RAMCARD

Emulateur d'extension ROM pour CPC

Manuel d'utilisation

INTRODUCTION

En premier lieu, je tiens à vous féliciter et à vous remercier d'avoir fait l'acquisition de ce kit d'extension Rom pour CPC. Ce kit vous permettra de gérer 128Ko de Roms additionnelles sur tous types de CPC/CPC+, AMSTRAD/SCHNEIDER, 464/6128.

La RAMCARD permet de gérer jusqu'à 8 Roms de 16Ko consécutifs parmi les 256 Roms d'extension possible (de 0 à 255). Chaque Rom d'extension pouvant être validée ou dévalidée individuellement par interrupteur directement accessible sur la carte.

Cette carte d'interface n'utilise aucune EPROM, elle utilise une RAM de 128Ko (ou 32Ko) qui permet de simuler 8 (ou 2) ROMs de 16Ko. L'écriture dans la RAM de la RAMCARD est rendue possible grâce à un interrupteur et se fait le plus simplement du monde par une écriture en RAM du CPC aux adresses de &C000 à &FFFF.

Vous pouvez donc utiliser la RAMCARD pour augmenter le potentiel de votre ordinateur en y ajoutant des Roms d'extension, pour développer vos propres programmes en ROM, ou encore comme une mémoire RAM supplémentaire. Ce manuel développera donc ces différents aspects.

Ce manuel n'est délivré qu'aux heureux possesseurs d'un kit RAMCARD (carte seule, carte + composants à monter ou carte montée prête à l'emploi), il a été toutefois mis sur Internet afin d'en faciliter sa diffusion, y a t'il un volontaire pour sa traduction ?

I) CABLAGE ET MONTAGE

Cette section ne vous est destinée que si vous avez entrepris la réalisation de A à Z de la RAMCARD. Pour ceux qui ont fait l'acquisition de la version montée prête à l'emploi vous pouvez aller directement au prochain chapitre.
Voici la liste des composants nécessaire à la réalisation de l'interface. Cette liste ne comporte que les composants propre à la carte, il ne faudra pas oublier le câble et les connecteurs de raccordement adaptés au CPC :

- 6 résistances de 10kW (R1 à R4, R7, R9)
- 2 résistances de 1kW (R5, R8)
- 1 résistance de 47kW (R10)
- 1 résistance de 100kW (R6)
- 10 diodes 1N4148 (D1 à D9, D12)
- 2 diodes BAT85 (D10, D11) ou équivalent (diodes à faible tension de seuil)
- 1 diode zener 3,3V (DZ)
- 3 condensateurs de 100nF (C2, C3, C4)
- 1 condensateur de 1µF (C1)
- 1 condensateur de 100µF (C5)
- 1 transistor BC547B (NPN)
- 2 DIPSWITCHs de 8 interrupteurs
- 2 supports 14 broches
- 2 supports 16 broches
- 1 support de 20 broches
- 1 support de 32 broches (ou de la barrette tulipe sécable)
- 1 circuit 74LS00 (IC2)
- 1 circuit 74LS32 (IC6)
- 1 circuit 74LS85 (IC5)
- 1 circuit 74LS138 (IC3)
- 1 circuit 74LS374 (IC1)
- 1 circuit RAM (IC4) 32Ko ou 128Ko (62256, 43256, 681000, 551000 ou équivalent)
- 1 connecteur HE10 mâle 50 points à souder sur Circuit Imprimé
- 1 pile 3V au lithium CR2032
- 1 support de pile CR2032

Un soin particulier est à apporter au choix des composants, notamment pour les 2 diodes à faible tension de seuil, la pile et son support. Le reste des composants est des plus classique vous pouvez donc choisir en fonction de vos goûts, circuit 74HC, autres valeurs de résistances, interrupteurs déportés ...

Dans la mesure où vous savez ce que vous faites vous pouvez modifier l'interface comme bon vous semble. Cette interface n'est pas une interface commerciale, il n'y a aucun copyright ni garantie. Vous êtes seul responsable de tout dommage pouvant être occasionné par un mauvais câblage ou tout autre mauvais fonctionnement de l'interface.

