# Le PPI

*— Basé sur les articles publiés dans Quasar CPC numéro 11 [http://futurs.cpcscene.net/FindexQuasarCPC11.html], Initiation à l'Assembleur, et numéro 12 [http://futurs.cpcscene.net/FindexQuasarCPC12.html], Assembleur : Software, par Zik.*

Nous allons donc ici nous pencher sur le 8255, plus connu sous le nom de PPI (en français : interface programmable pour périphériques) qui, comme chacun sait (!) s'occupe sur CPC de bon nombre des entrées/sorties. À travers le PPI sont gérés, le PSG (le AY-3-8912), le clavier (par l'intermédiaire du AY-3-8912), le lecteur de cassettes, le signal de VBL du CRTC, le signal "Busy" de l'imprimante et d'autres signaux indiquant la configuration de l'écran et la marque de l'ordinateur (le système s'en sert à l'initilisation de l'ordinateur).

# Le PPI en théorie

Alors, le PPI possède 3 ports de 8 bits chacun plus un registre de contrôle du mode de fonctionnement de ces ports. Ces ports sont situés aux adresses &F4xx, &F5xx et &F6xx, ils sont appelés respectivement port A, B et C. Le registre de contrôle est placé en &F7xx. Ce dernier sert à définir, pour chaque port, s'il est en entrée ou en sortie, ainsi que son mode de travail. Le port C a la particularité de pouvoir se scinder en deux moitiés, c'est-à-dire que l'on peut mettre ses bits 0 à 3 en sortie et ses bit 4 à 7 en entrée simultanément et vice versa.

## Les modes de travail

Le PPI dispose de trois modes de travail :

- le mode de travail 0 : simple entrée/sortie,
- le mode de travail 1 : entrée/sortie manipulable,
- le mode de travail 2 : bus à deux sens.

Je vais vous parler brièvement des trois modes mais sachez que sur CPC, seul le mode 0 est vraiment valable.

### Mode 0

Dans ce mode, on dispose des trois ports comme entrée ou sortie comme définit par le registre de contrôle (le port C étant défini par groupe de 4 bits).

### Mode 1

On ne parle alors plus de trois ports mais de deux groupes : le groupe A comprend le port A associé aux bits 4 à 7 du port C, le groupe B comprend le port B associé aux bits 0 à 3 du port C. Dans chaque groupe, les 4 bits du ports C fonctionnent alors comme des signaux de commande et de réception pour le port du groupe.

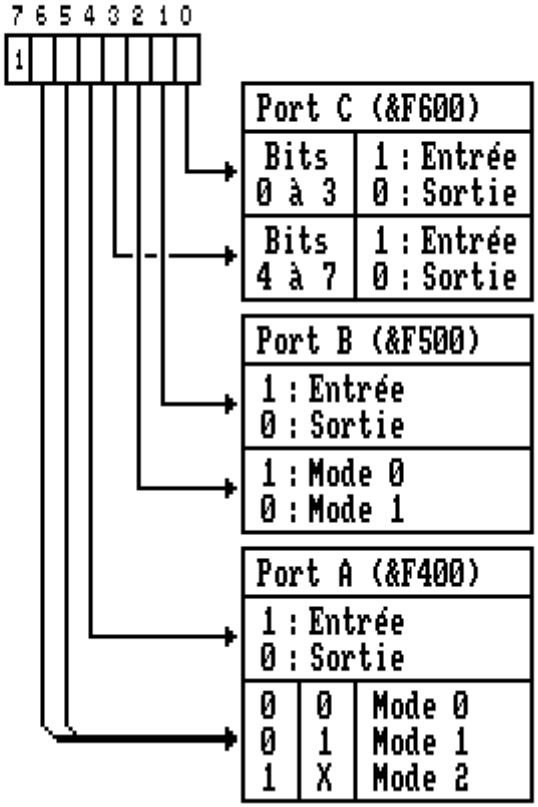
## Mode 2

Ce mode n'est disponible que pour le port A ; c'est un mode de travail bidirectionnel, c'est-à-dire que le port A peut alors être utilisé comme entrée et sortie en même temps. Les bits 3 à 7 du port C sont alors utilisés comme signaux de commande et de réception.

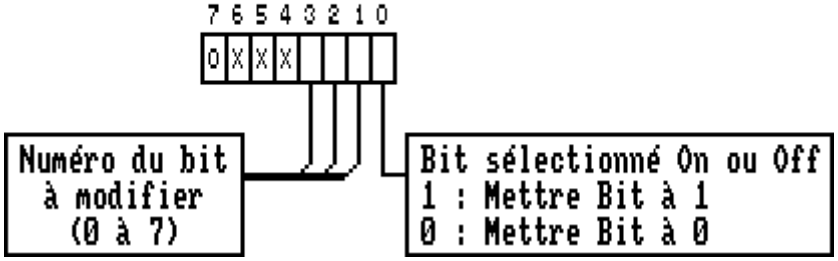
## Le registre de contrôle

Je vous demande de jeter un coup d'oeil aux schémas correspondants ci-après ; je ne vais pas répéter ici tout ce qu'ils indiquent.

