Setup LCD 2x16 Screen I2C Interface

Group Name: MegaTronSFU
Group Members: Devon Briere, NoorUllah Randhawa, Hansen William, Nathan Dhami
Course: CMPT 433 - Embedded Systems

The purpose of this guide is to show the user how to connect the LCD with I2C interface to the beaglebone green, setup GPIO pins, cross compile the LCD interface library, and run/test example code.

Beaglebone commands **#**
Host commands **$**

Getting Started

Before getting started, make sure that you have the right model. It should be the 2x16 LCD screen with the I2C module that comes with it.

Source: https://leeselectronic.com/en/product/15547.html

Tools and Hardware needed

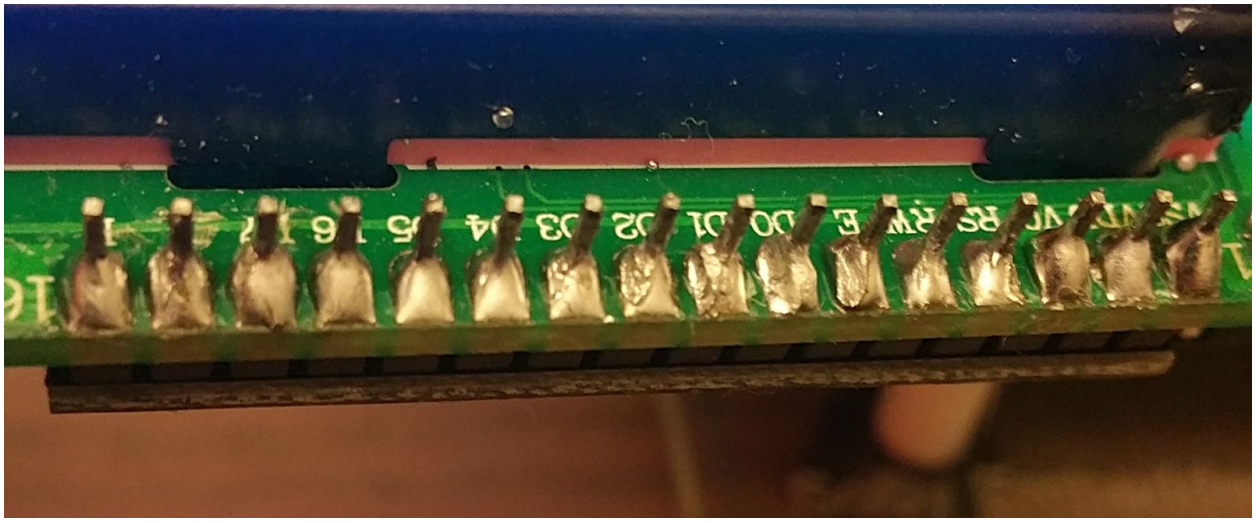
1. Soldering iron kit
2. LCD 2x16 with I2C interface
3. Beaglebone Green
4. 4x Jumper Wires Male to Female

Software Library needed

1. LCD library (github link provided in section 3)

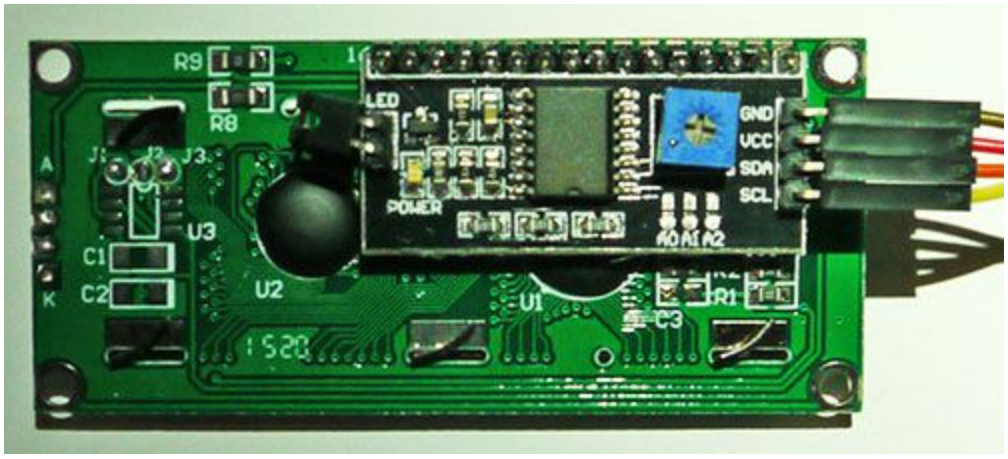
Installing backpack

The LCD screen and I2C backpack may not come attached. If so, the screen will first have to be soldered to the 16 pins of the backpack before proceeding.



If you are unsure how to solder, please watch this youtube video:
https://www.youtube.com/watch?v=uzxw1yl1s_M&t=347s

Setup GPIO Pins



We connected using Male to Female jumper wires to the following GPIO pins on the beaglebone green:
GND to P9 pin 1
VCC to P9 pin 7
SCL to P9 pin 17
SDA to P9 pin 18

Cross-compile Library

Since the driver for lcd is written in C++, we will have to download the g++ cross compiler on the host machine if not done already.

$ sudo apt-get install g++-arm-linux-gnueabihf

The official release of the liquidCrystal library used the Arduino specific 'Wire.h' library to communicate over I2C, but Kaj-Michael Lang uploaded a version using generic linux I2C functions. Clone this repository to your work directory.

