

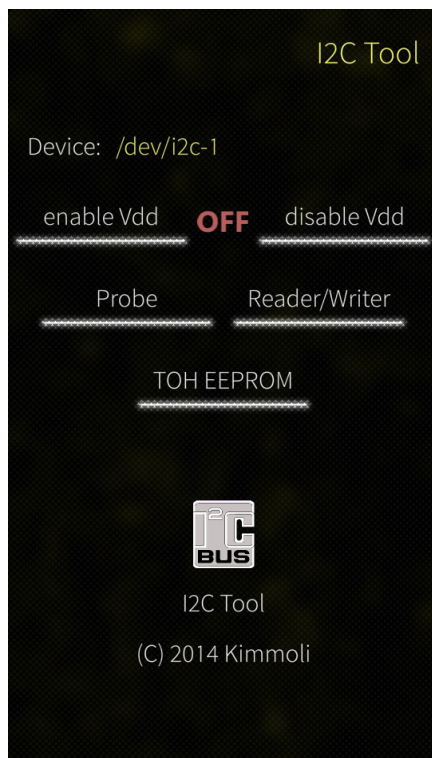


The i2ctool for Jolla is intended mainly for TOH developers.

Suggested additional reading:

TheOtherHalf Spec from developer kit <https://jolla.com/the-other-half-developer-kit>

I2C Primer e.g. here <http://www.i2c-bus.org/i2c-primer/>



Starting application gives you the main-menu view

← Bus/dev selection combo-box. TOH is in `/dev/i2c-1` (default), others are internal to phone.

← TOH VDD control. Controls and shows the state of 3.3V output pin.

← Probe. Scans I2C devices in selected bus.

← Reader/Writer gives you access to individual I2C devices.

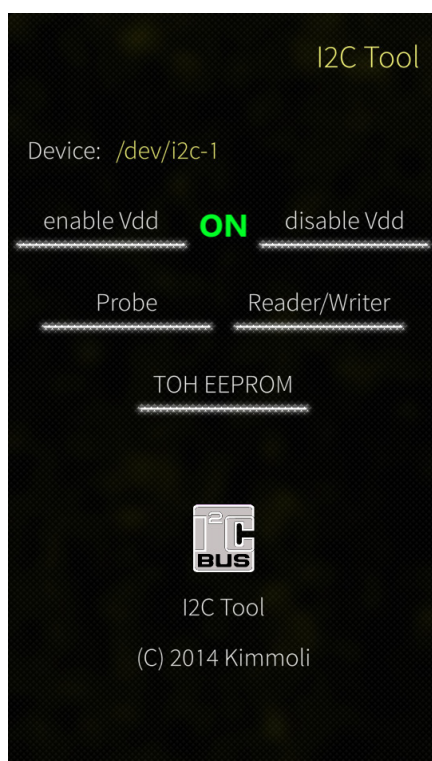
← TOH EEPROM is used to program 2kbit EEPROM at address 0x50 with contents that are fetched to sysfs by Jolla tohd (toh-daemon) when TOH is attached to the phone, and microswitch is pressed.



Clicking on the Device: Combobox, you can see the other busses, but it is recommended to use only `/dev/i2c-1`.

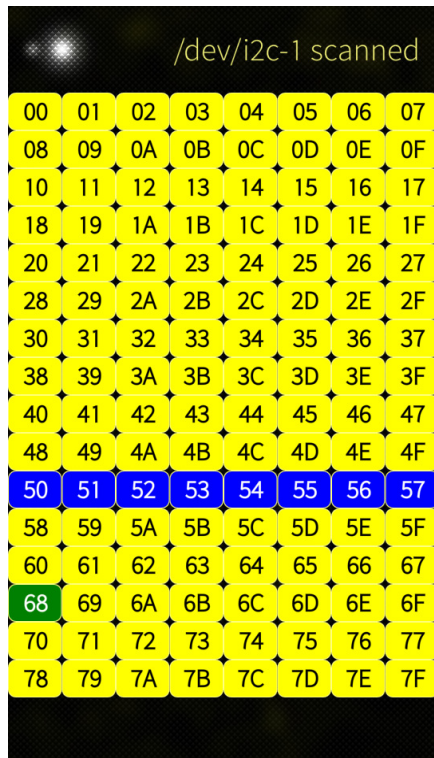
As a regular user you don't even have access to the other busses.

Consider that You have been warned.








Now, if you have TOH with I2C devices, or MyHalf, you might want to connect that to the phone.