Registre de contrôle du PPI (port &F700)
Choix du mode de travail des ports A, B et C :

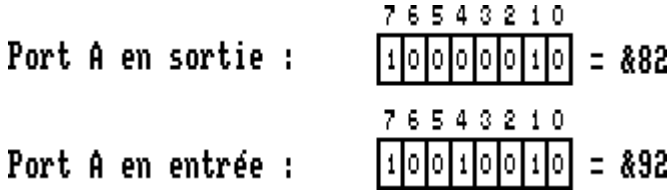



Registre de contrôle du PPI (port &F700)
Contrôle du Port C bit à bit :



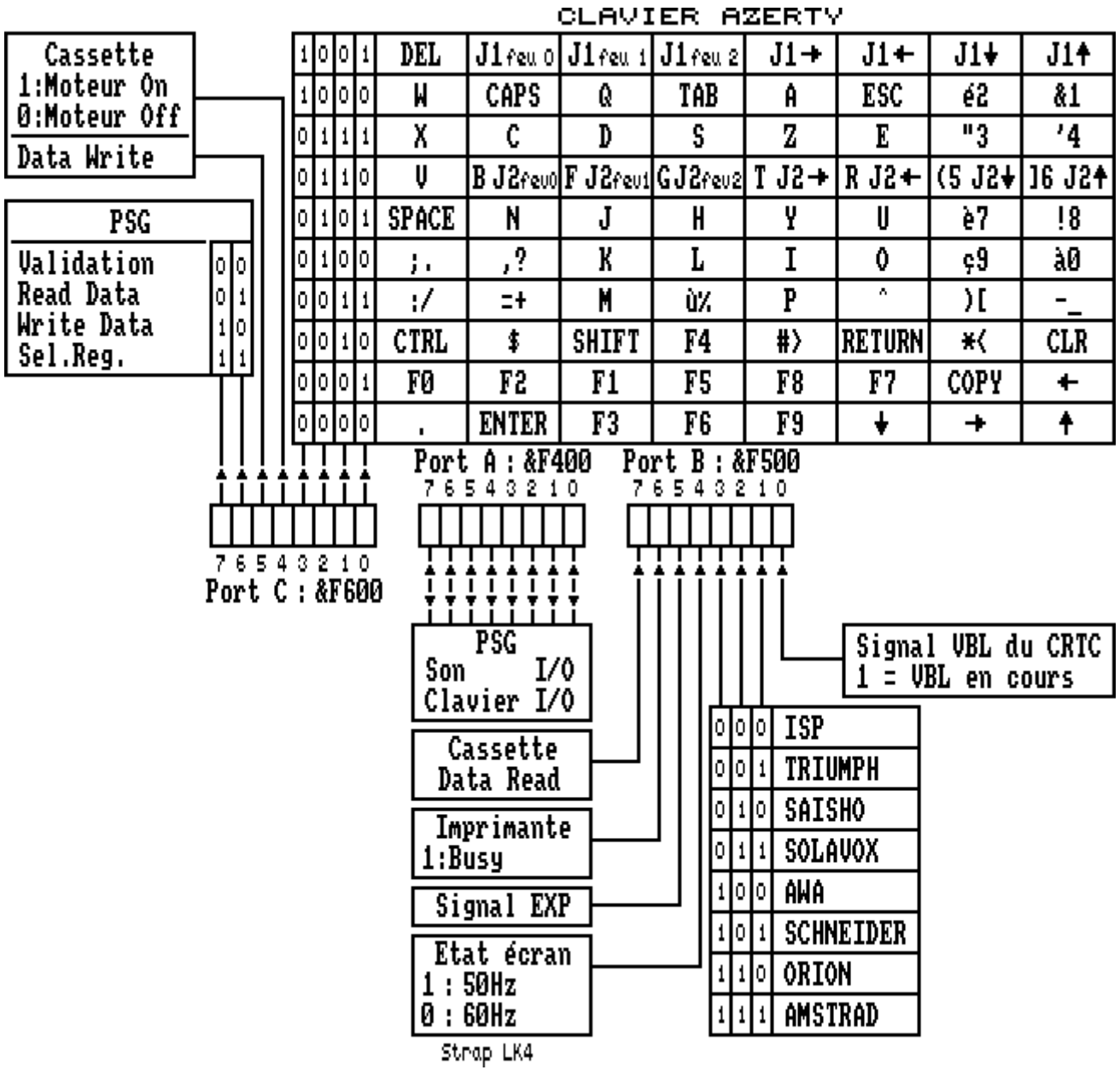

Vous vous êtes tous aperçus du rôle du bit 7 ; il définit la signification des autres bits du registre. Quant il est à 0, le registre permet de modifier la valeur du port C bit à bit, cela peut servir si l'on veut modifier l'état d'un bit sans toucher aux autres (évitant ainsi d'avoir à lire d'abord le contenu du port C avant de pouvoir y écrire). Si le bit 7 est à 1, le registre de contrôle définit quels ports sont en entrée ou en sortie et leur mode de travail.

