Appendix C: I2C Devices

The I2C standards only specify how data is sent from device to device. It does not specify the layout of the registers of a device. LEGO, however, has guidelines for 3rd party manufactures so that they can provide sensors with a (fairly) uniform register layout.

We call sensors that were designed following LEGO's guidelines **NXT/I2C** sensors. This common register layout lets us autodetect the type of sensor and proves access to the sensor via the lego-sensor class.

We refer to sensors that do not conform to LEGO's specifications as **Other/I2C** sensors. There are so many types of I2C chips in the wild that are already supported on Linux that we do not attempt to autodetect them. To use them, we just need to find a compatible driver and manually load it.

This page discusses both types of I2C sensors.

Addressing

I2C uses a 7-bit addressing scheme (there is also 10-bit addressing but it is not implemented in the ev3dev I2C driver). When sending an address over the bus, the address is shifted to the left 1 bit and the least significant bit is used to indicate read or write.

> **Note**
>
> The I2C address that is used in ev3dev is different from the other EV3/NXT programming languages/environments. **This means the address in your sensors' documentation is probably not the address that you need for ev3dev!** In ev3dev (and Linux in general), we used the **unshifted** 7-bit address.

I2C addresses 0x01 through 0x07 (unshifted) are reserved for special use by the I2C specifications. However, these addresses are used by some sensors anyway (most notably the NXT Ultrasonic sensor). The ev3dev kernel has been patched to allow these to work, but some userspace tools will not work with devices at these addresses. For example, we distribute a patched version of the `i2c-tools` package to work around this.

There is a table of I2C addresses at the end of the page.

Using NXT/I2C Sensors

See the page on the lego-sensor class for general usage. This page only covers the I2C specifics.

Polling

When we say "polled", we just mean that the EV3 brick initiates a read command to read data from the sensor. The data that is read depends on the current mode that is selected. You can change the polling rate using the *poll_ms* attribute (of the `lego-sensor` device). You can also disable polling by setting `poll_ms` to 0. When polling is disabled, you can initiate a data read by setting the mode again. By default, NXT/I2C sensors are polled every 100 milliseconds. The default value can be changed via a module parameter.

Direct Reading and Writing of the Sensor

> **Warning**
>
> Be very careful when reading from or writing to your sensors. It is theoretically possible to break them if you read or write to the wrong register.

In most cases, setting the mode of a sensor will write the proper data if necessary, so you don't actually need to write data using this method. However, it is possible to write arbitrary data to I2C sensors using the `direct` attribute. Use `seek` to specify the register to read or write from and always specify the number of bytes to read or write.

Example: Reading the white calibration data from the mindsensors.com Light Sensor Array. This reads 8 bytes from register 0x5A.

```
$ hd -s $(( 0x5A )) -n 8 direct
```

Example: Sending a "calibrate white" command to the mindsensor.com Light Sensor Array. This just writes the ascii character *W* to register 0x41.

```
$ echo -e -n "W" | dd bs=1 of=direct seek=$(( 0x41 ))
```

Manually Loading Devices

If you have autodetection disabled (e.g. using the `other-i2c` mode of a port) or if you have managed to change the I2C address of your sensor to something other than the default or you are using something that is not even a LEGO compatible sensor, you will have to manually load

a device in order to be able to use your sensor. We just have to tell the I2C adapter which driver to use and the address of the device.

The I2C adapter device nodes are at `/sys/bus/i2c/devices/i2c-N` where N is the number of the input port plus 2. To load a device, we write to the `new_device` attribute. NOTE: These nodes only exist when you have an I2C sensor plugged into an input port or the port was manually set to an I2C mode.

Example:

```
# echo nxt-i2c-sensor 0x0B > /sys/bus/i2c/devices/i2c-5/new_device
```

Using Other/I2C Sensors

As we already discussed, Other/I2C sensors generally have an existing Linux driver that you can use. This means that each sensor will work a bit differently. You can load a device just like for manually loading an NXT/I2C device, except we use a different driver name. You can find the names of drivers here.

Example: Using the mindsensors.com Realtime Clock Sensor on input port 2.

```
$ echo ds1307 0x68 > /sys/bus/i2c/devices/i2c-4/new_device
$ dmesg | tail
...
i2c-legoev3 i2c-legoev3.4: registered on input port 2
i2c i2c-4: new_device: Instantiated device ds1307 at 0x68
rtc-ds1307 4-0068: SET TIME!
rtc-ds1307 4-0068: rtc core: registered ds1307 as rtc1
rtc-ds1307 4-0068: 56 bytes nvram
$ cd /sys/class/rtc
$ ls
rtc0    rtc1
$ cd rtc1
$ ls
date  device   max_user_freq  since_epoch  time
dev   hctosys  name           subsystem    uevent
```

Now, I just need to figure out what to do with TWO realtime clocks!

