Chapitre 3

Accès aux périphériques

Nous avons vu, pour l'instant, comment le microprocesseur avait accès aux registres et à la mémoire vive. Il faut aussi avoir accès aux périphériques (d'entrée-sortie) de façon à introduire des données (*via* la clavier, la souris, un lecteur de disquettes, un disque dur, un lecteur de CD-ROM, une tablette à digitaliser, un scanner, un capteur...) et à communiquer les résultats (*via* l'écran du moniteur, une imprimante...). Nous allons voir dans ce chapitre comment le microprocesseur a accès à ceux-ci.

3.1 Principe

Les périphériques peuvent être d'une très grande complexité mais, du point de vue de la programmation et de l'échange des données, ils reposent sur un principe simple.

Contrôleur de périphérique.- Un périphérique est complexe, comprenant à la fois des parties mécaniques et des parties électroniques. La communication microprocesseur/périphérique et la programmation du périphérique se fait grâce à la partie électronique appelée **contrôleur**.

Adresse de périphérique.- Comme pour le microprocesseur, la partie essentielle d'un contrôleur de périphérique est un circuit intégré, que l'on peut considérer comme une boîte noire avec un certain nombre de broches. Un ensemble de huit de ces broches concernent les données. Elles sont reliées au microprocesseur de façon à correspondre à une **adresse** donnée **de périphérique**. Celle-ci n'est déterminée ni par le concepteur du microprocesseur, ni par celui du circuit intégré du contrôleur ; c'est le concepteur du système, l'ordinateur, qui la choisira de façon plus ou moins arbitraire.

Périphériques mappé en mémoire et indépendant.- Cette adresse peut être une adresse de mémoire vive (qui sera donc dédiée à un périphérique) : on parle alors de **périphérique mappé en mémoire**. Cette adresse peut également être spécifique aux périphériques, avec un jeu d'adresse indépendant du jeu d'adresses mémoire : on parle alors de **périphérique indépendant**.

Là encore, le fait qu'un périphérique donné soit, sur un système donné, mappé en mémoire ou indépendant ne dépend ni du périphérique, ni réellement du microprocesseur, mais du concepteur du système.

Cependant, si tous les microprocesseurs permettent le mappage en mémoire des périphériques, seuls certains microprocesseurs ont un jeu d'adresses indépendant pour les périphériques.

Notion de port.- Les microprocesseurs à entrée-sortie indépendante accèdent aux périphériques *via* un certain nombre de **ports**, chacun étant repéré par une adresse. Un port n'est rien d'autre qu'une liaison physique pour un octet.

Un tel microprocesseur indique, *via* une des broches, qu'il veut accéder à un port (et non à un élément mémoire). Il utilise le bus des adresses pour indiquer le numéro du port. Il utilise les broches des données pour envoyer ou recevoir une donnée.

Il appartient au concepteur de l'environnement du microprocesseur de relier comme il le veut, et donc de déterminer que tel ou tel port correspond à tel ou tel périphérique (ou partie de périphériques). Bien entendu, dans un environnement donné, tous les ports ne sont pas utilisés (reliés).

Registres internes des périphériques.- Les données sont rarement transmises directement. Un périphérique contient en général un ou plusieurs **registres internes** (toujours en très petit nombre, comme un microprocesseur) et on communique à travers ceux-ci.

Un périphérique utilisera en général plusieurs ports, souvent un par registre interne bien qu'on puisse aussi indiquer, à l'aide de broches spécifiques, à quel

registre on veut avoir affaire.

Programme interne.- Le contrôleur contient un programme (éventuellement câblé) qui détermine ce qu'il doit faire des données des registres internes. On parle de **firmware**. Celui-ci est indépendant du microprocesseur et ne nous concerne pas pour l'instant.

3.2 Cas du i8086

Les microprocesseurs `i8085` et `i8086` permettent à la fois le mappage en mémoire et l'accès indépendant via un port.

Nombre de ports.- Dans le cas du `i8085`, seules huit broches peuvent être utilisées pour déterminer l'adresse d'un port, ce qui en détermine le nombre maximum à 256, ce qui est certainement suffisant. Dans le cas du `i8086`, on utilise seize broches, ce qui porte le nombre maximum à 65 536.

Capacité d'un port.- Dans le cas du `i8086`, l'entrée ou la sortie *via* un port se fait octet par octet, et toujours *via* le registre `AL`.

Instructions.- L'instruction pour l'entrée est `in`, l'instruction pour la sortie est `out`. Pour chacune de celles-ci, on utilise l'une ou l'autre des deux formats suivants :

```
in al,port
```

```
mov dx,port
in al,dx
```

```
out port,al
```

```
mov dx,port
out dx,al
```

Les 256 premiers ports, numérotés de `00h` à `FFh`, peuvent être désignés directement par leur numéro alors que le numéro des autres ports (procédé également valable pour les 256 premiers), numérotés de `0000h` à `FFFFh`, doivent nécessairement être passés via le registre `DX`, c'est-à-dire qu'on est obligé d'utiliser le second format pour eux.