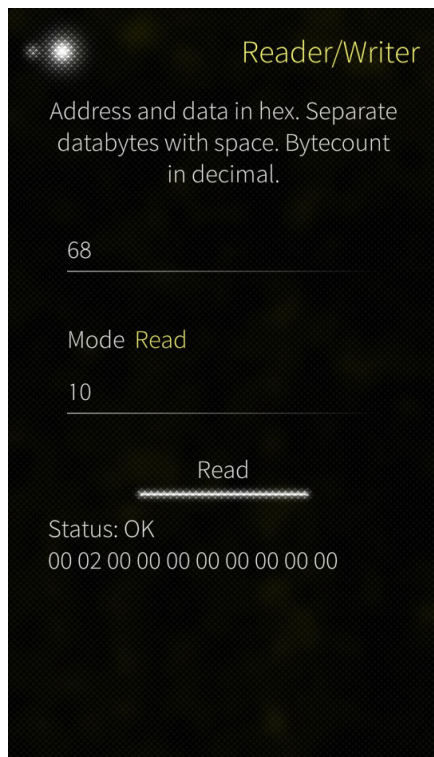
First thing is to click on enable Vdd button. It will enable the 3.3V on pogo pin.



Next thing to do is to make some probing.

The output is color-coded.

	Black - Address is not scanned yet
	Yellow – Address scanned but nothing didn't ACK.
	Blue – IOCTL failed to read, probably locked by kernel/tohd daemon
	Red – Open failed. Probably you are scanning other that /dev/i2c-1.
	Green – Success. I2C device did ACK in this address.



Now, we know that a device exists in I2C address 68. It is time to access that.

Note that address notation here is hexadecimal and 7-bit without the R/W bit.

Click on Reader/Writet button from main-menu, and click on Enter device address –field, and type the found device address (in this case 68).

The default mode is Read, we need just to enter number of bytes to read.

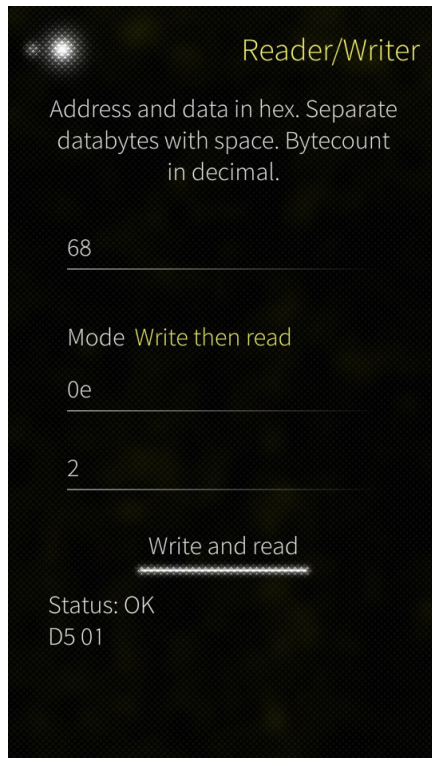
← Then click on Read button.

← Status shows ERROR or OK

← Below status, are the bytes read from the device.

A6	A5	A4	A3	A2	A1	A0	R/W
----	----	----	----	----	----	----	-----

Address notation consists from bits A6:0



Write then Read mode is one of the most common ways to make random read accesses to an I2C device. First you write pointer register value, then you read starting from that register.

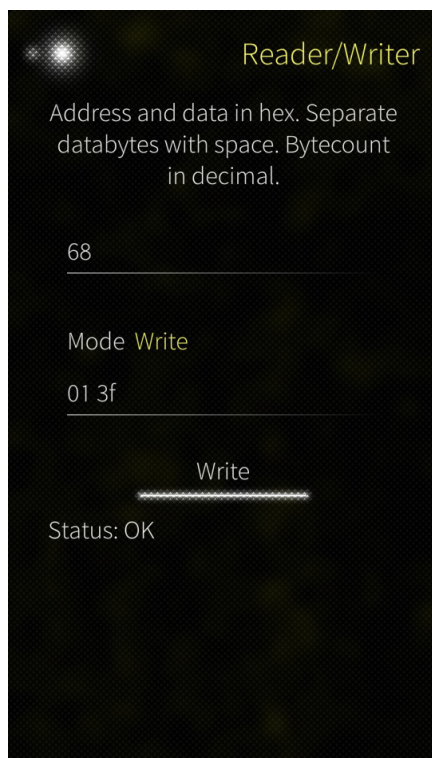
The device here is AMG88xx Infrared grid sensor, and according to the datasheet, it has thermistor, which value can be read from register 0x0E (low) and 0x0F (high).

Enter first address, 0e, to the databytes to write – field. This will be written to the device first.

Then enter 2 as number of bytes to read, and click Write and read.