Si vous respectez bien le schéma d'implantation donné en page 29 (dernière page), la liste des composants et que vous vérifiez votre travail, la RAMCARD fonctionne dès la première utilisation.
L'implantation des composants s'opérera en plusieurs phases, après chaqu'une des phases on soudera les composants au plus près du circuit imprimé. On réutilisera les chutes des résistances, diodes et condensateurs pour réaliser les quelques straps de la carte. L'ordre d'implantation est le suivant: résistances, diodes, straps (au nombre de 11), supports, condensateurs, transistor, dipswitchs, support de pile et finalement le connecteur. Respectez bien le sens des diodes, du transistor, des condensateurs chimique, du connecteur HE10 (attention au détrompeur) ainsi que les différents supports qui vous aideront à mieux placer les circuits intégrés.

Si vous trouvez que les dip-switchs sont d'un usage peu pratique vous pouvez les remplacer par des interrupteurs déportés. Des problèmes dans le circuit de sauvegarde liés à la diode zener ont été constatés, une alternative économique consiste à supprimer la diode D12 et le condensateur C1, à remplacer la résistance R8 par une résistance de 33kW et à mettre en place une résistance de 33kW en lieu et place de la diode zener DZ.

Vérifiez vos soudures et testez l'absence de court-circuits ou de micro-coupure à l'aide d'un testeur de continuité. Ne négligez pas ces vérifications, les pistes étant par endroits très denses, un court-circuit ou une coupure est vite arrivé et échappent le plus souvent à l'oeil nu. Implantez les circuits intégrés, la RAM et enfin en dernier la pile. Une RAM de 32Ko possède un nombre de broche inférieur à une RAM 128Ko, insérez-la de façon à laisser 4 broches vides sur la carte à coté de l'encoche. Vérifiez que la carte finale corresponde bien au schéma donné, vérifiez que tous les composants soient bien implantés et dans le bon sens.

Ce n'est qu'une fois que tout a été vérifié que vous pourrez connecter votre nouvelle interface à votre cher CPC. Toutes ces vérifications ne sont pas du temps de perdu, bien au contraire, elles vous éviterons d'avoir de mauvaises surprises, mais à vous de voir vous faites ce que vous voulez après tout ...

II) RACCORDEMENT AU CPC

Le raccordement à l'ordinateur se fait à l'aide d'un câble spécial à réaliser ou à se procurer en plus de la carte. Le câble doit être approprié à votre CPC c'est à dire qu'il vous faut soit un câble **connecteur encartable - connecteur HE10 femelle** pour les CPC AMSTRAD ancienne génération ou un câble **connecteur Centronic - connecteur HE10 femelle** pour les CPC+ et les CPC SCHNEIDER par exemple.

Les cartes livrées montées sont prêtes à l'emploi, elles ont été testées avant d'être expédiées, de même si vous montez vous même la RAMCARD et si vous avez bien pris soin de vérifier votre travail, vous êtes en possession d'une interface en parfaite état de marche. Tout non fonctionnement de votre interface serait dû à ce stade, à un problème de connexion (oxydation par exemple).

Je vous rappelle que toutes erreurs de conception ou la connexion d'une interface à l'envers peut s'avérer fatale à votre CPC. L'auteur décline toutes responsabilités sur les dommages occasionnés par un mauvais branchement de la RAMCARD. Faites donc attention à ce que vous faites...

Connectez le câble à votre RAMCARD et à votre CPC. La RAMCARD possède un détrompeur pour vous empêcher de connecter le câble en nappe à l'envers (j'espère que vous avez bien respecté le sens du connecteur HE10 à souder). Ne vous trompez pas dans le sens du connecteur sur le CPC, le câble doit relié le CPC et la RAMCARD sans vriller. Le tout est à connecter **CPC ETEINT !!!**

Les connecteurs HE10 utilisés sur la RAMCARD permettent de débrancher l'interface tout en laissant le câblage sur l'ordinateur. Vous pouvez ainsi changer d'interface très rapidement et éviter des problèmes de faux contacts sur le connecteur du CPC.