$ git clone https://github.com/oniongarlic/liquidcrystal-i2c.git

Adjust the makefile to use the arm g++ cross compiler to create the test program in the correct directory.

```
TARGET= lcd-test

TARGET= lcd
SOURCES= main.cpp I2CIO.cpp LCD.cpp LiquidCrystal_I2C.cpp
PUBDIR = $(HOME)/cmpt433/public/myApps
OUTDIR = $(PUBDIR)
CROSS_TOOL = arm-linux-gnueabihf-
CC_CPP = $(CROSS_TOOL)g++
CC_C = $(CROSS_TOOL)gcc
CFLAGS = -Wall -g -D _POSIX_C_SOURCE=200809L -std=c++11

all:
	$(CC_CPP) $(CFLAGS) $(SOURCES) -o $(OUTDIR)/$(TARGET)  $(LFLAGS) -lpthread
clean:
	rm -f $(OUTDIR)/$(TARGET)
```

At this point there will still be a few compile errors that need to be fixed:

1. Include the ioctl header in I2CIO.h
   #include <sys/ioctl.h>
2. 'I2C_SMBUS_WRITE' was not declared in this scope because the compiler does not have access to the arm i2c-dev.h. This can be circumvented by defining the missing functions inline using ioctl. Add the following code to I2CIO.CPP:

```
#define I2C_SMBUS_WRITE 0

#define I2C_SMBUS_BYTE       1

inline __s32 i2c_smbus_access(int file, char read_write, __u8 command,
                              int size, union i2c_smbus_data *data)

{
  struct i2c_smbus_ioctl_data args;

  args.read_write = read_write;
  args.command = command;
  args.size = size;
  args.data = data;
  return ioctl(file,I2C_SMBUS,&args);
}

inline __s32 i2c_smbus_write_byte(int file, __u8 value)
{
  return i2c_smbus_access(file,I2C_SMBUS_WRITE,value,
                          I2C_SMBUS_BYTE,NULL);
}
```

The test program will now cross compile and run, but nothing will display on the LCD until the the main code is configured to the correctly.

Example Code

Below is the example test code in liquidcrystal-i2c/main.cpp (shortened version). It shows the initialization code for the LCD and shows some methods that can be used to interact with the LCD.

```
/**
 ** i2c LCD test application
 **
 ** Author: Kaj-Michael Lang <milang@tal.org>
 ** Copyright 2014 - Under creative commons license 3.0 Attribution-ShareAlike CC BY-SA
 **/
#include "LiquidCrystal_I2C.h"

int main (int argc, char *argv []) {
// i2c address
uint8_t i2c=0x3f;
// Control line PINs
uint8_t en=2;
uint8_t rw=1;
uint8_t rs=0;

// Data line PINs
uint8_t d4=4;
uint8_t d5=5;
uint8_t d6=6;
uint8_t d7=7;

// Backlight PIN
uint8_t bl=3;

// LCD display size
uint8_t rows=2;
uint8_t cols=16;

LiquidCrystal_I2C lcd("/dev/i2c-1", i2c, en, rw, rs, d4, d5, d6, d7, bl, POSITIVE);

lcd.begin(cols, rows);

lcd.on();
lcd.clear();
...
}
```

The first highlighted text is the i2c address of the LCD device. The address can be either 0x27 or 0x3f. If you want to be sure the address is correct, use i2c tools to see which i2c bus the LCD is connected to and then find the i2c address assigned to the LCD. Check the first i2c bus with:

```
$ i2cdetect -y -r 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- 1c -- -- --
20: 20 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- 3f
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

You must also need to make sure the hardware bus for I2C1 is enabled by exporting the correct cape with the command:

```
# echo BB-I2C1 > /sys/devices/platform/bone_capemgr/slots
```

This guide will not cover details on how to use i2c tools or install i2c tools but please see Brian's guide on using i2c in linux for more information.

The second and third highlighted source code are the variables for configuring the display size. SInce this is a 2x16 LCD screen, set rows = 2 and cols = 16.

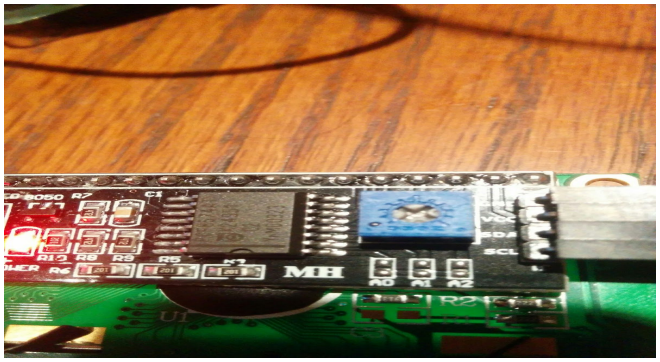
Here are steps on how to run example code on the beaglebone:

1. $ cd liquidcrystal-i2c
2. $ make
3. `# cd /mnt/remote/myApps`
4. `# ./test-lcd`

You should see white characters being displayed on the LCD screen while the program is running.

Troubleshoot

- If you followed all directions correctly and still don't see anything on the LCD screen after running the test code, please check the contrast level and make sure that it is not too low. To change the contrast level, use a screwdriver to twist the middle of the blue compartment shown in the picture below clockwise. Twisting it counterclockwise will lower the contrast levels.



- Remember to set the row and col variable to 2 and 16 respectively
- Make sure the correct I2C address is configured. It is either 0x27 or 0x3f. It would better to check using i2c tools to know for sure. Please see Brian's guide on how to use i2c in linux.
- Check that the i2c backpack is soldered correctly to the back of the LCD screen. Be very careful when soldering, it will be hard to undo it unless you're experienced in soldering.
- Ensure the GPIO setup is correct. See section 2 for mapping.