En fait, la description que je viens de faire traite du PPI en général, mais sur CPC, le port C est utilisé généralement en sortie et le port B en entrée. Seul le port A est utilisé en entrée et en sortie. On a donc plus que deux cas sur le port &F7xx :




## L'interfaçage

Pour finir, voici un schéma qui reprend l'interfaçage complet du PPI dans le CPC avec ses différents ports.




Voilà, j'espère que vous avez à peu près tout compris. Dans la section qui suit, nous allons voir des applications pratiques…

## Le PPI en pratique

# Le test clavier


Qui dit test clavier, dit PSG et donc PPI. Si vous avez lu la section sur le PSG, vous avez (ou vous auriez !) dû remarquer que celui-ci dispose, en plus de ses 14 registres audio, d'un registre entrée/sortie dédié au clavier.

En effet, General Instrument avait développé le PSG pour l'inclure dans des petits jeux électroniques ce qui explique qu'il gère un petit clavier… Et ce qui explique que l'on ait besoin de 4 bits supplémentaires pour définir les touches sur CPC !

Ces 4 bits sont les bits inférieurs du port &F6xx, ils définissent "la ligne clavier" que l'on veut tester. Remarquez que l'on lit l'état de huit touches à la fois, reportez-vous au tableau ci-dessus pour connaître la correspondance entre le numéro de ligne, les touches, et le bit qui donne l'état d'une touche donnée (0 signifie que la touche est enfoncée). Tout de suite un exemple, le programme suit…

Donc, si vous voulez par exemple tester la touche espace, la ligne clavier correspondante est la 5 ; si le bit 7 de l'octet récupéré est à 0 c'est que la fameuse touche est enfoncée…

Maintenant voici le programme (que je vous laisse le soin d'optimiser !) :

```
;
; Test clavier de la ligne
; dont le numéro est dans D
; D doit contenir une valeur de 0 à 9
;
        ld bc,&f40e  ; Valeur 14 sur le port A
        out (c),c
        ld bc,&f6c0  ; C'est un registre
        out (c),c    ; BDIR=1, BC1=1
        ld bc,&f600  ; Validation
        out (c),c
        ld bc,&f792  ; Port A en entrée
        out (c),c
        ld a,d       ; A=ligne clavier
        or %01000000 ; BDIR=0, BC1=1
        ld b,&f6
        out (c),a
        ld b,&f4     ; Lecture du port A
        in a,(c)     ; A=Reg 14 du PSG
        ld bc,&f782  ; Port A en sortie
        out (c),c
        ld bc,&f600  ; Validation
        out (c),c
; Et A contient la ligne
```


Je vous rappelle que le port A est le port &F400. À part ça, je pense que cette routine est assez claire, un détail quand même : je considère que le port A est en sortie au lancement. Quant à lui, le port C (&F600) est toujours en sortie. Et quand je parle de BC1 et BDIR, il s'agit respectivement des bits 6 et 7 du port C…

Il est possible de mettre le clavier en sortie, pour cela il faut modifier le bit 6 du registre 7 en conséquence (voir cette section). Cette possibilité permet d'utiliser le port joystick comme un port de communication entrée/sortie.

## Le son

Je vais passer ce paragraphe beaucoup plus rapidement puisque c'est exactement la même chose que pour le clavier et que vous avez déjà tous les programmes ici.

Dans le cas où vous voulez écrire dans un registre du PSG, en considérant que le port A est en sortie, la procédure à suivre (en pseudo BASIC) est :

```
OUT &F400,numéro de registre
OUT &F600,&C0 ' BDIR1 et BC1=1 => Le PSG sélectionne le registre
OUT &F600,0   ' Validation
OUT &F400,valeur à mettre dans le registre
OUT &F600,&80  ' BDIR=1 et BC1=0 => Le PSG lit la valeur
OUT &F600,0    ' Validation
```

Pour la lecture du contenu d'un registre du PSG, c'est exactement la même chose que pour le test clavier (bien sur vous n'avez pas à sélectionner une ligne clavier !).

Ça donnera donc :

```
OUT &F400,numéro de registre
OUT &F600,&C0 ' BDIR=1 et BC1=1 => Le PSG sélectionne le registre
OUT &F600,0   ' Validation
OUT &F700,&92 ' Port A en entrée
OUT &F600,&40 ' BDIR=0 et BC1=1 => Le PSG renvoie la valeur
A=INP(&F400)  ' A contient la valeur du registre
OUT &F700,&82 ' Port A en sortie
OUT &F600,0   ' Validation
```

Bon, je crois que je vais arrêter là avec les routines sur le PPI, je ne vous ferai pas l'affront de donner la routine de test VBL qui consiste juste à tester le bit 0 du port &F5xx en continu jusqu'à ce qu'il passe à 1 (annonçant le début du balayage écran).

Je pense que je vous ai assez matraqué avec le PPI !



Quasar Net