Project 12 - Writing to a serial LCD with the Raspberry Pi

Outline

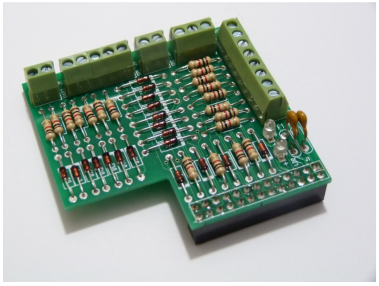
This application note describes how to write to a 2-line LCD display the I2C bus of the Raspberry Pi. Any of the Raspberry Pi models can be used including B, B+ and the Raspberry Pi 2.

Hardware

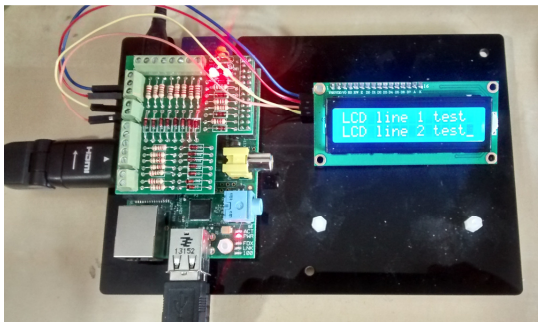
2-Line LCD display: The standard 2 lines x 16 character display has 14 connections. However these are also supplied with a piggy back board which uses the I2C bus. This means that the Raspberry Pi can drive this display using just the SDA and SCL pins. The other 2 pins are supply at 5V and a 0V connection.

There are different versions of this LCD available. The ones used for this project had LCM1602 IIC V1 marked on it.

Custard Pi 1: The Raspberry Pi GPIO is not easy to connect to and can be damaged if the wrong voltage is connected to certain pins. We are going to use the Custard Pi 1 break out board as this has some protection built in to prevent accidental damage.



For ease of assembly, this project uses a Custard Pi base and a prototyping board. The assembled hardware is shown below.



Connecting up the LCD is very easy. Just connect 4 wires from the Custard Pi 1 to the LCD piggy back board. These are 5V and 0V (on connector J3) and SCL and SDA (on connector J7). The LCD looks better with backlighting. To enable this, please solder a wire from pin 1 to pin 16 of the LCD.

Software

The Python code presented here writes to both lines of the LCD and writes blanks to clear the display. It uses the LCD routine which is listed in the Appendix 2. The I2C address of the piggy back board on the LCD is 0x27.

The I2C bus needs to be enabled on the Raspberry Pi before we can use it. Appendix 1 shows how to do this as well as to check the I2C bus address of the piggy back board.

```
#1/usr/bin/env python

import time
import pylcdlibseg

#start program

lcd = pylcdlibseg.lcd(0x27,1)

while True:
    lcd.lcd_clear
    time.sleep (0.5)
    lcd.lcd_puts("LCD line 1 test",1)  #display text on line 1
    time.sleep (0.5)
    lcd.lcd_clear
    lcd.lcd_puts("LCD line 2 test",2)  #display text on line 1
    time.sleep (0.5)
    lcd.lcd_puts("                ",1)
    lcd.lcd_puts("                ",2)
    time.sleep(1)

import sys
sys.exit()
```

(Download this code from here: http://www.sf-innovations.co.uk/downloads)

Notes:
The Custard Pi 1 and Custard Base are available from amazon.co.uk and directly from SF Innovations. The code presented here can be downloaded from the SF Innovations website.

Ideas for Advanced projects

Buy a 4 line display and adapt the demo code to write to this.

APPENDIX 1

Setting up the I2C Bus

By default, the I2C bus routines are turned off in the operating system. The following steps need to be followed to enable these.

Step 1

At the command prompt type:

```
sudo nano /etc/modules
```

This uses the nano editor to make some changes to the modules file.Add the following two lines to this file

i2c-bcm2708
i2c-dev

Then save and exit the file using CTRL-x and Y.

Step 2

Make sure that you have the I2C utilities installed by executing the following two commands. The Pi will need to be connected to the Internet for this.

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

If you get a 404 error do an update first:

```
sudo apt-get update
```

Note : The installation could take a few minutes to do, depend on how busy the server is.

Now add a new user to the i2c group:
`sudo adduser pi i2c`

Step 3

On the Raspberry Pi, the I2C and the SPI buses are usually disabled. This is done in the /etc/modprobe.d/raspi-blacklist.conf file.

If this file is not present then there is nothing to be done. Otherwise edit the file by typing the following at the command prompt.

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

If the I2C and the SPI is blacklisted, you will see the following commands.

blacklist spi-bcm2708
blacklist i2c-bcm2708
Insert a # in front of these to comment them out.

Then save and exit the file using CTRL-x and Y.

After editing the file, you will need to reboot for the changes to take effect.