Une fois votre interface connectée à votre CPC et avant d'allumer votre ordinateur, placer les interrupteurs 1 à 5 de SW1 (l'interrupteur 1 est celui situé le plus près de la pile) sur la position ON, tous les autres étant sur OFF, la carte se trouve ainsi dévalidée. Allumez votre CPC, l'ordinateur doit vous rendre la main normalement, si ce n'est pas le cas, éteignez votre CPC, déconnectez le câble en nappe du CPC et reconnectez-le de nouveau. Un non fonctionnement de la RAMCARD provient à 99% d'un problème de contact au niveau du connecteur du CPC, débranchez et rebranchez le connecteur (Attention: le CPC DOIT ETRE ETEINT !!!) jusqu'à obtenir le fonctionnement normal de votre ordinateur.

Lorsque le CPC démarre normalement (si ce n'est pas le cas éteignez-le, débranchez l'interface et reconnectez-la), vous allez pouvoir tester le bon fonctionnement de votre interface à l'aide du programme ROMTEST qui permet de localiser toutes les ROMs du CPC (de 0 à 255). Votre interface étant dévalidée (interrupteurs sur OFF) vous ne devez détecter que la ROM numéro 7 (AMSDOS) et éventuellement vos ROMs d'extension provenant d'une autre interface (ROMCARD par exemple) ou les ROMs cartouche sur CPC+.

Amusez-vous à changer la position des interrupteurs 1 à 8 de SW2, vous devez constater, au bout de quelques secondes, l'apparition ou non du numéro de ROM associée, ex: ROM 2 pour le switch 3, ROM 6 pour le switch 7 ...
Si le basculement d'un switch de SW2 ne provoque pas l'apparition d'une ROM après 5 ou 6 secondes (délai de latence de l'affichage) veuillez éteindre l'ordinateur et reprendre la procédure de connexion au CPC, l'interface n'est pas "vu" par le CPC il y a donc sûrement un problème de câblage.

Essayez également le basculement des switch 1 à 5 de SW1, ces interrupteurs permettent de décaler le numéro de ROM par 128, 64, 32, 16 ou 8, essayez également une combinaison de ces switchs et observez comment cela influence les numéros de ROM. Pour terminer vos premiers pas avec la RAMCARD basculez l'interrupteur 8 de SW1, les numéros de ROM de la RAMCARD apparaissent en rouge indiquant que l'écriture dans ces ROMs est possible.

Avec ces quelques essais vous avez pu entrevoir le fonctionnement des interrupteurs de la RAMCARD. Le prochain chapitre termine la présentation de la RAMCARD et vous dévoile en détail l'utilisation des différents switchs.

III) CONFIGURATION DES INTERRUPTEURS

Votre RAMCARD vous permet d'ajouter jusqu'à 128Ko de pseudo-ROM (géré en 8 blocs de 16Ko) à votre CPC. Chaque "bloc" porte un numéro d'identification qui DOIT ETRE UNIQUE, c'est le numéro de ROM. Lorsque le CPC accède à une ROM d'extension, il spécifie avant tout ce numéro afin que seule la ROM désirée réponde.

Le CPC gère un numéro de ROM sur 8 bits soit 256 combinaisons. La RAMCARD contrairement à certaines interfaces gère complètement les numéros de ROM sur 8 bits ce qui permet de placer les 8 "blocs" (les 8 ROMs) de la RAMCARD n'importe où entre 0 et 255. Par contre les 8 blocs ont obligatoirement des numéros consécutifs (ex: 0 à 7, 32 à 39 ou 128 à 135, etc...).

Toutes les ROMs d'extension occupent les mêmes adresses en mémoire que la ROM BASIC c'est à dire de &C000 à &FFFF. Cette zone est également partagée par la RAM et contient, dans la majeure partie des cas, les informations vidéo (mémoire écran). La lecture dans la zone &C000 à &FFFF retourne le contenu de la RAM ou de la ROM d'extension, par contre l'écriture dans cette zone écrit toujours en RAM. Les ROMs d'extension étant des mémoires que l'on ne peut pas écrire et qui n'ont pour finalité que le stockage de programmes cela se comprend.

La RAMCARD émule parfaitement ce comportement ce qui permet son utilisation comme une vulgaire ROMBOARD ou ROMBOX pour l'utilisation de programmes en ROM. La pile au lithium permet de conserver les données de la RAMCARD pendant plusieurs années !!! L'illusion est totale, la RAMCARD se comporte comme n'importe quelle extension ROM !!!