Direct I2C Communication (Going Driverless)

You actually don't need a driver to use your I2C sensors. Drivers do make it much safer and easier, but if you really want full control, it is yours for the taking. There are symlinks for each I2C adapter to make finding them easy.

```
$ ls /dev/i2c-in*
/dev/i2c-in2  /dev/i2c-in3
```

Note

The symlinks and the underlying I2C device are only present when an I2C sensor is plugged into a port. Also, if a driver is loaded for a particular I2C device, you will get an error that it is in use. You should disable probing in the `nxt-i2c-sensor` module (or blacklist the driver in `/etc/modprobe.d`).

You can use the `i2c-tools` package or an I2C library in your programming language of choice to communicate with I2C devices this way. You don't want to do this if a device is already loaded so you will want to disable autodetection first if the sensor is the autodetected type. Beware that many sensors, including the NXT Ultrasonic Sensor use an address of 0x01, which is illegal according to the I2C standards. `i2c-tools` and any library that does some error checking may prevent you from accessing the sensor. In ev3dev-stretch, the `i2c-tools` package has been patched to work around this.

Practical examples

Changing the Polling Rate

Using the NXT Ultrasonic Sensor:

```
$ cat poll_ms
100
$ while true; do cat value0; done
22
23
26
27
30
25
...
22
24
26
26
22
22
^C
$ echo 1000 > poll_ms
$ while true; do cat value0; done
22
22
22
22
22
22
22
22
25
25
25
25
25
25
25
25
...
^C
$ echo 0 > poll_ms
$ cat value0 # value0 will be last value measured before polling stopped
23
$ cat value0 # move the sensor and try again
23
$ cat mode
[NXT-US-CM] NXT-US-IN NXT-US-SI-CM NXT-US-SI-IN NXT-US-LIST
$ echo NXT-US-CM > mode # reads data
$ cat value0
29
$ cat value0 # move the sensor and try again
29
```

`^C` means you have to press CTRL+C to make the loop stop.

Sample /etc/modprobe.d/nxt-i2c-sensor.conf

```
# Module configuration for nxt-i2c-sensor

# Uncomment this line to disable polling
#options nxt-i2c-sensor default_poll_ms=0

# Uncomment this line to disable autodetection
#options nxt-i2c-sensor allow_autodetect=N
```

How to find the I2C adapter node without adding 2

```
$ IN2_I2C_ADAP=$(udevadm info -q path -n /dev/i2c-in2)"/../.."
$ echo $IN2_I2C_ADAP
/devices/platform/legoev3-ports/in2/in2:nxt-i2c-host/i2c-legoev3.4/i2c-4/i2c-dev/i2c-4/../..
```

Using i2c-tools

With the mindsensors.com Realtime Clock Sensor on input port 2:

```
& i2cdump 4 0x68
No size specified (using byte-data access)
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-4, address 0x68, mode byte
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 11 35 00 01 01 01 00 03 50 71 48 60 f5 01 6b 0c    ?5.???.?PqH`??k?
10: 78 e3 2d 4e 92 6e c7 69 25 61 6b 5b 04 34 15 05    x?-N?n?i%ak[?4??
20: cc 3e 4e 4b 41 8a 59 09 1b f3 1a 2a 7c 47 a7 90    ?>NKA?Y????*|G??
30: 20 6a 95 7a 3b da 5b de 73 31 a2 3a 6e 59 ed f8     j?z;?[?s1?:nY??
40: 11 35 00 01 01 01 00 03 50 71 48 60 f5 01 6b 0c    ?5.???.?PqH`??k?
50: 78 e3 2d 4e 92 6e c7 69 25 61 6b 5b 04 34 15 05    x?-N?n?i%ak[?4??
60: cc 3e 4e 4b 41 8a 59 09 1b f3 1a 2a 7c 47 a7 90    ?>NKA?Y????*|G??
70: 20 6a 95 7a 3b da 5b de 73 31 a2 3a 6e 59 ed f8     j?z;?[?s1?:nY??
80: 11 35 00 01 01 01 00 03 50 71 48 60 f5 01 6b 0c    ?5.???.?PqH`??k?
90: 78 e3 2d 4e 92 6e c7 69 25 61 6b 5b 04 34 15 05    x?-N?n?i%ak[?4??
a0: cc 3e 4e 4b 41 8a 59 09 1b f3 1a 2a 7c 47 a7 90    ?>NKA?Y????*|G??
b0: 20 6a 95 7a 3b da 5b de 73 31 a2 3a 6e 59 ed f8     j?z;?[?s1?:nY??
c0: 12 35 00 01 01 01 00 03 50 71 48 60 f5 01 6b 0c    ?5.???.?PqH`??k?
d0: 78 e3 2d 4e 92 6e c7 69 25 61 6b 5b 04 34 15 05    x?-N?n?i%ak[?4??
e0: cc 3e 4e 4b 41 8a 59 09 1b f3 1a 2a 7c 47 a7 90    ?>NKA?Y????*|G??
f0: 20 6a 95 7a 3b da 5b de 73 31 a2 3a 6e 59 ed f8     j?z;?[?s1?:nY??
$ i2cget -y 4 0x68 0x01 | sed s/0x// # read minutes
35
$ i2cset -y 4 0x68 0x08 0x46 0x72 0x65 0x65 0x20 0x72 0x61 0x6d 0x20 0x73 0x70 0x61 0x63 0x65
0x21 i
$ i2cdump -y -r 0x08-0x16 4 0x68
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00:                         46 72 65 65 20 72 61 6d            Free ram
10: 20 73 70 61 63 65 21                                space!
```

