same I2C bus.



1 of 1 people found this helpful

Actions Like (0)

modalpdx Jul 26, 2016 8:56 PM (in response to shabaz)

Yay, more excuses to buy more components! This sounds like a simple and fun one to try out. Thanks for writing it up!

Actions Like (1)

Login or **Register** to comment