La RAMCARD s'ouvre une nouvelle dimension lorsque l'on bascule le "switch magique", le switch d'écriture. Lorsqu'il est sur la position ON toute écriture dans la zone entre &C000 à &FFFF écrit dans la RAM centrale du CPC (comportement traditionnel du CPC) mais l'écriture va également à la RAM de la RAMCARD. La relecture de la ROM de la RAMCARD renverra la donnée modifiée, c'est comme si l'on avait "écrit dans une ROM". Il est donc ainsi possible de programmer ses ROMs d'extensions par CPC et il est même possible, en sacrifiant les 16Ko en mémoire RAM &C000-&FFFF d'avoir une extension RAM de 128Ko.

Pour bien configurer les switchs de la RAMCARD il faut encore comprendre comment le FIRMWARE du CPC gère les numéros de ROM. Sur les 256 combinaisons, le FIRMWARE ne gère que les 252 premières (0 à 251), les numéros de 252 à 255 ne sont pas accessibles par les routines systèmes. Au démarrage de l'ordinateur seules les 16 premières ROM (ou les 7 premières pour le 464) sont testées et initialisées si elles existent (la ROM est exécutée permettant ainsi de se réserver de la mémoire).

Du fait que seules les 16 premières ROMs sont testées de nombreuses interfaces se limitent à ces numéros, la RAMCARD elle, permet de gérer les 256 combinaisons permettant à l'utilisateur d'en faire un autre usage que celui permis par le système.

Le bloc d'interrupteur DIPSW1 de la RAMCARD permet de définir la plage des 8 numéros de ROM consécutifs et de gérer l'écriture. Les micro-interrupteurs 1 à 5 de DIPSW1 sont affectés d'un poids binaire, l'interrupteur 1 étant affecté du plus fort poids, le 5 du plus faible poids. Autrement dit le switch 1 correspond à la valeur 128, le switch 2 à 64, le switch 3 à 32, le switch 4 à 16 et le switch 5 à 8.
Les différentes combinaisons de ces 5 interrupteurs définissent 32 plages différentes où débutent les 8 numéros de ROMs consécutifs. L'interrupteur 8 autorise ou non l'écriture dans la RAMCARD. Un seul switch gère l'écriture pour toute la carte, l'écriture si elle est autorisée est valable pour toutes les ROMs déclarées actives. Il est impossible de rendre certains numéros accessibles en lecture/écriture et d'autres en lecture seule. Les switchs 6 et 7 quant à eux ne sont pas utilisés.

Les micro-interrupteurs 1 à 8 de DIPSW2 permettent de valider/invalider un numéro de ROM spécifique. Le switch 1 est affecté au numéro de ROM base+0, le switch 2 à base+1, le switch 3 à base+2, ..., le switch 8 à base+7, "base" étant le numéro de base définit par DIPSW1. Lorsqu'une ROM est invalidée il est impossible d'y lire ou d'y écrire des données. Si une ROM est validée il est possible de lire son contenu et éventuellement, en fonction du switch 8 de DIPSW1, d'écrire dedans. Je rappelle qu'une écriture en ROM écriera aussi la donnée en RAM interne.

Le tableau qui suit résume toutes les combinaisons pour les différents interrupteurs. Les interrupteurs sont représentés par un 1 lorsque l'interrupteur est fermé (position ON) et par un 0 lorsque l'interrupteur est ouvert (position OFF), un 'x' signifiant que l'interrupteur peut être indifféremment ouvert ou fermé. Dans le tableau les inters sont lus de gauche à droite comme sur la carte lorsque le connecteur pour CPC est placé en bas de carte et la pile sur le coté gauche.

