Saturday, March 2, 2013

Programming I2C

Although you can perform simple i2c reads and writes using the command line tools `i2cget` and `i2cset`, for a more integrated approach you can use a programming language to talk to the bus.

The are dozens of languages which make claims about ease of use and learning etc. and I am sure you can program i2c from them.

What I will demonstrate here is the simple way to do it from c. Although I don't aim to teach how to program in c, I will try and explain what the code is doing so you can follow along even if you are new to c.

This will use some basic i2c read and writes as described at http://www.kernel.org/doc/Documentation/i2c/dev-interface
We will also need to perform some IO Control (ioctl) which are i2c specific.

First we need some code to get us started. The `#include` basicly make certain function calls and constants available to the rest of our program.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <linux/i2c-dev.h>
```

All of our code will live in the main function for now. `main()` is where all c programs start.

We need some variables which must be declared at the start of the function.

```
int main( int argc, char **argv )
{
        int i;
        int r;
        int fd;
        unsigned char command[2];
        unsigned char value[4];
        useconds_t delay = 2000;

        char *dev = "/dev/i2c-1";
        int addr = 0x48;
```

The `[ ]` syntax means an array so `char command[2]` is actually a variable which can hold 2 char values.

Some of our variables have been initialised to specific values so they are ready to use. The ones which have not been initialised will contain random values so we must assign a value to them before they can be used.

`0x48` means hexadecimal 48 (which is decimal 72).

Next we print out a banner to show that the program is running

```
printf("PCF8591 Test\n");
```

The we get down to business and open the i2c device

```
        fd = open(dev, O_RDWR );
        if(fd < 0)
        {
                perror("Opening i2c device node\n");
                return 1;
        }
```

and select our slave device

```
        r = ioctl(fd, I2C_SLAVE, addr);
        if(r < 0)
        {
                perror("Selecting i2c device\n");
        }
```

Now we have an infinite loop

```
         while(1)
        {
```

There will be no way to end the program except by pressing Control C.

Next we have another loop which will run four times

```
                for(i = 0; i < 4; i++)
                {
```

Then we build a command for the pcf8591. The value of this is specified in the data sheet http://doc.chipfind.ru/pdf/philips/pca8591.pdf

In the first 8 bits of the command we will enable the analog output bit (`0x40`) and select which of the 4 inputs to read (`(i + 1) & 0x03`). We do a bitwise or to combine these values together with the `|` symbol.

```
command[0] = 0x40 | ((i + 1) & 0x03); // output enable | read input i
```

The `//` is the start of a comment so you can explain you code to the reader.

In the next 8 bits we increment the value for the analog output

```
command[1]++;
```

Now we are ready to send the command to the i2c bus

```
r = write(fd, &command, 2);
```

It is not clear why, but we need to wait for the command to be processed

```
usleep(delay);
```

Now we are ready to read a value. Remembering that the read is always one value behind the selected input (hence the +1 we used above).

```
r = read(fd, &value[i], 1);
if(r != 1)
{
        perror("reading i2c device\n");
}
usleep(delay);
```

Then we end the loop

```
        }
```

and now we can print out our results

```
        printf("0x%02x 0x%02x 0x%02x 0x%02x\n", value[0], value[1],
value[2], value[3]);
```

end our infinite loop

```
    }
```

and although we may never reach here, we will clean up and quit.

```
  close(fd);
  return(0);
}
```

Now, if you enter all the code into a file called `pcf8591d.c` (you can copy the complete code as show below) then you are ready to compile it with this command

```
gcc -Wall -o pcf8591d pcf8591d.c
```

This says to compile the `.c` file and write the output (`-o`) to `pcf8591d` (if you don't specify an output file the default of `a.out` will be used which can be a but confusing). `-Wall` will make sure all warnings are printed out by the compiler.

Assuming the compile (and link) was successful you are ready to run

```
./pcf8591d
```

and you should see output like this:

```
PCF8591 Test
0x5f 0xd3 0xac 0x80
0xc1 0xd3 0xae 0x80
0xc1 0xd3 0xb0 0x80
0xc1 0xd3 0xb2 0x80
0xc1 0xd3 0xb7 0x80
0xc1 0xd3 0xba 0x80
0xc1 0xd3 0xde 0x80
0xc1 0xd3 0xdc 0x80
0xc1 0xd3 0xe0 0x80
```

To stop the program press ^c (Control + C).

Here is the complete source code:

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <linux/i2c-dev.h>



// Written by John Newbigin jnewbigin@chrysocome.net
// Based on info from http://www.kernel.org/doc/Documentation/i2c/dev-interface
// And http://doc.chipfind.ru/pdf/philips/pca8591.pdf

int main( int argc, char **argv )
{
        int i;
        int r;
        int fd;
        unsigned char command[2];
        unsigned char value[4];
        useconds_t delay = 2000;

        char *dev = "/dev/i2c-1";
        int addr = 0x48;
        printf("PCF8591 Test\n");

        fd = open(dev, O_RDWR );
        if(fd < 0)
        {
                perror("Opening i2c device node\n");
                return 1;
        }

        r = ioctl(fd, I2C_SLAVE, addr);
        if(r < 0)
        {
                perror("Selecting i2c device\n");
        }

        while(1)
        {
                for(i = 0; i < 4; i++)
                {
                        command[0] = 0x40 | ((i + 1) & 0x03); // output enable | read input i
                        command[1]++;
                        r = write(fd, &command, 2);
                        usleep(delay);
                        // the read is always one step behind the selected input
                        r = read(fd, &value[i], 1);
                        if(r != 1)
                        {
                                perror("reading i2c device\n");
                        }
                        usleep(delay);
                }
                printf("0x%02x 0x%02x 0x%02x 0x%02x\n", value[0], value[1], value[2],
value[3]);
        }

        close(fd);
        return(0);
}
```

In the next blog I will improve the output to print a graph of the values so you can see them move up and down.

Posted by John Newbigin at 4:40 PM

Labels: Code, Hardware hacking, Raspberry Pi