Introduction

Lien utilisateur-noyau

/sys/class

MMU

Datasheet

Informatique embarquée 10/16

J.-M Friedt

FEMTO-ST/département temps-fréquence

jmfriedt@femto-st.fr

transparents à `jmfriedt.free.fr`

8 mars 2018

Buildroot

Environnement cohérent pour fournir

- une chaîne de cross-compilation
- un noyau linux
- des programmes en espace utilisateur
- un bootloader (initialisation CPU + chargement noyau)

pour Redpitaya (Zynq) : 6 GB d'espace disque pour une images de 200+ MB dans `output/images/sdcard.img`

Accès au port série (matériel)

```
jmfriedt@dhcp-221:~$ ls -l /dev/ttyS*
crw-rw---- 1 root dialout 4, 64 Oct  8 18:43 /dev/ttyS0
crw-rw---- 1 root dialout 4, 65 Oct  8 18:43 /dev/ttyS1
crw-rw---- 1 root dialout 4, 66 Oct  8 18:43 /dev/ttyS2
crw-rw---- 1 root dialout 4, 67 Oct  8 18:43 /dev/ttyS3
```

Exemple de l'accès au port série :

```
int fd;
struct termios oldtio,newtio;
fd=open("/dev/ttyS0", O_RDWR | O_NOCTTY );
tcgetattr(fd,&oldtio); /* save current serial port settings */
newtio.c_cflag = BAUDRATE | CS8 | CLOCAL | CREAD;  /* _no_ CRTSCTS */
tcsetattr(fd,TCSANOW,&newtio);
```

puis

```
unsigned char cmd;
read(fd,&cmd,1);
printf"(%x) ",(cmd&0xff));fflush(stdout);
```

Outil pour transférer des données sur le port série sous GNU/Linux :

`stty -F /dev/ttyS0 9600 && cat < /dev/ttyS0` ou `minicom`

Accès au matériel depuis l'espace utilisateur

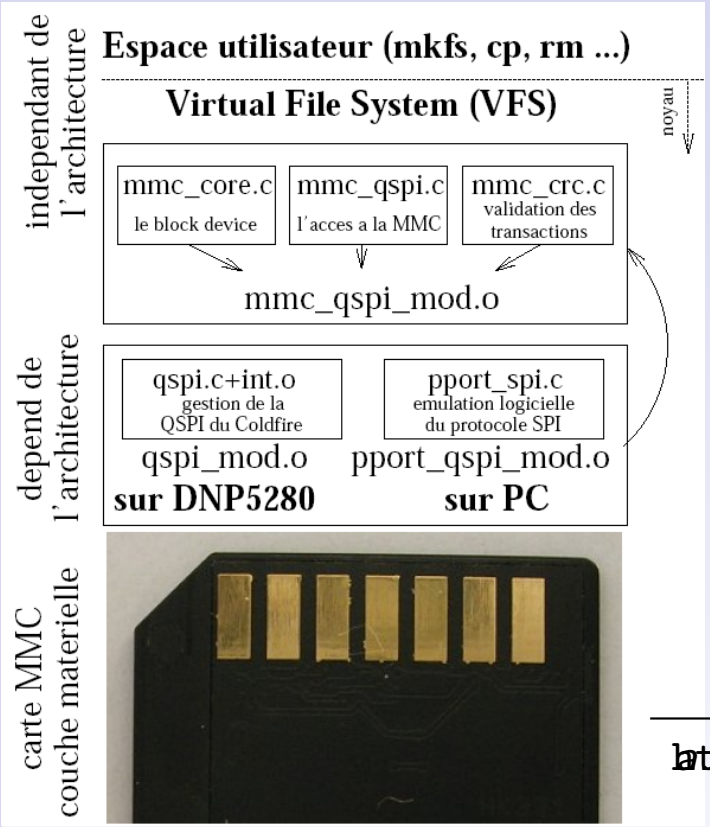
Au travers de `/dev`

- écriture, lecture ou contrôle (`ioctl`)
- passage au travers du module noyau qui implémente les diverses méthodes
- chaque périphérique est identifié par sa classe (*major number*) et son indice (*minor number*)

```
brw-rw----  1 root     disk      8,   0 Feb 28 06:21 sda
brw-rw----  1 root     disk      8,   1 Feb 28 06:21 sda1
brw-rw----  1 root     disk      8,   2 Feb 28 06:21 sda2
brw-rw----  1 root     disk      8,   3 Feb 28 06:21 sda3
[...]
crw-rw----  1 root     dialout   4,  64 Feb 28 07:21 ttyS0
crw-rw----  1 root     dialout   4,  65 Feb 28 07:21 ttyS1
crw-rw----  1 root     dialout   4,  66 Feb 28 07:21 ttyS2
crw-rw----  1 root     dialout   4,  67 Feb 28 07:21 ttyS3
```