Etat switch 8 de DIPSW1

0	Ecriture dans la RAMCARD impossible
1	Ecriture possible dans les ROMs activées

Sélection de ROM avec DIPSW2

1	ROM base+0 (ON/OFF)
2	ROM base+1 (ON/OFF)
3	ROM base+2 (ON/OFF)
4	ROM base+3 (ON/OFF)
5	ROM base+4 (ON/OFF)
6	ROM base+5 (ON/OFF)
7	ROM base+6 (ON/OFF)
8	ROM base+7 (ON/OFF)

Configurations possibles
avec DIPSW1

+128	+64	+32	+16	+8			write	**No ROM**
1	1	1	1	1	x	x	-	000 à 007
1	1	1	1	0	x	x	-	008 à 015
1	1	1	0	1	x	x	-	016 à 023
1	1	1	0	0	x	x	-	024 à 031
1	1	0	1	1	x	x	-	032 à 039
1	1	0	1	0	x	x	-	040 à 047
1	1	0	0	1	x	x	-	048 à 055
1	1	0	0	0	x	x	-	056 à 063
1	0	1	1	1	x	x	-	064 à 071
1	0	1	1	0	x	x	-	072 à 079
1	0	1	0	1	x	x	-	080 à 087
1	0	1	0	0	x	x	-	088 à 095
1	0	0	1	1	x	x	-	096 à 103
1	0	0	1	0	x	x	-	104 à 111
1	0	0	0	1	x	x	-	112 à 119
1	0	0	0	0	x	x	-	120 à 127
0	1	1	1	1	x	x	-	128 à 135
0	1	1	1	0	x	x	-	136 à 143
0	1	1	0	1	x	x	-	144 à 151
0	1	1	0	0	x	x	-	152 à 159
0	1	0	1	1	x	x	-	160 à 167
0	1	0	1	0	x	x	-	168 à 175
0	1	0	0	1	x	x	-	176 à 183
0	1	0	0	0	x	x	-	184 à 191
0	0	1	1	1	x	x	-	192 à 199
0	0	1	1	0	x	x	-	200 à 207
0	0	1	0	1	x	x	-	208 à 215
0	0	1	0	0	x	x	-	216 à 223
0	0	0	1	1	x	x	-	224 à 231
0	0	0	1	0	x	x	-	232 à 239
0	0	0	0	1	x	x	-	240 à 247
0	0	0	0	0	x	x	-	248 à 255

Dans le choix des numéros de ROM il faudra faire attention à ne pas configurer la carte avec des numéros de ROM déjà utilisés, soit par une autre interface (Silicon Disk, extension ROM, Le Hacker...), soit par le CPC+ (numéros supérieurs à 128 utilisés par la cartouche) ou encore par l'AMSDOS (ROM 7). **NE VALIDEZ JAMAIS** des ROMs ayant le même numéro.
Exemple d'utilisation des switchs, pour utiliser la RAMCARD avec les numéros 33, 35, 37 et 38 en lecture et écriture il faut placer les switchs 1, 2, 4, 5 de DIPSW1 sur ON et le switch 3 sur OFF afin de choisir la plage des numéros entre 32 et 39. On ne validera que les numéros 33, 35, 37 et 38 grâce à DIPSW2, les interrupteurs 2, 4, 6, 7 sur ON tandis que les ROMs 32, 34, 36 et 39 seront dévalidées à l'aide des inters 1, 3, 5, 8 sur OFF. Pour finir l'écriture sera rendue possible en positionnant le switch 8 de DIPSW1 sur ON. L'écriture ne concernera que les ROMs 33, 35, 37 et 38 puisque ce sont celles qui sont activées.

Remarque: Si vous utilisez une RAM de 32Ko, la RAMCARD ne dispose plus que de 2 ROMs au lieu de 8. Ces 2 ROMs sont associées à un numéro de ROM paire et un numéro impaire. Les 16 premiers Ko appartiennent aux numéros de ROM base+0, +2, +4, +6 tandis que les 16 derniers Ko appartiennent aux numéros base+1, +3, +5, +7. Tous les numéros pairs sont associés à la même ROM, tous les numéros impairs sont associés à l'autre ROM.
Si l'on configure le switch 2 et le switch 5 et que l'adresse de base est 0, on valide la ROM 1 et la ROM 4 ce qui correspond bien au 2 ROMs distinctes qu'autorise une RAM de 32Ko.
La validation de 2 ROMs paires par exemple est aussi possible mais sans grand intérêt car les 2 ROMs ainsi activées renverront les mêmes données vu qu'elles font référence à la même partie de la RAM de 32Ko.

