Date:	01 April 2020
Author:	David Witten
Pages:	7

Setting up the Single Board Computers (SBCs) running Linux for testing the Magnetometer Support Boards (MSBs)

Setting I2C up on Single Board Computers(SBCs) is usually fairly simple. Here are some examples. Most boards running Linux kernels with recent distributions of Ubuntu, Debian, or related distributions will be similar.

These instructions assume the use of a single board computer with an equivalent to the 40 pin GPIO bus used by Raspberry Pi's. It is possible to use either of the Magnetometer Support Board variants with Arduino or other microcontrollers that support a 3.3v I2C bus, but that is not covered here.

The Magnetometer Support Boards are NOT 5.5v tolerant on the I2C bus. Only the extender version uses 5v, and it only uses it on the 'External 5v' connector.

Warning: Applying 5v (or higher) to any other pins will damage the magnetometer.

I also assume that you have the following incidental items:

- SparkFun Qwiic pHAT v2.0 for Raspberry Pi https://www.sparkfun.com/products/15945
- Qwiic Cable – 100mm (or longer) https://www.sparkfun.com/products/14427

For MSB with Extender function:

- 1 pair Jumper Wires 6" M/F (like https://www.sparkfun.com/products/9140)
- For bench testing - Five foot sheilded twisted pair CAT-6a cable: https://www.amazon.com/dp/B00HEM5MZU/ref=psdc_9938478011_t3_B00HEM653S

And/Or:

- For remote use - 100 foot sheilded twisted pair CAT-6a cable: https://www.amazon.com/gp/product/B00HEM653S/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1

Notes for setting I2C up to work on Odroid N2

Setting the I2c bitrate on the Odroid-N2

The Personal Space Weather Station project specifies an I2C bit rate of 100k. The default on many buses (like the Odroid N2 bus 1) is 400k or higher. This needs to be adjusted as the extender function may not work reliably at higher rates.

How to detect current I2C bus speed:

```
dave@n2:~$ sudo cat /sys/bus/i2c/devices/i2c-2/
```

It will probably return 400000, indicating that the default bit rate is set to 400k - to fast for our project!

How to change I2c bus speed until next reboot:

```
dave@n2:~$ sudo su
root@n2:~# sudo echo 100000 > /sys/bus/i2c/devices/i2c-2/
root@n2:~# exit
```

How to change the I2C bus speed so that the new speed will work across reboots:

```
dave@n2:~$ sudo apt update
dave@n2:~$ sudo apt install device-tree-compiler
dave@n2:~# sudo cp /media/boot/meson64_odroidn2.
dave@n2:~# sudo dtc -I dtb -O dts /media/boot/meson64_odroidn2.dtb >
        /media/boot/meson64_odroidn2_disasm.dts
```

(ignore all the warnings)

then edit :

```
root@n2:~# nano /media/boot/meson64_odroidn2_disasm.dts
```

Find and change:

```
i2c@1d000 {
        compatible = "amlogic,meson-g12b-i2c";
        status = "okay";
        reg = <0x0 0x1d000 0x0 0x20>;
        interrupts = <0x0 0xd7 0x1 0x0 0x5e 0x1>;
        #address-cells = <0x1>;
```

```
        #size-cells = <0x0>;
        clocks = <0x2 0x2a>;
        clock-names = "clk_i2c";
        pinctrl-names = "default", "gpio_periphs";
        pinctrl-0 = <0x26>;
        pinctrl-1 = <0x27>;
        clock-frequency = <0x61a80>;
        phandle = <0xaf>;
};
```

To:

```
i2c@1d000 {
        compatible = "amlogic,meson-g12b-i2c";
        status = "okay";
        reg = <0x0 0x1d000 0x0 0x20>;
        interrupts = <0x0 0xd7 0x1 0x0 0x5e 0x1>;
        #address-cells = <0x1>;
        #size-cells = <0x0>;
        clocks = <0x2 0x2a>;
        clock-names = "clk_i2c";
        pinctrl-names = "default", "gpio_periphs";
        pinctrl-0 = <0x26>;
        pinctrl-1 = <0x27>;
        clock-frequency = <0x186a0>;
        phandle = <0xaf>;
};
```

NOTICE: changed part is: clock-frequency = <0x61a80>; changed to: clock-frequency = <0x186a0>;

Then save to a new file name - I used: '/media/boot/meson64_odroidn2_mod-i2c-2@1000k.dts'

Next, compile the new file:

root@n2:~# sudodtc -I dts -O dtb '/media/boot/meson64_odroidn2_mod-i2c-2@1000k.dts' > /media/boot/meson64_odroidn2.dtb (ignore all the warnings)

Then reboot.