Useful Info

Table of I2C addresses

Shifted Address (write/read)	Unshifted Address (hex (dec))	Notes
0x00/0x01	**0x00** (0)	*I2C spec: General call address / START byte*
0x02/0x03	**0x01** (1)	LEGO NXT Ultrasonic and many 3rd party sensors *I2C spec: CBUS address*
0x04/0x05	**0x02** (2)	LEGO Energy Storage *I2C spec: Reserved for different bus format*
0x06/0x07	**0x03** (3)	mindsensors.com Motor Multiplexer *I2C spec: Reserved for future purposes*

Shifted Address (write/read)	Unshifted Address (hex (dec))	Notes
0x08/0x09	**0x04** (4)	*I2C spec: Hs-mode master code*
0x0A/0x0B	**0x05** (5)	*I2C spec: Hs-mode master code*
0x0C/0x0D	**0x06** (6)	*I2C spec: Hs-mode master code*
0x0E/0x0F	**0x07** (7)	*I2C spec: Hs-mode master code*
0x10/0x11	**0x08** (8)	Some HiTechnic sensors
0x12/0x13	**0x09** (9)	
0x14/0x15	**0x0A** (10)	mindsensors.com Light Sensor Array
0x16/0x17	**0x0B** (11)	
0x18/0x19	**0x0C** (12)	mindsensors.com PPS58-Nx Pressure Sensor
0x1A/0x1B	**0x0D** (13)	
0x1C/0x1D	**0x0E** (14)	
0x1E/0x1F	**0x0F** (15)	
0x20/0x21	**0x10** (16)	
0x22/0x23	**0x11** (17)	mindsensors.com AbsoluteIMU Accel/Compass/Gyro
0x24/0x25	**0x12** (18)	
0x26/0x27	**0x13** (19)	
0x28/0x29	**0x14** (20)	
0x2A/0x2B	**0x15** (21)	
0x2C/0x2D	**0x16** (22)	
0x2E/0x2F	**0x17** (23)	
0x30/0x31	**0x18** (24)	mindsensors.com GlideWheel-AS Angle Sensor
0x32/0x33	**0x19** (25)	
0x34/0x35	**0x1A** (26)	
0x36/0x37	**0x1B** (27)	
0x38/0x39	**0x1C** (28)	
0x3A/0x3B	**0x1D** (29)	
0x3C/0x3D	**0x1E** (30)	
0x3E/0x3F	**0x1F** (31)	
0x40/0x41	**0x20** (32)	
0x42/0x43	**0x21** (33)	

Shifted Address (write/read)	Unshifted Address (hex (dec))	Notes
0x44/0x45	**0x22** (34)	
0x46/0x47	**0x23** (35)	
0x48/0x49	**0x24** (06)	
0x4A/0x4B	**0x25** (37)	
0x4C/0x4D	**0x26** (38)	
0x4E/0x4F	**0x27** (39)	
0x50/0x51	**0x28** (40)	
0x52/0x53	**0x29** (41)	
0x54/0x55	**0x2A** (32)	
0x56/0x57	**0x2B** (43)	
0x58/0x59	**0x2C** (44)	
0x5A/0x5B	**0x2D** (45)	
0x5C/0x5D	**0x2E** (46)	
0x5E/0x5F	**0x2F** (47)	
0x60/0x61	**0x30** (48)	
0x62/0x63	**0x31** (49)	
0x64/0x65	**0x32** (50)	
0x66/0x67	**0x33** (51)	
0x68/0x69	**0x34** (52)	
0x6A/0x6B	**0x35** (53)	
0x6C/0x6D	**0x36** (54)	
0x6E/0x6F	**0x37** (55)	
0x70/0x71	**0x38** (56)	PCF8574 IC
0x72/0x73	**0x39** (57)	
0x74/0x75	**0x3A** (58)	
0x76/0x77	**0x3B** (59)	
0x78/0x79	**0x3C** (60)	
0x7A/0x7B	**0x3D** (61)	
0x7C/0x7D	**0x3E** (62)	
0x7E/0x7F	**0x3F** (63)	