Remarque sur le CPC+ : les ROMs de la cartouche ne souffrent pas de la contrainte de numérotation unique, il est ainsi possible de remplacer la ROM 7 (AMSDOS) du CPC+ (contenue dans la cartouche) par une ROM externe.

IV) REALISER VOS PROGRAMMES EN ROM

Si vous désirez mettre vos propres programmes en ROM, il faut que vous respectiez les contraintes des ROMs sur CPC, c'est à dire, programme qui commence en &C000 et qui finit en &FFFF, programme qui ne s'automodifie pas et surtout utiliser au maximum les vecteurs systèmes. N'essayer pas d'écrire et de lire à l'écran directement, une lecture aux adresses &C000 à &FFFF vous renverra le contenu de la ROM. Il faut également que vous sachiez comment sont organisées les ROMs au niveau logiciel sur CPC afin de les utiliser correctement.

Sur CPC on dispose de 3 types de ROMs additionnelles, les premiers octets d'une ROM sont réservés à l'identification du type de ROM. On distingue donc :

- Les ROMs de premier plan (foreground)
- Les ROMs de second plan (background)
- Les ROMs d'extension

Si le premier octet de la ROM est un 0, il s'agit d'une ROM de premier plan. Une telle ROM, lorsqu'elle est appelée, prend le contrôle de l'ordinateur et ne le rend plus, contrairement aux ROMs de second plan (premier octet de valeur 1) qui permettent de rendre le contrôle au programme principal (BASIC ou autre) après son exécution.

Dans les ROMs d'extension nous retrouvons toutes les ROMs ne contenant pas, comme type de ROM, l'octet d'identification &00 ou &01, autrement dit toutes ROMs ne contenant pas un programme directement reconnu et exécutable par le système. Il est possible d'avoir dans une telle ROM des données ou des programmes qui doivent être lancés "manuellement".

En plus de cette différentiation des ROMs au niveau logiciel, le CPC gère différemment les ROMs au niveau physique. Les ROMs de second plan doivent avoir un numéro de 0 à 15, les ROMs de premier plan peuvent avoir n'importe quel numéro. Le numéro 0 est réservé pour une ROM spéciale afin qu'elle prenne directement le contrôle du CPC avant le BASIC (le HACKER par exemple). Les ROMs de premier plan ayant un numéro supérieur à 16 ne sont reconnues que si ces ROMs existent dans un ordre croissant (ex: 16, 17, 18, 19 sont reconnues, une ROM numéro 21 ne serait pas reconnue car il n'y a pas de ROM 20 installée).

Une ROM d'extension peut se situer n'importe où puisqu'il s'agit en fait de toutes les ROMs qui ne sont pas reconnues par le CPC, soit pour une raison logicielle (type de ROM inconnu), soit pour une raison matérielle (numéro de ROM hors limite). Le CPC laisse l'utilisateur les gérer comme bon lui semble ce qui permet par exemple de gérer la RAMCARD comme un buffer mémoire.

Pour ce qui est des ROMs de premier et de second plan, le CPC attend outre l'octet de type de ROM une structure de longueur variable qui définit les RSXs de la ROM. Le fait d'avoir divisé les ROMs en plusieurs types avec une structure bien définit permet au CPC de gérer directement une table de RSXs en ROM. Vous n'avez besoin que de respecter cette structure pour installer vos RSXs, le CPC s'occupe du reste.

****

C000 = Type (0 ou 1)
C001 = Mark Number
C002 = Version Number
C003 = Modification Level

C004 = Adresse de la table des instructions (C006+N*3)

C006 = Table des N instructions Jump

C006+N*3 = Table des N noms des RSXs (16 caracteres maximum)
terminé par un octet 0

****

Le premier octet indique le type de ROM comme vu précédemment, la valeur 0 indique qu'il s'agit d'une ROM de premier plan tandis que la valeur 1 identifie une ROM de second plan. Sur cet octet seul les 2 bits de plus faible poids sont significatifs. Le bit 7 de l'octet est réservé pour la ROM interne du CPC, il doit être à 0 pour toutes les ROMs. Les autres bits n'ont pas de signification, ils ne sont pas utilisés.