En l'absence d'une entrée (par `udev`) dans `/dev` : `mknod`

Accès au matériel depuis l'espace utilisateur [1] [2]



- `block` (b) device ("fichier") se comporte *du point de vue utilisateur* comme `char` (c) device (*pipe*) [a]
- les échanges avec le matériel se font par blocs de données (tampon, cache) au lieu d'octet par octet
- remplacer `file_operation` par `block_operation`
- les transferts sont régis selon une "géométrie" de disque définie dans une structure `gendisk`

a `http://www.makelinux.net/ldd3/chp-16-sect-1`

1. P. Ficheux, *Programmation noyau sous Linux : les pilotes en mode bloc*, GNU/Linux Magazine France, 109, pp.4-10 (Octobre 2008)
2. S. Guinot, J.-M Friedt, *Stockage de masse non-volatile : un block device pour MultiMediaCard*, GNU/Linux Magazine France, Hors Série 25 (Avril 2006)

Accès au matériel depuis l'espace utilisateur

Au travers de `/sys/class`

- échange d'informations en trames ASCII
- configuration au travers du fichier approprié (pas de `ioctl`)
- particulièrement approprié pour une interaction avec l'utilisateur (programmation *shell*)

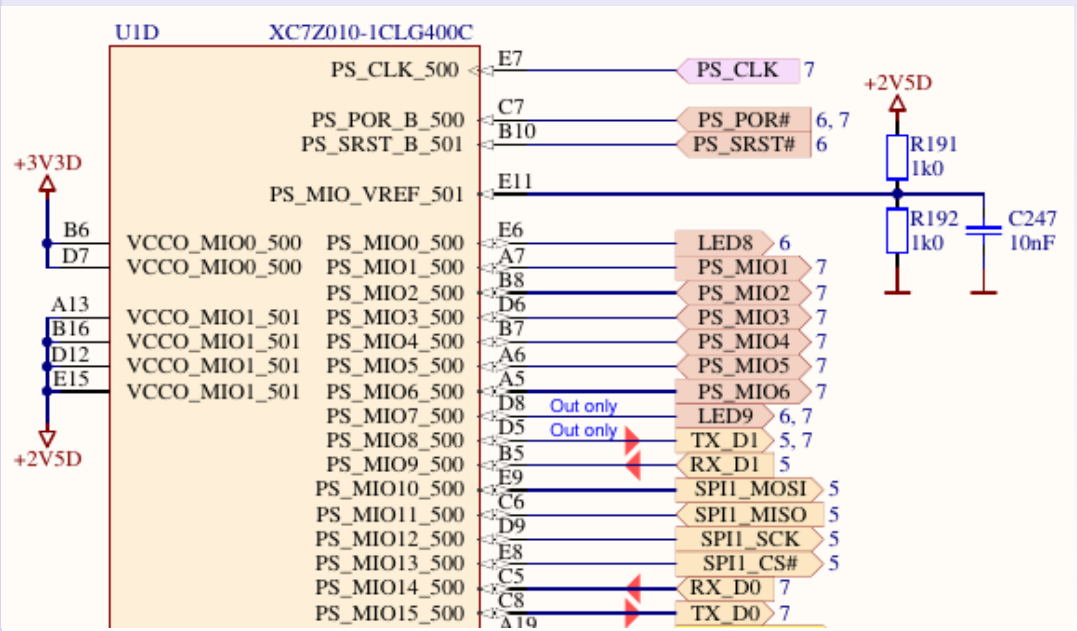
```
$ cat /sys/class/rtc/rtc0/time
07:37:02
# cat /sys/class/backlight/intel_backlight/max_brightness
4500
# echo "1000" >  /sys/class/backlight/intel_backlight/brightness
```

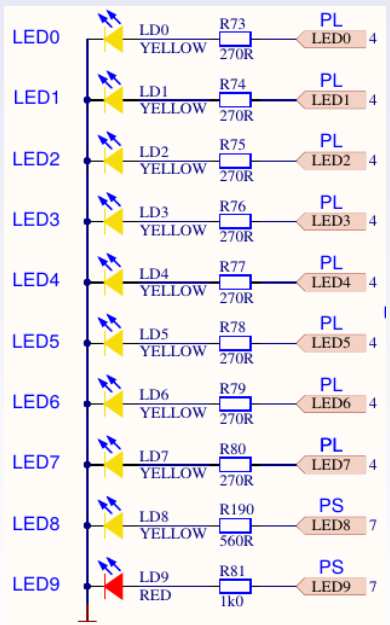
Attention en C : `fseek` à l'offset 0 en `SEEK_SET` pour plusieurs accès sans refaire `fopen` et `fclose`