Comme d'habitude, la première façon de faire provient du `i8085` et elle a été maintenue pour une raison de compatibilité.

Ports réservés.- Le microprocesseur ne réserve aucun port mais pour un ordinateur *compatible PC* un certain nombre de ports sont réservés, par exemple le haut-parleur interne est relié au port `61h`.

3.3 Un exemple : voyants lumineux du clavier MF II

On aimerait bien commencer par un exemple permettant d'entrer un caractère via le clavier. Cependant, nous réservons un tel cas à plus tard car nous verrons que ce n'est pas très simple. Nous allons donc considérer un exemple beaucoup plus simple, la manipulation des voyants lumineux du clavier.

Les voyants lumineux.- Sur un clavier dit MF Il (et ultérieurs), il y a trois voyants lumineux, situés en haut à droite du clavier, dénommés "NumLock", "CapsLock" et "ScrollLock". Les deux premiers permettent de rappeler respectivement à l'utilisateur que le pavé numérique de droite sert à entrer des chiffres et non des déplacements et qu'on entre des majuscules (et non des minuscules).

Les voyants lumineux ne sont pas pilotés par le contrôleur du clavier, lorsqu'on appuie sur la touche de verrouillage du pavé numérique par exemple, mais par le microprocesseur de l'unité centrale et, plus exactement, par le système d'exploitation.

Le registre du clavier.- Nous verrons plus tard en détail le fonctionnement du clavier et son interfaçage au microprocesseur. Voyons pour l'instant que ce qui nous intéresse pour notre propos.

Le contrôleur d'un clavier pour compatible PC possède quatre registres internes, appelés **tampon d'entrée**, **tampon de sortie**, **registre de contrôle** et **registre de statut**. Sur un compatible PC, on utilise les deux ports d'adresses 6Oh et 64h pour accéder à ces registres. Il suffit de deux ports puisque chacun de ces registres est seulement accessible soit en écriture, soit en lecture. Pour le tampon d'entrée, le seul qui nous intressera ici, on écrit sur le port 60h.

Les commandes du clavier.- Le comportement des claviers AT et MF II peut être programmé grâce à un jeu de commandes que l'on transmet au contrôleur de clavier *via* le tampon d'entrée. Nous verrons plus tard, lorsque nous étudierons le clavier en détail, ce jeu de commandes. Contentons-nous ici de la commande qui permet d'allumer et d'éteindre les voyants lumineux du clavier MF II.

On passe la commande :

```
EDh
```

au tampon d'entrée. Le contrôleur du clavier répond par un accusé de réception `ACK`, ne s'occupe plus des touches et attend un **octet d'indication**, qui est également passé *via* le tampon d'entrée.

La structure de l'octet d'indication est la suivante :

```
0 0 0 0 0 CPSL NUML SCRL
```

avec `CPSL` égal à 1 pour que le voyant lumineux de "CapsLock" soit allumé, `NUML` égal à 1 pour que le voyant lumineux de "NumLock" soit allumé et `SCRL` égal à 1 pour que le voyant lumineux de "ScrollLock" soit allumé.

Exemple.- Utilisons le programme suivant permettant d'allumer le voyant 1umineux de "NumLock" (sans que le pavé numérique ne soit verrouillé) avec `debug` :

```
C>debug
-a
15BD:0100 mov al,ED
```

```
15BD:0102 out 60,al
15BD:0104 mov al,02
15BD:0106 out 60,al
15BD:0108 int3
15BD:0109
-g

AX=0002  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=15BD  ES=15BD  SS=15BD  CS=15BD  IP=0108   NV UP EI PL NZ NA PO NC
15BD:0108 CC            INT     3
-q
```

Remarques.- 1^o) On ne peut pas utiliser la commande de trace `T` de `debug` car au bout de deux appels cette commande, le clavier est en attente de l'octet d'indication et apparaît bloqué.

2^o) Il faut utiliser un vrai système d'exploitation MS-DOS pour lancer ce programme et non une fenêtre MS-DOS de Windows ou une émulation sous Linux. En effet ces deux derniers systèmes d'exploitation fonctionnent en deux modes : le *mode noyau* et le *mode utilisateur*. Le mode noyau permet de lancer ces systèmes d'exploitation et n'est plus accessible après (y compris par le superutilisateur `root`). Seul le mode noyau a accès aux entrées-sorties (c'est-à-dire aux instructions `in` et `out`) ; en mode utilisateur on accède aux entrées-sorties à travers des *appels système*, à condition qu'ils soient prévus. C'est la raison pour laquelle il faut recompiler le noyau de Linux lorsqu'on change de périphérique et qu'il faut redémarrer l'ordinateur pour Windows.