Le type de ROM est donc à prendre modulo 4, le type 0 correspond donc à toutes valeurs 0, 4, 8, 12, 16, etc..., le type 1 aux valeurs 1, 5, 9, 13, etc... Les autres valeurs entre dans le cas des ROMs d'extension qui ne sont pas gérées par le CPC, la structure qui vient d'être donnée n'a plus de raison d'être dans ce cas.

Les 3 octets qui suivent le type de ROM ne servent qu'à vous renseigner sur le numéro de version et les éventuelles modifications, vous pouvez y mettre ce que vous voulez, c'est juste pour identifier les différentes versions de ROMs.

Après ces 4 octets d'identification on retrouve dans les ROMs la même structure que les programmes de RSXs en RAM, c'est à dire l'adresse de début des noms des RSXs sur 2 octets puis une table de Jump pour chaque RSX. La table des Jumps doit contenir au moins le même nombre d'entrée (et même plus si vous voulez) qu'il y a de noms de RSX.

Les noms des RSXs doivent être écrits en majuscule et la dernière lettre du nom doit avoir le bit 7 à 1. Si vous ne désirez pas que votre RSX soit appelé depuis le BASIC vous pouvez écrire en minuscules, avec des espaces ou tout autre signe. La longueur d'un nom est limitée à 16 caractères, mais le nombre de noms n'est pas limité. Pour terminer la table, celle-ci doit absolument se terminer par un octet nulle.

NB: Vous devez toujours avoir au moins 1 entrée dans la table, c'est à dire un Jump et un nom de RSX, dans le cas des ROMs de second plan il s'agira du RSX d'initialisation.
L'exemple suivant pourrait être l'en-tête d'une ROM :

```
C000  DB    &01           ;Type
C001  DB    &00           ;Mark number
C002  DB    &02           ;Version number
C003  DB    &06           ;Modification level
C004  DW    &C012         ;Adresse table des noms
C006  JP    SAUT1
C009  JP    SAUT2
C00C  JP    SAUT3
C00F  JP    SAUT4
C012  DB    "RSXU","N"+&80
C017  DB    "RSXDEU","X"+&80
C01E  DB    "RSXTROI","S"+&80
C026  DB    "RSXQUATR","E"+&80
C02F  DB    &00           ;Fin de la table
```

Le type de ROM &01 (modulo 4) qui identifie une ROM de second plan (ou ROM d'arrière plan) est sans aucun doute le plus utilisé sur CPC, il permet d'effectuer des tâches supplémentaires (RSXs) et ensuite de rendre le contrôle au programme appelant. Pour ce faire, une telle ROM doit être initialisée, soit par la ROM de premier plan utilisée ou bien par le programme d'application (programme en assembleur) avant de pouvoir être utilisée.

L'initialisation se fait par l'appel du vecteur &BCCE pour initialiser une ROM particulière ou par l'appel du vecteur &BCCB pour initialiser toutes les ROMs de 0 à 15 (0 à 7 sur 464). Sachez que par exemple le BASIC initialise toutes les ROMs en appelant le vecteur &BCCB, voilà pourquoi vous disposez des différents RSXs (comme ceux du lecteur de disquette) sous BASIC. Je vous rappelle que les ROMs de second plan porte obligatoirement un numéro de 0 à 15.

Note: Bien que le numéro 0 soit réservé pour un programme de premier plan (puisqu'il prend le contrôle du CPC automatiquement) on peut y loger une ROM de second plan. Ce qui ne manquera pas de poser un gros problème car là où la ROM s'initialise et rend le contrôle, le CPC ira exécuter, à la même adresse, le programme qui doit remplacer le BASIC et qui lui ne doit pas rendre la main. Si vous n'avez pas compris ce n'est pas grave retenez seulement qu'il ne faut pas utiliser la ROM 0.

Le système initialise une ROM de second plan en appelant le premier RSX (Jump situé en C006), c'est à ce moment que la ROM se réserve la place qu'elle désire, en diminuant le HIMEM. Attention certains programmes ne tournent pas lorsque des ROMs sont présentes du fait qu'ils n'aiment pas que le HIMEM diminue.

