**[Raspberry Pi Resources - Using the I2C interface](https://raspberry-projects.com/pi/programming-in-c/i2c/using-the-i2c-interface)**

/[Programming in C/C++](https://raspberry-projects.com/pi/category/programming-in-c) / [I2C](https://raspberry-projects.com/pi/category/programming-in-c/i2c) / Using the I2C interface

**Enabling The I2C Port**

The I2C port needs to be enabled in Rasbian before it can be used. See [here](https://raspberry-projects.com/pi/pi-operating-systems/raspbian/io-pins-raspbian/i2c-pins).

**Checking For Connected Devices**

Install the I2C tools (i2c-tools is a set of I²C programs that make it easy to debug I²C devices without having to write any code):

sudo apt-get install i2c-tools

sudo apt-get update

At the command prompt type one of these depending on whether you are using the I2C0 or I2C1 port:

sudo i2cdetect -y 0

//or

sudo i2cdetect -y 1

The 7 bit I2C address of all found devices will be shown (ignoring the R/W bit, so I2C address 0000 0110 is displayed as hex 03).

**Useful Resources**

<http://www.skpang.co.uk/blog/archives/575>

<http://www.hobbytronics.co.uk/raspberry-pi-raspbian-distro>

**Reading And Writing I2C**

#include <unistd.h> //Needed for I2C port

#include <fcntl.h> //Needed for I2C port

#include <sys/ioctl.h> //Needed for I2C port

#include <linux/i2c-dev.h> //Needed for I2C port

int file\_i2c;

int length;

unsigned char buffer[60] = {0};

//----- OPEN THE I2C BUS -----

char \*filename = (char\*)"/dev/i2c-1";

if ((file\_i2c = open(filename, O\_RDWR)) < 0)

{

//ERROR HANDLING: you can check errno to see what went wrong

printf("Failed to open the i2c bus");

return;

}

int addr = 0x5a; //<<<<<The I2C address of the slave

if (ioctl(file\_i2c, I2C\_SLAVE, addr) < 0)

{

printf("Failed to acquire bus access and/or talk to slave.\n");

//ERROR HANDLING; you can check errno to see what went wrong

return;

}

//----- READ BYTES -----

length = 4; //<<< Number of bytes to read

if (read(file\_i2c, buffer, length) != length) //read() returns the number of bytes actually read, if it doesn't match then an error occurred (e.g. no response from the device)

{

//ERROR HANDLING: i2c transaction failed

printf("Failed to read from the i2c bus.\n");

}

else

{

printf("Data read: %s\n", buffer);

}

//----- WRITE BYTES -----

buffer[0] = 0x01;

buffer[1] = 0x02;

length = 2; //<<< Number of bytes to write

if (write(file\_i2c, buffer, length) != length) //write() returns the number of bytes actually written, if it doesn't match then an error occurred (e.g. no response from the device)

{

/\* ERROR HANDLING: i2c transaction failed \*/

printf("Failed to write to the i2c bus.\n");

}