Shifted Address (write/read)	Unshifted Address (hex (dec))	Notes
0x80/0x81	**0x40** (64)	
0x82/0x83	**0x41** (65)	
0x84/0x85	**0x42** (66)	
0x86/0x87	**0x43** (67)	
0x88/0x89	**0x44** (68)	
0x8A/0x8B	**0x45** (69)	
0x8C/0x8D	**0x46** (70)	
0x8E/0x8F	**0x47** (71)	
0x90/0x91	**0x48** (72)	PCF8591 IC
0x92/0x93	**0x49** (73)	
0x94/0x95	**0x4A** (74)	
0x96/0x97	**0x4B** (75)	
0x98/0x99	**0x4C** (76)	LEGO Temperature Sensor
0x9A/0x9B	**0x4D** (77)	
0x9C/0x9D	**0x4E** (78)	
0x9E/0x0F	**0x4F** (79)	
0xA0/0xA1	**0x50** (80)	mindsensors.com EV3 Sensor Multiplexer
0xA2/0xA3	**0x51** (81)	mindsensors.com EV3 Sensor Multiplexer
0xA4/0xA5	**0x52** (82)	mindsensors.com EV3 Sensor Multiplexer
0xA6/0xA7	**0x53** (83)	
0xA8/0xA9	**0x54** (84)	
0xAA/0xAB	**0x55** (85)	
0xAC/0xAD	**0x56** (87)	
0xAE/0xAF	**0x57** (87)	
0xB0/0xB1	**0x58** (88)	mindsensors.com 8 Channel Servo Controller
0xB2/0xB3	**0x59** (89)	
0xB4/0xB5	**0x5A** (90)	
0xB6/0xB7	**0x5B** (91)	
0xB8/0xB9	**0x5C** (92)	
0xBA/0xBB	**0x5D** (93)	

Shifted Address (write/read)	Unshifted Address (hex (dec))	Notes
0xBC/0xBD	**0x5E** (94)	
0xBE/0xBF	**0x5F** (95)	
0xC0/0xC1	**0x60** (96)	
0xC2/0xC3	**0x61** (97)	
0xC4/0xC5	**0x62** (98)	
0xC6/0xC7	**0x63** (99)	
0xC8/0xC9	**0x64** (100)	
0xCA/0xCB	**0x65** (101)	
0xCC/0xCD	**0x66** (102)	
0xCE/0xCF	**0x67** (103)	
0xD0/0xD1	**0x68** (104)	mindsensors.com Realtime Clock
0xD2/0xD3	**0x69** (105)	
0xD4/0xD5	**0x6A** (106)	
0xD6/0xD7	**0x6B** (107)	
0xD8/0xD9	**0x6C** (108)	
0xDA/0xDA	**0x6D** (109)	
0xDC/0xDD	**0x6E** (110)	
0xDE/0xDF	**0x6F** (111)	
0xE0/0xE1	**0x70** (112)	
0xE2/0xE3	**0x71** (113)	
0xE4/0xE5	**0x72** (114)	
0xE6/0xE7	**0x73** (115)	
0xE8/0xE9	**0x74** (116)	
0xEA/0xEB	**0x75** (117)	
0xEC/0xED	**0x76** (118)	
0xEE/0xEF	**0x77** (119)	
0xF0/0xF1	**0x78** (120)	*I2C spec: 10-bit slave addressing*
0xF2/0xF3	**0x79** (121)	*I2C spec: 10-bit slave addressing*
0xF4/0xF5	**0x7A** (122)	*I2C spec: 10-bit slave addressing*
0xF6/0xF7	**0x7B** (123)	*I2C spec: 10-bit slave addressing*

Shifted Address (write/read)	Unshifted Address (hex (dec))	Notes
0xF8/0xF9	**0x7C** (124)	*I2C spec: Reserved for future purposes*
0xFA/0xFB	**0x7D** (125)	*I2C spec: Reserved for future purposes*
0xFC/0xFD	**0x7E** (126)	*I2C spec: Reserved for future purposes*