CPCI

Table des matières

1 Introduction

Cette section décrit le fonctionnement des éléments à notre dispositions, qui s'articulent tous autour du bus COMPACT-PCI.

- Des châssis (plus communément appelés 'fond de panier' ou 'backplane') **compact pci** 21 slots. Les bus CPCI ne peuvent pas excéder 8 éléments. Aussi il s'agit de 3 châssis de 7 slots reliés par le biais de *bridges*. Ils sont de la marque *schroff*. Voici la documentation que j'ai trouvé.

- Des cartes contrôleur (documentation). Il s'agit de cartes équipées d'un processeur POWER PC et sur lesquelles tourne un OS LINUX FROM SCRATCH. Ces cartes contrôleurs sont identiques et donc interchangeables. Le bios PPCMON (power pc monitor) est un *shell* donnant accès à des outils de diagnostique et de configuration. Ce bios est accessible via un lien série.

Durant la phase de configuration du bus (p593) il y a 'adressages géographiques' (par slots) afin de déterminer des adresses qui seront ensuite utilisées vie 'adressage physique'. Cette phase d'initialisation consiste à lire et à compléter les premiers registres de chaque carte PCI que l'on nomme le BAR (p353) :

- Device ID et Vendeur ID (on utilise celui du Cern) Ils servent à connecter la carte via le driver générique *pci*.

- Subsystem ID et Vendeur ID Ils peuvent éventuellement venir compléter la recherche précédente.

- Base Address [0-5] (plages de mémoires. Par exemple sur les cartes FIFO il y a 2 plage : une de 128k et l'autre de 8k). Contient initialement le nombre de mémoire à allouer
  Est écrasé avec l'adresse affectée.

En fait, le terme '*master*' désigne la carte qui est à l'initiative de l'échange. Le bus PCI ne peut avoir qu'un seul *arbitre* et il semble qu'il y ait une confusion des rôles qu'il va falloir éclaircir.

2 État des lieux (avant les patchs)

Par défaut, la carte contrôleur installée dans le premier slot du châssis (celui le plus à droite) est configurée comme *master*. Le deuxième contrôleur, installé dans n'importe quel autre slot sera alors définit en tant qu'*esclave*.

2.1 Procédure

Sur le contrôleur *master* la commande `diag cpci` nous permet de voir :

- l'adresse mémoire à laquelle les cartes esclaves pourront accéder à la mémoire du *master*.
- Le numéro des slots sur lesquels se trouvent les cartes esclaves.

Sur le contrôleur *slave* la commande `diag cpci` nous permet de voir :

- l'adresse mémoire à laquelle les autres cartes pourront accéder à la mémoire du contrôleur esclave.
- Le numéro du slots sur lesquels se trouve le contrôleur esclaves.

Remarque : le numéro localisant le contrôleur esclave retourné par les 2 contrôleurs ne correspond pas.

On peut tester l'accès mémoire de chacun des contrôleur. Lecture de la mémoire du *master* de le contrôleur *slave* :

```
master> pm.l 0
        ...
                                slave> dm.l <adresse master>
```

Lecture de la mémoire du *slave* de le contrôleur *master*.

```
                                slave> pm.l 0
                                ...
master> dm.l <adresse slave>
```

2.2 Résultats

La mémoire est indiqué en millions. Lorsqu'un chiffre figure (40, 42, 44...) cela signifie que la mémoire est indiqué explicitement et qu'elle est accessible par l'autre carte contrôleur conformément aux procédures décrites ci-dessus.

Numérotation des slots :

n. de droite à gauche	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
numérotation *master*								9	A	B	C	D	E	F		6	5	4	3	2	
numérotation *slave*	8	9	10	11	12	13	14	8	9	10	11	12	13	14	1	2	3	4	5	6	

Adresses mémoire affectées avec 2 cartes contrôleur en fonction de l'emplacement choisi pour la carte *slave* :

slot *slave*	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	aucun
adresse *master*	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	42	42	42	42	42	40
adresse *slave*								42	42	42	42	42	42	42		40	40	40	40	40	

Adresses mémoire affectées avec 2 cartes contrôleur et jusqu'à 2 cartes fifo :