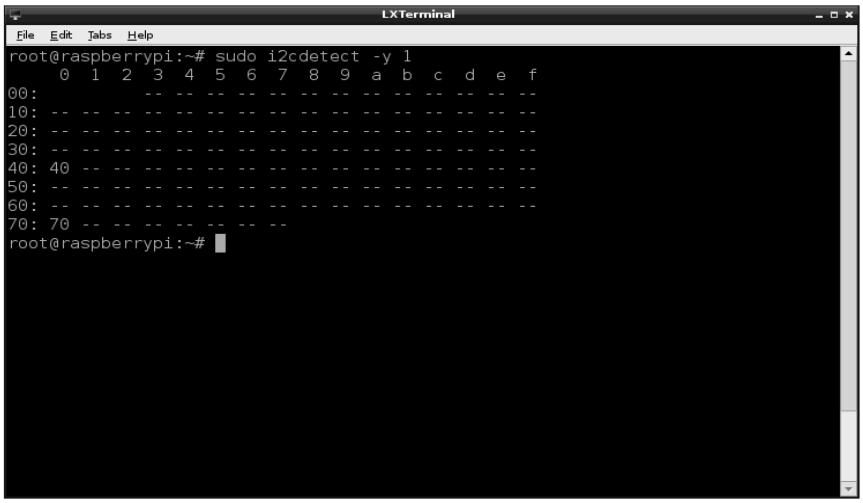
Step 4

Now we need to test if the I2C bus is working correctly.
Connect up the Custard Pi 6 board (or any other I2C bus device) and run the following command.

`sudo i2cdetect -y 1` (for Rev 2 boards which uses port 1)

Or
`sudo i2cdetect -y 0` (for Rev 1 boards which uses port 0)

If everything is OK, then the I2C address of the device will be shown as on the following slide. This shows two devices with address 40 and 70 in hexadecimal code. **(Note: We should get 27 for our project when the LCD is connected up).**



Now that we have set up the I2C serial bus routines, we are ready to drive the relays on the Custard Pi 6.

APPENDIX 2

This routine initialises the LCD and writes text to line 1 or line 2 of the display using the lcd_puts function.

```
import smbus
from time import sleep

# General i2c device class so that other devices can be added easily
class i2c_device:
 def __init__(self, addr, port):
  self.addr = addr
  self.bus = smbus.SMBus(port)

 def write(self, byte):
  self.bus.write_byte(self.addr, byte)

 def read(self):
  return self.bus.read_byte(self.addr)

 def read_nbytes_data(self, data, n): # For sequential reads > 1 byte
  return self.bus.read_i2c_block_data(self.addr, data, n)


class lcd:
 #initializes objects and lcd
 def __init__(self, addr, port):
  delay=0.0005
  self.lcd_device = i2c_device(addr, port)
  self.lcd_device.write(0x30)
  self.lcd_strobe()
  sleep(delay)
  self.lcd_strobe()
  sleep(delay)
  self.lcd_strobe()
  sleep(delay)
  self.lcd_device.write(0x20)
  self.lcd_strobe()
  sleep(delay)

  self.lcd_write(0x28)
```

```
 self.lcd_write(0x08)
 self.lcd_write(0x01)
 self.lcd_write(0x06)
 self.lcd_write(0x0C)
 self.lcd_write(0x0F)

# clocks EN to latch command
def lcd_strobe(self):
  self.lcd_device.write((self.lcd_device.read() | 0x04))
  self.lcd_device.write((self.lcd_device.read() & 0xFB))

# write a command to lcd
def lcd_write(self, cmd):
  self.lcd_device.write((cmd >>4) <<4)
  self.lcd_strobe()
  self.lcd_device.write((cmd & 0x0F)<<4)
  self.lcd_strobe()
  self.lcd_device.write(0x0)

# write a character to lcd (or character rom)
def lcd_write_char(self, charvalue):
  self.lcd_device.write((0x01 | (charvalue >>4)<<4))
  self.lcd_strobe()
  self.lcd_device.write((0x01 | (charvalue & 0x0F)<<4))
  self.lcd_strobe()
  self.lcd_device.write(0x0)

# put char function
def lcd_putc(self, char):
 self.lcd_write_char(ord(char))

# put string function
def lcd_puts(self, string, line):
 if line == 1:
  self.lcd_write(0x80)
 if line == 2:
  self.lcd_write(0xC0)
 if line == 3:
  self.lcd_write(0x94)
 if line == 4:
  self.lcd_write(0xD4)

 for char in string:
  self.lcd_putc(char)

# clear lcd and set to home
def lcd_clear(self):
 self.lcd_write(0x1)
 self.lcd_write(0x2)

# add custom characters (0 - 7)
def lcd_load_custon_chars(self, fontdata):
 self.lcd_device.bus.write(0x40);
 for char in fontdata:
  for line in char:
   self.lcd_write_char(line)
```

(Download this code from here: http://www.sf-innovations.co.uk/downloads)