Introduction

Objectives

The objective of this tutorial is to complete the ADXL345 device driver started in the previous tutorial and to add the possibility for the user space to interact with it via the misc framework.

Pre-requisites (Make sure it is ready before/in lab)

Environment of Practical session

We will use the directory created in the previous tutorial.

```
$ export TPROOT=xxx  # to adapt
$ cd $TPROOT
$ cd pilote_i2c
```

Remember to adapt the above commands according to the name given to the directory during the first practice.

As a reminder, the link to the documentation of the accelerometer :

Documentation of ADXL345

Work to Do!!
First Step (Assignment):

In this first step, we will just register with the misc framework.

1. Declare a struct adxl345_device structure containing for the moment a single struct miscdevice field

2. In the probe function

    - Dynamically allocate memory for an instance of the struct adxl345_device
    - Associate this instance with the struct i2c_client
    - Fill the content of the struct miscdevice structure contained in the instance of the struct adxl345_device structure (see in the course the list of important fields to initialize in this structure)

    Note about the name field: this string will be used as a name to automatically create a special file in /dev. As several accelerometers may be present, this name must be unique for each accelerometer. We will use a name like adxl345-x where x is the number of the accelerometer (e.g. adxl345-0, adxl345-1...). To generate this name, we will maintain a global variable indicating the number of accelerometers present (variable incremented with each call to the probe function and decremented with each call to the remove function), then in probe, we will use the kasprintf function (seen in progress) to generate the appropriate character string for the name field. Remember to free the memory allocated by kasprintf but not too soon!

    - Register with the misc framework

3. In the remove function
    - Retrieve the instance of the struct adxl345_device from the struct i2c_client retrieved as argument
    - Unregister from the misc framework

4. Compile and test your driver (check that the special file appears in /dev after loading your driver)

Second Step:

In this second step, we will implement the read callback function to allow an application to retrieve data from the application.

We will first assume that this function returns the data of the X axis of the accelerometer. To simplify, if the application requests one byte, we will return the high byte of the last sample, and if the application requests more than one byte, we will return the full value of one sample (16 bits) but no more (so even if the application requests 6 bytes, our function will only provide 2).

1. Write the function adxl345_read
- From the struct file structure retrieved as argument, retrieve the instance of the struct adxl345_device structure and the instance of the struct i2c_client structure corresponding to your device

- Get a sample from the accelerometer
- Pass all or part of this sample to the application

2. In the probe function, declare this function adxl345_read
3. Compile and load your driver
4. Write a small application in C to test

- Write the application in C on your PC (take the opportunity to consult the man page of open and read via the man 2 open and man 2 read command to see the arguments to pass to these system calls)
- Compile this application with the command arm-linux-gnueabihf-gcc -Wall -o main main.c (assuming that the source file is called main.c)
- Put the file main in your driver_i2c directory
- Restart QEMU. After mounting, you will find your application in /mnt/main

Third Step:

We now want to allow the application to choose the axis (X, Y or Z) it wants to retrieve, thanks to the ioctl system call.

To do this, read carefully the kernel documentation (file Documentation/driver-api/ioctl.rst) in order to respect the conventions around ioctl.

At first, we will accept that the modification of the chosen axis impacts all the applications that use the accelerometer in question. Then we will think about how to make sure that this change only impacts the application that requested it.

It's up to you!