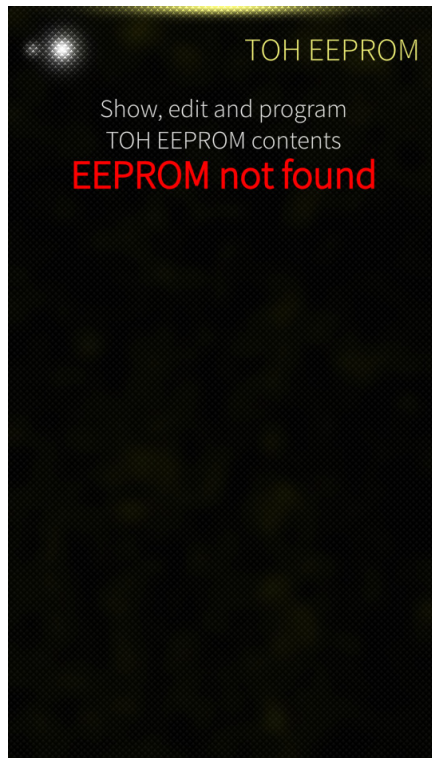
Now the device returned values D5 and 01.

0x01d5 → resolution 0.0625 °C/LSB → 29.3125 °C which sounds realistic.



To just write, typically enter register address as first byte, and data as second byte.

In this example writing 0x3F to register at 0x01



EEPROM stuff.

If you click the TOH EEPROM button when you have a TOH which presses the microswitch (You will also see in probe that Blue bingo at 50..57 addresses, you are not able to access the EEPROM. It is locked by tohd or toh-core.

You need to unbind toh-core first either by starting i2ctool as root user (not recommended) or by entering following command as root in terminal:

```
echo toh-core.0 >  
/sys/bus/platform/drivers/toh-  
core/unbind
```

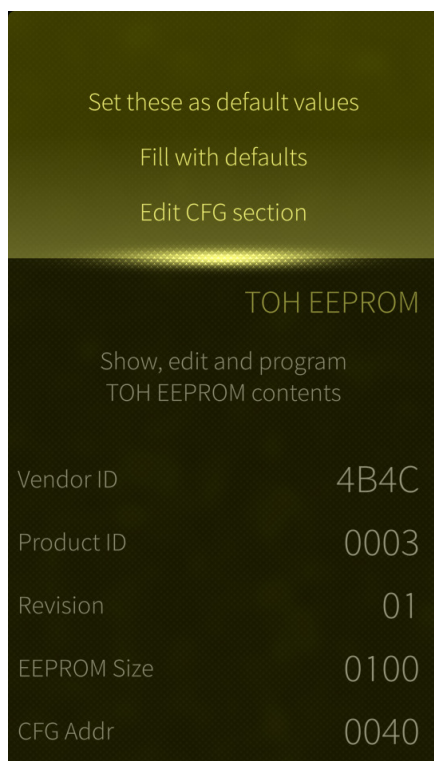


After unbinding, and reclicking TOH EEPROM button (or if you didn't have that dimple on your TOH) you'll see page with values read from the EEPROM. If your EEPROM is empty, these will all be FFFF.

By clicking a value, dialog to edit that value opens.

After setting values, click Write. It will write them to the EEPROM.

Note: Unfortunately there is no feedback about success or fail – maybe in next version.



Pull-down menu has few selections.

“Set these as default values” stores the present values to QSettings of i2ctool application.

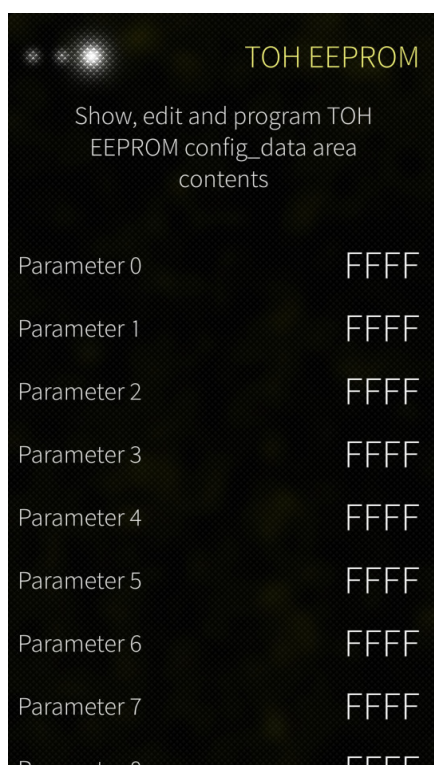
“Fill with defaults” will set the values according to the saved defaults.

*Note Vendor ID 0x4B4C is **my** vendor id. Do not use that. You can register your own vendor id by letting me know. (as there is not yet official Jolla way to do it, I had to take own control on this issue)*

Edit CFG section allows you to edit configuration data section. (UDATA not used)

CFG Addr: 0040, CFG Size: 2x Parameters +1

UDATA Addr 0080, Size 0000



Configuration data editor allows you to set 16-bit values, which can be accessed through sysfs.

EEPROM contents are read when entering this page.

More details about this can be obtained by e.g. contacting me.

Click parameter to edit it. At the end of list of 32 parameters, there is a Write –button to write these to the EEPROM.