If this has worked, you can do this again to check the new default speed:

dave@n2:~$ sudo cat /sys/bus/i2c/devices/i2c-2/

It should return: 100000

Notes for setting I2C up to work on RAS-Pi 3 & 4

I will assume that you will use the current default full Raspbian OS. It uses the same image for all versions:

I suggest:

Raspbian Buster with desktop
Image with desktop based on Debian Buster
Version:February 2020
Release date:2020-02-13
Kernel version:4.19
Size:1136 MB
https://downloads.raspberrypi.org/raspbian_latest

When you have this zip file, you need to uncompress it with a zip64 compatible unzip tool:

My favorite for Windows is 7-Zip. for Mac, apparently The Unarchiver is equivalent. for Linux Unzip works.

The instructions from the official Raspberry Pi site say:

“balenaEtcher is a graphical SD card writing tool that works on Mac OS, Linux and Windows, and is the easiest option for most users. balenaEtcher also supports writing images directly from the zip file, without any unzipping required. To write your image with balenaEtcher:

Download the latest version of balenaEtcher and install it.
Connect an SD card reader with the SD card inside.
Open balenaEtcher and select from your hard drive the Raspberry Pi .img or .zip file you wish to write to the SD card.
Select the SD card you wish to write your image to.
Review your selections and click 'Flash!' to begin writing data to the SD card.
Note: for Linux users, zenity might need to be installed on your machine for balenaEtcher to be able to write the image on your SD card.

For more advanced control of this process, see our system-specific guides:

Linux
Mac OS
Windows
Chrome OS
There are other ways to do ths as well, but this should be safe.”

Now you need to plug in monitor, keyboard, and mouse. I prefer a wireless keyboard/mouse combo. If possible, connect a wired network connection at first, but a reliable WiFi environment should work.

Then power up the Pi and let it guide you through the setup. Then let it restart,

In recent versions, you can select Menu > Preferences > Raspberry Pi Configuration and then you get an applet thing. It you select the tab "Interfaces" you can turn I2C and I also Select SSH. You might also consider Serial Port and VNC.

Alternatively, from a terminal window/command prompt, you can type sudo raspi-config and make these changes.

Again, you should reboot the Pi.

Procedure for all SBCs

Assuming that you have completed the appropriate configuration above, the next step is to make sure the appropriate devices are now available.

Open a terminal window and type ls /dev/i2c*. You should see it respond with entries like "/dev/i2c-1". This indicates that the system driver is loaded for I2C bus 1. It will be different on different types of SBC. I have one that lists 0 – 8.

The Raspberry Pi boards should show: "/dev/i2c-1".
The Odroid N2 will show: "/dev/i2c-2".
There may be others, but these are the ones we care about.

For Raspberry Pi:

Now you want to type the command "i2cdetect -y 1"

For Odroid N2:

Now you want to type the command "i2cdetect -y 2"

You should see something like this:

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

The dashes are unoccupied addresses (hexadecimal) on I2C bus 1

The numbers or dashes indicate the presence (or absence) of devices at those addresses, for example at 0x37 (in C syntax)

Now connect the SBC to the MSB module. Plug in the Qwiic cable and the Magnetometer unit (use either connector).

With the Simple MSB attached you should see:

```
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:         -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- 18 -- -- -- -- -- -- --
20: 20 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

On the **Simple MSB** there two devices. The one at 0x18 is the MCP9808 precision temperature sensor. The one at 0x20 is the magnetometer module.

MSB with I2C Extender

Initially you may see only one device. It is on the local board. The device should be at 0x19 and is the MCP9808 precision temperature sensor on the local module.

```
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:         -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- 19 -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

When the shielded twisted pair Cat 6a cable is attached and 5v is applied to the local module the sensors on the remote board should appear as well. The one at address 0x18 is the remote MCP9808 precision temperature sensor. The one at address 0x20 is the magnetometer module.

```
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:         -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- 18 19 -- -- -- -- -- --
20: 20 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

It is the job of the software to request register values from these addresses and to format them into something we can read.