Accès au matériel depuis l'espace utilisateur

Au travers de `/sys/class` : cas de `/sys/class/gpio`

- vérifier que le module est chargé : `gpio_zynq`
- un GPIO est défini par son indice 906+MIO
- demander l'autorisation d'accès à un périphérique : `export`
- placer la broche en entrée ou en sortie
- définir son état
- *attention aux autorisations d'accès*





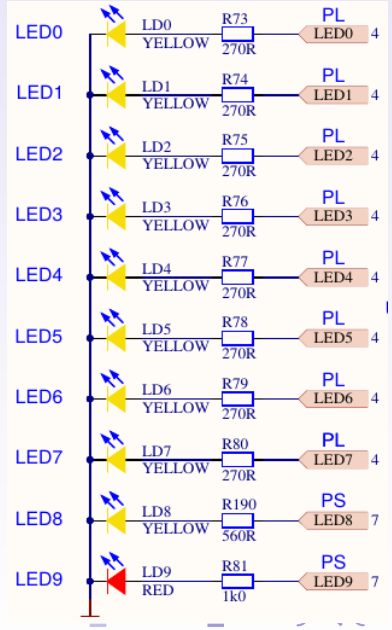
Accès au matériel depuis l'espace utilisateur

Au travers de `/sys/class` : cas de `/sys/class/gpio`

- vérifier que le module est chargé : `gpio_zynq`
- un GPIO est défini par son indice 906+MIO
- demander l'autorisation d'accès à un périphérique : `export`
- placer la broche en entrée ou en sortie
- définir son état
- *attention aux autorisations d'accès*

The default pin assignment for GPIO is described in the next table.

FPGA	con-nector	GPIO	MIO/EMIO index	`sysfs` index	color, dedicated meaning
		`exp_p_io [7:0]`	`EMIO[15: 8]`	`906+54+[15: 8]=[975:968]`	
		`exp_n_io [7:0]`	`EMIO[23:16]`	`906+54+[23:16]=[983:976]`	
		LED `[7:0]`	`EMIO[ 7: 0]`	`906+54+[ 7: 0]=[967:960]`	yellow
		LED " [8]"	`MIO[ 0]`	`906+ [ 0] = 906`	yellow = CPU heartbeat (user defined)
		LED " [9]"	`MIO[ 7]`	`906+ [ 7] = 913`	red = SD card access (user defined)



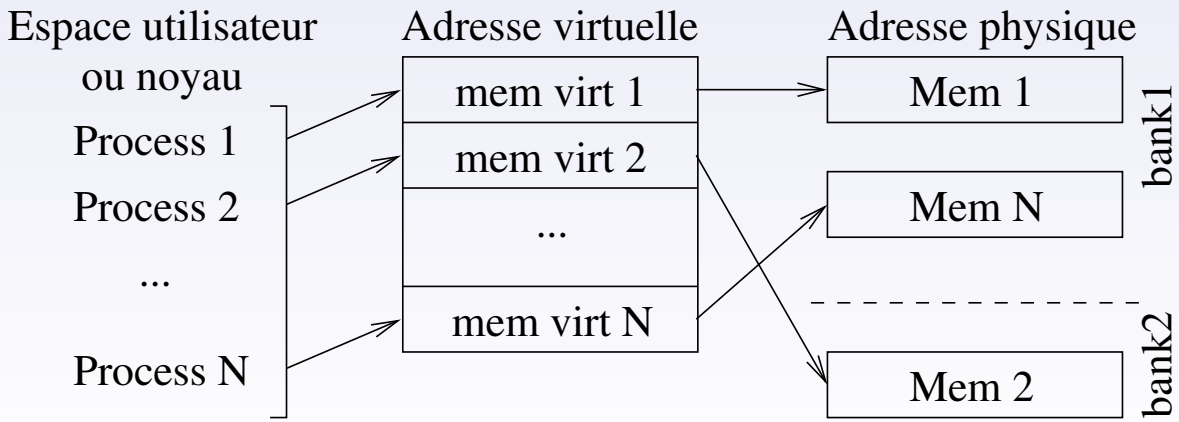
Mémoire virtuelle/mémoire matérielle

Mémoire matérielle

- mémoire matérielle : une adresse sur le bus d'adresse pour identifier un périphérique
- chaque périphérique décode le bus d'adresse pour savoir si le message lui est destiné
- un seul périphérique par adresse physique (sinon, conflit)