21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	aucun
																			F	40
																		40	F	42
																	F	40	F	42
													F				40			42
																40	F		F	42
													F			40			F	42
												F	F			40				42
											40	F	F							44
											40		F							44
										42									F	40

FIGURE 1 : représentation du châssis 21 slots

2.3 Conclusion

- On peut penser que les numéro de slot *master* et *slave* discordent mais n'empêchent pas les cartes de communiquer.
- Les cartes contrôleur sont interchangeables sans que les adresse *master* et *slave* changent.
- Les contrôleurs *slave* sur les slots 14 à 21 peuvent accéder à la mémoire *master* mais pas l'inverse.
- Les adresses mémoires affectées change en fonction de la disposition des cartes sur le châssis.

S'agit-il de bridge transparents ou non ?

3 Mapping depuis le second slot (patch 1)

Nous souhaitons que la carte contrôleur qui n'est pas dans le premier slot puisse récupérer les ressources de configuration cPCI.

- Patcher le noyau. Étrangement, il n'y a pas de soucis avec le problèmes annoncé. Cf le mail de **Rupert** :

```
Normally the CES-provided kernel will only enumerate if it detects itself
as system slot. However this can be changed easily in the file
arch/powerpc/platforms/cesrio/pci.c, by writing a 1 into
cpci_cfg->sys_slot in the function rio_find_bridges, before the loop where
rio_init_pci_bus() is called. We may plan to make this configurable in the
next release.

Because normal linux PCI device drivers occupy all devices they find,
it would lead to problems if the same device is found by two rios. This
is the reason why only the master slot enumerates.

To work around that problem, newer SugarHat releases (I think since 3.2.0 or
3.2.1) support filtering PCI slots, in a way that they are invisible to the
operating system. We have implemented this for other boards
with two processors, but is should be also usable in your case.
```

- Compiler puis booter le noyau avec l'extension *-shl-3.2.1* car elle est vérifiée lors du chargement des drivers *papat26**.
- Compiler puis charger le module *xpc* et *xpc_dma*

```
$ telnet 192.168.1.158
# cd /usr/src/xpc/driver
# make
# make install
# depmod -r
# modprobe -l
# modprobe xpc
# ls /proc/bus/xpc/
cpci  lpci  xpci
# modprobe xpc_dma
```

- Charger les modules

```
# insmod /home/camera/modules/papat26_dat.ko
# insmod /home/camera/modules/papat26_slc.ko
```

Attention : les fond de tiroirs PCI peuvent recevoir des cartes des 2 côtés. On peut endommager les cartes en les branchant dans un slot libre faisant face à un slot occupé.

4 Mapping derrière le second bridge (patch 2)

Les derniers slots ne sont pas mappés. Cela ce voit par l'absence des allocation mémoires énumérées par la commande `lspci -vv`. Ici, la 3ème carte n'es pas mappée :

```
# lspci -vv
0001:80:0b.0 Class 1180: 10dc:2ccc
        Control: I/O- Mem+ BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping+ SERR- FastB2B-
        Status: Cap- 66Mhz- UDF- FastB2B+ ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR-
        Interrupt: pin A routed to IRQ 32
        Region 0: Memory at 0000000042000000 (32-bit, non-prefetchable) [size=128K]
        Region 1: Memory at 0000000042100000 (32-bit, non-prefetchable) [size=8K]

0001:80:0e.0 Class 1180: 10dc:2ddd
        Control: I/O- Mem+ BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping+ SERR- FastB2B-
        Status: Cap- 66Mhz- UDF- FastB2B+ ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR-
        Interrupt: pin A routed to IRQ 29
        Region 0: Memory at 0000000042200000 (32-bit, non-prefetchable) [size=128K]
        Region 1: Memory at 0000000042300000 (32-bit, non-prefetchable) [size=8K]

0001:88:0c.0 Class 1180: 10dc:2ccc
        Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping+ SERR- FastB2B-
        Status: Cap- 66Mhz- UDF- FastB2B+ ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR-
        Interrupt: pin A routed to IRQ 31
```

Voici le patch de rupert. Il faut extraire l'archive directement dans les sources du noyau.

