vient donc de préciser l'adresse de l'esclave sur le bus I2C. La commande *ioctl* est une sorte de fourre-tout du driver qui nous sert ici à préciser cette adresse de destination,

- *Write* : afin de transmettre des données au périphérique I2C, il faut les envoyer au driver au travers de l'opération *write*,
- *Read* : afin de lire des données en provenance d'un périphérique I2C, il faut le demander au driver au travers de l'opération *read*.

Si vous souhaitez savoir comment utiliser ces opérations et comment les coder en langage C, nous vous invitons à consulter un ouvrage qui traite des accès aux fichiers en C, ou tout simplement en tapant dans un shell la commande `man open` pour la commande `open` par exemple.

Chargement du driver

Une fois généré, le driver devra être transféré sur la cible puis chargé en mémoire. Pour éviter de refaire cette opération après chaque démarrage de la carte nous allons le sauvegarder en mémoire NAND et modifier le script de démarrage. La compilation du driver a permis de copier le fichier sous le répertoire */tftpboot* de la station. Nous allons pouvoir transférer le fichier à l'aide du programme *tftpget*. Nous pourrions aussi utiliser FTP…

Chargez votre module à l'aide de la commande `insmod i2c_s3c44.o`. Effectuez à nouveau un `lsmod` pour constater que le noyau à terminé le chargement du module. Ensuite, vous pouvez si vous le souhaitez décharger le driver à l'aide de la commande `rmmod i2c_s3c44`, puis le recharger à l'aide de la commande `insmod i2c_s3c44.o`.

Création d'une application

Nous allons à présent coder une application qui utilise le driver i2c préalablement chargé. Cette application configurera les 16 bits du MCP23016 en sortie afin d'afficher sur les 2 afficheurs 7 segments des valeurs numériques.

Codage de l'application

Un exemple d'utilisation du driver *i2c_s3c44.o* au travers de l'interface */dev/i2c0* se trouve dans le Listing 1.

L'exécution du code se déroule en 3 temps :

- analyse des paramètres , ouverture de l'interface spécifiée (*/dev/i2c0*) et indication

Listing 1. Utilisation du driver i2c_s3c44

```
/* Exemple I2C sur S3C44B0X */
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <linux/i2c.h>
/* Valeur en hexa à appliquer aux 7 segments pour afficher un chiffre */
unsigned char tab[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8,
0x80, 0x90};
/* accessMCP /dev/i2c0 42 */
int main(int argc, char**argv) {
   unsigned char data[10];
   int fd, ret, i, j, n;
   char i2c_device[255] ;
   if (argc < 3) {
      printf("Usage : accessMCP [DEVICE] [Addr]hex\n");
      printf("Ex : accessMCP /dev/i2c0 42\n");
      return (-1);
   }
   strcpy(i2c_dev, argv[1]) ;
   if ((fd = open(i2c_device,O_RDWR)) < 0 ) {
      perror("open");
      printf("Error opening %s\n", i2c_device);
      return (-1);
   }
   printf("\n--- Test d'ecriture I2C ---\n\n");
   sscanf(argv[2], "%x", &n);
   ret = ioctl(fd, I2C_SLAVE, n);
   if (ret <0) {
      perror("I2C_SLAVE ioctl cmd");
      close(fd);
      return (-1);
   }
   /* Paramétrage du MCP23016 : 16 bits en sortie */
   data[0] = 0x06; data[1] = 0x00; write(fd, &data[0], 2); // IODIR0
   data[0] = 0x07; data[1] = 0x00; write(fd, &data[0], 2); // IODIR1
   data[0] = 0x02; data[1] = 0xff; write(fd, &data[0], 2); // IOLAT0
   data[0] = 0x03; data[1] = 0xff; write(fd, &data[0], 2); // IOLAT0
   for (i = 0, j= 0; i < sizeof(tab); i++) {
      data[0] = 0x02;
      data[1] = tab[i];
// printf("Mise a ON des segments (@MCP23016:0x%x / Reg= 0x%02x /
//  Data= 0x%02x)\n", n, data[0], data[1]);
      write(fd, &data[0], 2);
      if (i == 9) {
         i = 0;
         j++;
         data[0] = 0x03;
         data[1] = tab[j];
         write(fd, &data[0], 2);
      }
      if ((i == 6) && (j == 4)) break;
      usleep(50000);
   }
   close(fd);
   return 0;
}
```