Mémoire virtuelle

- chaque processus a sa plage d'adresses
- organisation de la mémoire indépendante des contraintes physiques
- MMU : traduction entre mémoire matérielle et virtuelle



Accès au matériel depuis l'espace utilisateur

Au travers de `/dev/mem`

- avantage : ne pas passer par le noyau (rapide)
- inconvénient : ne pas passer par le noyau (pas de gestion de l'accès aux ressources)

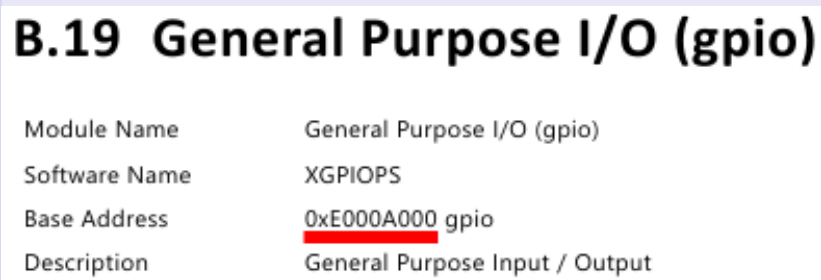
```
#include <fcntl.h>
#include <sys/mman.h>
#define MAP_SIZE 4096UL
#define MAP_MASK (MAP_SIZE - 1)

int main(int argc, char **argv) {
 int fd; void *map_base, *virt_addr; unsigned long read_result, writeval;
 off_t target;
 int access_type = 'w'; // Args : { address } [ type [ data ] ]
 target = strtoul(argv[1], 0, 0);
 if(argc > 2) access_type = tolower(argv[2][0]);
 fd = open("/dev/mem", O_RDWR | O_SYNC);
 map_base = mmap(0, MAP_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, target & ~MAP_MASK);
 virt_addr = map_base + (target & MAP_MASK);
 switch(access_type) { case 'b': read_result = *((unsigned char *) virt_addr); break;
                       case 'h': read_result = *((unsigned short *)virt_addr); break;
                       case 'w': read_result = *((unsigned long *) virt_addr); break;
 }
 printf("Value at address 0x%X (%p): 0x%X\n", target, virt_addr, read_result);
 if(argc > 3) {
  writeval = strtoul(argv[3], 0, 0);
  switch(access_type) { case 'b': *((unsigned char *) virt_addr) = writeval; break;
                        case 'h': *((unsigned short *)virt_addr) = writeval; break;
                        case 'w': *((unsigned long *) virt_addr) = writeval; break;
              }       }
 munmap(map_base, MAP_SIZE); close(fd); return 0;
}
```

Exemple : `busybox-1.27.1/miscutils/devmem.c`

Description des registres

Datasheet du Zynq 70xx [3] chapitre 14 (GPIO)

- 4 banques de GPIO (32 bits)
- offset des registres de contrôle des GPIO par rapport à une adresse de base 0xEA00A000
- description de la procédure d'activation d'un GPIO en 14.3 (*Programming Guide*)
- description de chaque registre dans l'appendice B



B.19 General Purpose I/O (gpio)

Module Name	General Purpose I/O (gpio)
Software Name	XGPIOPS
Base Address	0xE000A000 gpio
Description	General Purpose Input / Output

- comme sur les microcontrôleurs : enable, output, data

3. www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf

Description des registres

Attention : l'**horloge** de la banque de GPIO doit être activée

- les configurations globales sont en 0xF8000000 (*System Level Control Registers*)

14.3.1 Start-up Sequence

Main Example: Start-up Sequence

1. **Resets:** The reset options are described in section 14.4.2 Resets.
2. **Clocks:** The clocks are described in section 14.4.1 Clocks.
3. **GPIO Pin Configurations:** Configure pin as input/output is described in section 14.3.2 GPIO Pin Configurations.
4. **Write Data to GPIO Output pin:** Refer to example in section 14.3.3 Writing Data to GPIO Output Pins.
5. **Read Data from GPIO Input pin:** Refer to example in section 14.3.4 Reading Data from GPIO Input Pins .
6. **Set GPIO pin as wake-up event**: Refer to example in section GPIO as Wake-up Event.