5 État des lieux (après les patchs)

Détails de la commande `lspci` :

```
# lspci
00:01.0 Class 0200: 1022:2000 (rev 44)
00:02.0 Class 0200: 8086:1008 (rev 02)
0001:20:04.0 Class 1180: 10dc:2bbb
0001:20:08.0 Class 0604: 3388:0026 (rev 04)
0001:80:08.0 Class 0604: 3388:0026 (rev 04)
0001:80:0e.0 Class 1180: 10dc:2aaa
```

La commande `lspci` fournit les détails suivants :

- bus et slot : [0001] :bb :ss.O
- class : xxxx
- id : xxxx :xxxx
- identifiant : `xxxx:xxxx`

id	carte
3388 :0026	bridge
1022 :2000	ctrl slot1
8086 :1008	ctrl slot2 (vu depuis ctrl1)
10d9 :4065	ctrl slot2 (vu depuis ctrl2)
10dc :2aaa	slow controle
10dc :2bbb	fifo data
10dc :2ccc	triger
10dc :2ccc	triger management

- domaine et bus `[dddd]:bb:xx.0`
  domaine : rien=LPCI, 0001=CPCI
  bus :

00	domaine LPCI (premier segment)
20	1er segment
80	2ème segment (derrière le 1er bridge)
88	3ème segment (derrière le 2ème bridge)

- slot `[0001]:xx:ss.0`
  Le châssis est composé de 3 segments de 7 slots numérotés différemment :

de 0x01 à 0x07	1er segment
de 0x0f à 0x09	2ème segment (derrière le 1er bridge)
de 0x0f à 0x09	3ème segment (derrière le 2ème bridge)

remarque : le bridge est affecte au slot 8 que ce soit pour le premier ou le deuxième segment.
remarque : on obtient le même chiffrage a partir du premier slot que ce soit sur un châssis de 14 ou de 21 slots.

```
PCI-TREE
├── LPCI Bridge BUS 0x00
│   ├── 0000:00:01.0
│   └── 0000:00:02.0
└── CPCI Bridge BUS 0x20
    ├── 0001:20:01.0
    ├── ...
    └── 0001:20:07.0
        └── 0001:20:08.0 BUS 0x80
            ├── 0001:80:08.0 BUS 0x88
            │   ├── 0001:88:09.0
            │   ├── ...
            │   └── 0001:88:0f.0
            ├── 0001:80:09.0
            ├── ...
            └── 0001:80:0f.0
```

FIGURE 2 : *liste des devices par Linux*

6 Nouveau code source shl-3.2.3

- don't boot

```
Please try to append "cpci_full_enum" to the boot_line. I had to
disable this feature by default because it is not useful to all
customers.

> By using the new kernel source, I cannot boot on the first slot, with or
> without a second controller card in the second slot. However there is no
> problem booting on the second slot.
```

- mapping pci

```
Please try the file in the attachment. It replaces arch/powerpc/kernel/pci_32.c.

> 'lspci -vv' shows :
> Region 0: Memory at <ignored> (32-bit, non-prefetchable)
```

- 2 Processeurs sur les slots 1 et 2 :
  Lorsqu'il y a des cartes au delà du premier segment, il faut ajouter `cpci_full_enum` aux paramètres du noyau du processeur en slot 1.

- 2 Processeurs sur les slots 1 et 2 avec un châssis 14 slots :
  Les cartes ne charge pas leur firmware et clignotent tous les deux en bleu.

- 2 Processeurs sur les slots 1 et 8 :
  Il faut recompiler le noyau en commentant le slot 8 dans le fichier *arch/powerpc/platforms/cesrio/rio3Resources.cesR*
  Le résultat est que le processeur en slot 1 ne peut plus franchir le premier bridge.

