2^{eme} année 2007-2008

ModApp système/architecture

Juin 2008

1 Présentation générale

Le but de ce projet est de réaliser un pilote de périphérique nous permettant de communiquer entre deux stations par le biais du port parallèle. Nous souhaitons plus précisément être capable de véhiculer des paquets IP au travers de cette liaison, et permettre ainsi son utilisation par toute application basée sur cette pile de protocoles.

Plutôt que définir nous-même un nouveau protocole, nous allons implanter celui défini par Russ Nelson et connu sous le nom de *Crynwr*.

2 Le protocole crynwr

Nous allons utiliser le port parallèle dans son mode de fonctionnement le plus ancien. Nous aurons ainsi des performances particulièrement mauvaises, mais cela nous permettra d'observer plus finement certains aspects. Il sera de plus interropérable avec les autres implantations, y compris sous Linux.

Observons les différents niveaux d'une telle communication.

2.1 Niveau physique

La couche physique fournit un service de transmission de demi-octet, ou quadret (ou *nibble*) grâce au cablage décrit par la figure 1.

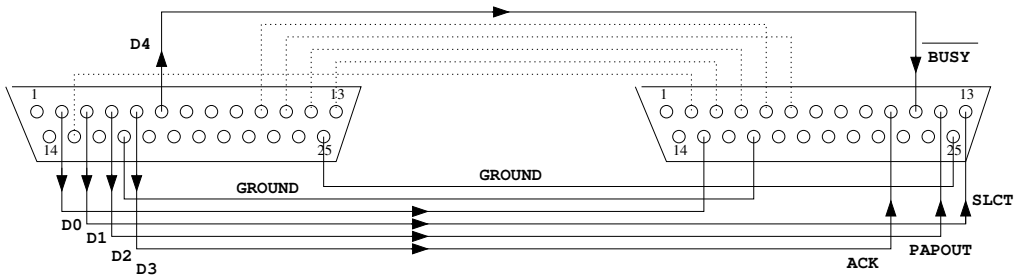

FIG. 1 – La connectique

La figure 2 montre alors comment le registre de données d'un port parallèle est reçu via le registre d'état du port parallèle en vis-à-vis.

Chaque demi-octet ne peut être émis que lorsque le vis-à vis en a donné la possibilité en positionnant son bit `D4` à 1 pour le quadret de poids faible ou à 0 pour celui de poids fort.

La figure 3 donne le chronogramme de transfert d'un octet par le port parallèle, du point de vue de l'émetteur, et en supposant que l'interruption ait été déclenchée sur le récepteur :

1. Le récepteur positionne à 0 son bit `D4` ce qui active le bit `BUSY` de l'émetteur.

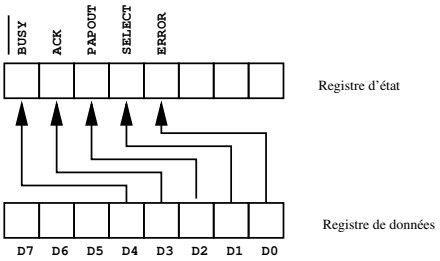

FIG. 2 – Relation entre les registres

2. L'émetteur peut alors placer les quatre premiers bits de donnés sur ses quatre bits `D0 .. D3`.
3. L'émetteur positionne alors son bit `D4` à 1 (ce qui fait passer à 0 le bit `BUSY` du récepteur) ce qui permet au récepteur de lire les données.
4. Le récepteur positionne à 1 son bit `D4` ce qui passe à 0 le bit `BUSY` de l'émetteur.
5. L'émetteur peut alors placer les quatre derniers bits de donnés sur ses quatre bits `D0 .. D3`.
6. L'émetteur positionne alors son bit `D4` à 0.

Attention, ce schéma reste relativement théorique ; certaines implantations fusionnent les opérations 2 et 3 (et, en toute logique, les 5 et 6). On veillera donc à traiter cela si l'on souhaite garder une compatibilité avec ces implantations.

Ce motif est répété autant de fois que nécessaire pour transmettre toute une trame de données. On veillera à l'accusé de réception du dernier quadret.