14.3.2 GPIO Pin Configurations

Each individual GPIO pin can be configured as input/output. However, bank0 [8:7] pins must be configured as outputs. Refer to section 14.2.3 Bank0, Bits[8:7] are Outputs for further details.

Example: Configure MIO pin 10 as an output

1. **Set the direction as output:** Write `0x0000_0400` to the gpio.DIRM_0 register.

Register (slcr) APER_CLK_CTRL

Name	APER_CLK_CTRL
Relative Address	0x0000012C
Absolute Address	0xF800012C
Width	32 bits
Access Type	rw
Reset Value	0x01FFCCCD
Description	AMBA Peripheral Clock Control

Register APER_CLK_CTRL Details

Please note that these clocks must be enabled if you want to read from the peripheral register space.

Field Name	Bits	Type	Reset Value	Description
reserved	31:25	rw	0x0	Reserved. Writes are ignored, read data is zero.
SMC_CPU_1XCLKACT	24	rw	0x1	SMC AMBA Clock control 0: disable, 1: enable
LQSPI_CPU_1XCLKACT	23	rw	0x1	Quad SPI AMBA Clock control 0: disable, 1: enable
GPIO_CPU_1XCLKACT	22	rw	0x1	GPIO AMBA Clock control 0: disable, 1: enable

- en cas d'échec de la configuration, vérifier le vérou sur les registres

Description des registres

Attention : l'**horloge** de la banque de GPIO doit être activée

- les configurations globales sont en 0xF8000000 (*System Level Control Registers*)

Register SLCR_LOCK Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	wo	0x0	Reserved. Writes are ignored, read data is zero.
LOCK_KEY	15:0	wo	0x0	Write the lock key, 0x767B, to write protect the slcr registers: all slcr registers, 0xF800_0000 to 0xF800_0B74, are write protected until the unlock key is written to the SLCR_UNLOCK register. A read of this register returns zero.

Register (slcr) SLCR_UNLOCK

Name	SLCR_UNLOCK
Relative Address	0x00000008
Absolute Address	0xF8000008
Width	32 bits
Access Type	wo
Reset Value	0x00000000
Description	SLCR Write Protection Unlock

Register SLCR_UNLOCK Details

Field Name	Bits	Type	Reset Value	Description
reserved	31:16	wo	0x0	Reserved. Writes are ignored, read data is zero.
UNLOCK_KEY	15:0	wo	0x0	Write the unlock key, 0xDF0D, to enable writes to the slcr registers. All slcr registers, 0xF800_0000 to 0xF800_0B74, are writeable until locked using the SLCR_LOCK register. A read of this register returns zero.

Register (slcr) APER_CLK_CTRL

Name	APER_CLK_CTRL
Relative Address	0x0000012C
Absolute Address	0xF800012C
Width	32 bits
Access Type	rw
Reset Value	0x01FFCCCD
Description	AMBA Peripheral Clock Control

Register APER_CLK_CTRL Details

Please note that these clocks must be enabled if you want to read from the peripheral register space.

Field Name	Bits	Type	Reset Value	Description
reserved	31:25	rw	0x0	Reserved. Writes are ignored, read data is zero.
SMC_CPU_1XCLKACT	24	rw	0x1	SMC AMBA Clock control 0: disable, 1: enable
LQSPI_CPU_1XCLKACT	23	rw	0x1	Quad SPI AMBA Clock control 0: disable, 1: enable
GPIO_CPU_1XCLKACT	22	rw	0x1	GPIO AMBA Clock control 0: disable, 1: enable

- en cas d'échec de la configuration, vérifier le vérou sur les registres

Mise en pratique

Découverte du Zynq 7010 sur Redpitaya [4]
Accès au matériel depuis l'espace utilisateur ... [5]

- ... depuis le shell par `/sys/class/gpio` [6] : `Device Drivers` → `GPIO Support`
- ... depuis le shell par `devmem` pour manipuler les registres du processeur
- compiler un programme C depuis *buildroot* : le PATH doit contenir `$BR/output/host/usr/bin`
- démontrer l'accès aux registres depuis un programme utilisateur écrit en C faisant appel à `/dev/mem` (masquer l'adresse avec la taille de la page de la mémoire virtuelle)

→ familiarisation avec la manipulation des périphériques en vue d'écrire un pilote au niveau du noyau

4. `redpitaya.readthedocs.io/en/latest/developerGuide/125-14/shem.html`
5. `http://jmfriedt.free.fr/redpitaya.pdf`
6. `https://www.kernel.org/doc/Documentation/gpio/sysfs.txt`