7 Carte rio sur le second segment (patch petit nicolas)

Code ajouté au fichier *arch/powerpc/platforms/cesrio/pci.c* :

```
/***** Add for Hess2 *****/

static int slaveInsertedOnSecondSegment = 0;
static int __init parse_slaveInsertedOnSecondSegment(char *arg)
{
        slaveInsertedOnSecondSegment = 1;
        return 0;
}
early_param("seg_2", parse_slaveInsertedOnSecondSegment);

static void fixup_hess2_enable2ndBridge(struct pci_dev *dev) {
  if (slaveInsertedOnSecondSegment){
    printk("RIO_PCI: %s call by slave rio on 21 slot crate\n", __func__);

    pci_slot_info[2][1].visible=1;      // enable device
  }
}
DECLARE_PCI_FIXUP_EARLY(0x8086, 0x1008, fixup_hess2_enable2ndBridge);

static void fixup_hess2_dontMapSlave(struct pci_dev *dev) {
  printk("RIO_PCI: %s call by master rio on 21 slot crate\n", __func__);

  dev->class = PCI_CLASS_NOT_DEFINED;  // don't map device
  dev->hdr_type = 3;                   // free device
}
DECLARE_PCI_FIXUP_EARLY(0x10d9, 0x4065, fixup_hess2_dontMapSlave);

/* end Add for Hess2 *****/
```

- Modifications du comportement du noyau :
  - L'esclave inséré sur le segment 2 est capable de voir le second bus (utile seulement si des cartes sont présentes sur le 3ème segment).
  - Le maître ne mappe plus la carte esclave afin de voir les même carte que l'esclave a partir du deuxième segment.
- Résultats :
  - non régression : le châssis fonctionne (comme avant) avec les 2 rios dans les premiers slots.
  - Les 2 premiers segments du châssis fonctionnent (drivers, big et zora) bien avec l'esclave sur le second segment.
  - Il semble que le 3ème segment soit correctement mappe lorsque l'esclave boot avec au moins une seconde de retard sur le second segment. Cependant, cela frise parfois complètement le maître lors du montage NFS ou du login.
  - rq : les interruption ne sont pas convenablement reparties.
  - rq : les cartes du 2ème segment sont numérotées dans le sens oppose par les 2 rios : il ne vaut mieux pas y melanger les types de cartes "a" et "b".
- Tests :
  - l'esclave "B" boot une seconde apres le maitre "A".
  - le maitre "A" charge le driver des cartes "a".
  - l'esclave "B" charge le driver des cartes "b".
  - Tests de non regression :
    - options du noyau de la rio maitre :
      `boot_line : cpci_full_enum`
    - options du noyau de la rio esclave :
      `boot_line : ip=:::::eth1 cpci_full_enum`

21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
									b	b	b	b					a	a	B	A
			a	b						b	b	b					a		B	A
			a	b						b	b	b					a	B		A
			a	b						b	b	b				B	a			A
				b								b	a	b	B	b	a			A

- – Tests pour la camera 20 tiroirs :
  - ∗ options du noyau de la rio maitre :
    `boot_line : cpci_full_enum`
  - ∗ options du noyau de la rio esclave :
    `boot_line : ip=:::::eth1 cpci_full_enum seg_2`

21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
												b	B						a	A
											b		B						a	A
										b			B						a	A
									b				B						a	A
								b					B						a	A
							b						B						a	A
							b					B							a	A

- – Tests pour la grande camera :
  - ∗ options du noyau de la rio maitre :
    `boot_line : cpci_full_enum`
  - ∗ options du noyau de la rio esclave :
    `boot_line : ip=:::::eth1 cpci_full_enum seg_2`

21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
								b	b	b			B					a	a	A
								b	b	b	b		B				a	a		A
							b	b	b	b			B				a	a		A
							b	b	b	b			B			a	a			A
							b	b	b	b			B			a	a			A
							b	b	b	b		B					a	a		A
							b	b	b			B	b				a	a		A
							b	b			B	b	b			a	a			A
							b	b		B		b	b			a	a			A
							b		B		b	b	b			a	a			A
								B		b	b	b	b			a	a			A
							B			b	b	b	b			a	a			A
	a						B			b	b	b	b	a						A
	a						b	b	b	b		B						a		A
	a							b	b	b	b		B					a		A
					b	b				b	b		B				a	a		A
					a	b			b	b	b		B					a		A

- – Cas douteux :
  - ∗ options du noyau de la rio maitre :
    `boot_line : cpci_full_enum`
  - ∗ options du noyau de la rio esclave :
    `boot_line : ip=:::::eth1 cpci_full_enum seg_2`

21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
							a	b	b	b	b		B					a		A