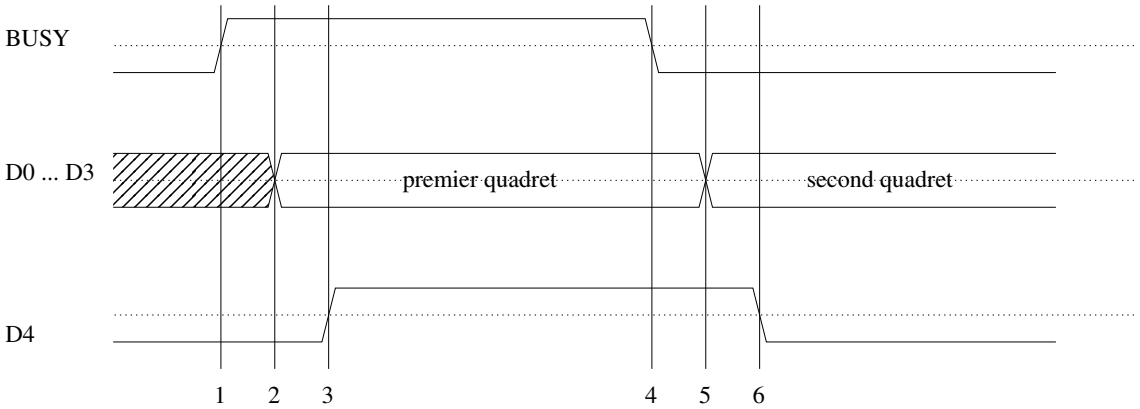

FIG. 3 – Chronogramme d'émission de deux quadrets consécutifs

Il est également important de bien noter que le chronogramme, du point de vue du récepteur, est légèrement différent du fait du brochage qui relie le bit de donnée `D4` au bit de contrôle `ACK` qui est actif à niveau bas.

2.2 Niveau liaison

Le format d'une trame *crynwr* est donné par la figure 4.

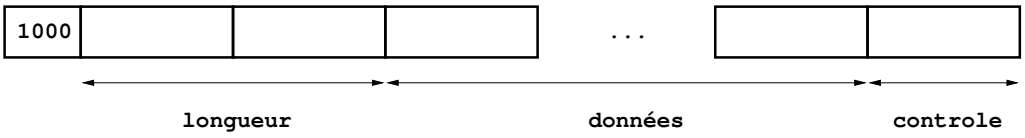

FIG. 4 – Format d'une trame crynwr

Le premier quadret émis ne véhicule pas d'information, il ne sert qu'à prévenir l'entité vis-à-vis du fait qu'une trame va être émise.

La longueur de la trame est donc envoyée d'abord, en commençant par son octet de poids faible, puis suit la charge utile, puis pour finir un code de contrôle, qui est simplement la somme des octets constituant la trame.

Pour chaque octet, c'est le quadret de poids faible qui est émis en premier.

Nous n'utiliserons le résultat de notre travail que pour véhiculer de l'IP, aussi nous ne définirons pas de champ permettant de multiplexer les protocoles de niveau réseau.

3 L'implantation

Le pilote sera réalisé sous Linux 2.4 sous la forme d'un module. Divers documents vous seront fournis pour vous aider dans cette réalisation.

4 Utilisation

En supposant que votre module se nomme `iop` et qu'il soit chargé dans le système, il sera rendu utilisable par la pile IP de la façon suivante :

```
# ifconfig iop0 192.168.1.1 pointopoint 192.168.1.2
```

Le nom de l'interface et les adresses sont à choisir en fonction de votre configuration. Naturellement, l'interface vis-à-vis devra avoir une configuration symétrique.

Puis on pourra tester le fonctionnement de l'interface grâce à la commande ping

```
# ping -c 1 192.168.1.2
```

Vous trouverez les différents paramètres de cette commande dans sa page de manuel.

Enfin, il est possible d'observer l'état de l'interface, ainsi que les statistiques associées grâce à la commande `ifconfig`

```
# ifconfig iop0
```

5 Déroulement du travail

Le travail sera réalisé en respectant les étapes suivantes

1. Écriture et compilation d'un module (qui affiche un message au chargement et un message au déchargement). Interfaçage avec le sous-système de gestion des ports parallèle.
2. Émission et réception d'un octet.

3. Émission et réception d'un tableau d'octets (construit statiquement). La réception se fera en dehors du gestionnaire d'interruption.
4. Interfaçage avec le sous-système réseau.
5. Construction et émission d'une trame. Réception et extraction du paquet contenu.

Attention, cette organisation est donnée à titre indicatif afin de vous permettre de planifier votre travail, elle ne constitue en aucun cas un premier raffinage du code à produire.