Lors de l'initialisation (appel du premier RSX) le registre HL contient la valeur du HIMEM, la ROM peut la diminuer si elle doit se réserver de la place. Les registres BC et A sont indéterminés (ils peuvent servir dans le RSX comme registre de travail) tandis que DE contient une valeur (normalement l'adresse basse en mémoire lowmem = &0040) qui sera transmise à toutes les ROMs, si l'une d'elle modifie DE, la ROM suivante aura cette valeur modifiée. Les registres d'index ne sont pas utilisés dans ces routines. Dans tous les cas le RSX d'initialisation devra mettre le flag carry avant le retour pour que le système prenne en considération cette ROM (initalisation ROM ok).

NB: La valeur du HIMEM est diminuée de 4 octets minimum pour chaque ROM initialisée, en effet le système se réserve 4 octets en mémoire pour une utilisation interne, les fameux 4 octets "KERNAL" des RSXs en RAM. De ce fait si vous disposez de 5 ROMs de second plan vous perdez 20 octets sous BASIC, de même si vous ne disposez que d'une ROM qui se réserve 256 octets en RAM, vous perdez en fait 260 octets sous BASIC.

Les ROMs de premier plan ne prennent aucune place en mémoire, elles n'ont pas besoin d'initialisation. Lorsqu'une ROM de premier plan est appelé tout le système est réinitialisé, la ROM ne rendra plus la main et peut donc utiliser toute la RAM disponible mais pour utiliser une ROM de second plan (les fonctions disque en ROM 7 par exemple) il faudra qu'elle active la ou les ROMs d'arrière plan nécessaires (par appel du vecteur &BCCB ou &BCCE).

Une ROM de premier plan est idéale pour contenir un programme d'application complet. La seule ROM tirant avantage actuellement de cette propriété c'est LE HACKER qui porte en plus le numéro de ROM 0. La ROM 0, si elle est détectée, prend le contrôle de l'ordinateur avant même le BASIC LOCOMOTIVE.

Une ROM de premier plan comme les ROMS de second plan peuvent prendre n'importe quel numéro entre 0 et 15. Les ROMs de premier plan peuvent en plus porter un numéro supérieur à 16, à cet effet il faut que les numéros de ROMs à partir de 16 soient consécutifs. La recherche d'une ROM de premier plan à partir d'un numéro supérieur à 15 s'arrête lorsqu'il y a un "trou" (en fait lorsque la ROM interne est détectée à l'aide du bit 7 du type de ROM).

V) TOUS SUR LES APPELS DE RSXs EN ROM

Voici comment appeler une ROM externe sur CPC à partir d'un programme, si vous désirez appeler un programme en ROM à partir du BASIC vous utiliserez la commande RSX adéquate précédée de la barre verticale **"|"** pour un clavier QWERTY (SHIFT et @) ou le u accentué **"ù"** pour un clavier AZERTY, la commande sera automatiquement exécutée et la ROM rendra ou non la main au BASIC suivant le type de ROM (second ou premier plan).

En assembleur vous pouvez utiliser le vecteur KL_FIND_COMMAND (&BCD4) pour appeler un RSX :

```
            LD    HL,RSXNAME
            CALL  &BCD4              ;KL_FIND_COMMAND
            . . . . .
RSXNAME     DB    "PROG","1"+&80
```

Dans cet exemple le RSX PROG1 est appelé, on remarquera que le bit 7 du dernier caractère du nom est à 1 pour indiquer la fin du RSX. Le vecteur &BCD4 fonctionne différemment suivant le type de ROM, avec une ROM de premier plan le RSX trouvé est automatiquement lancé, le système est par ailleurs initialisé. Il n'y a pas de retour après le vecteur &BCD4. Pour une ROM de second plan, l'appel du vecteur &BCD4 retourne en HL l'adresse en ROM, en C le numéro de ROM du RSX et le CARRY est à 1 si le RSX a été trouvé. Le CARRY est à 0 pour indiquer que le RSX n'existe pas (ROM non initialisé ou nom incorrect). Après avoir pris connaissance de l'adresse et du numéro de ROM, le RSX est exécuté en appelant le vecteur KL_FAR_PCHL (&001B) ou par un RST &0018.

```
            LD    HL,&C009
            LD    C,&04
            CALL  &001B              ;Appel de la ROM
```

Le vecteur &001B attend en entrée l'adresse du programme en ROM en HL et le numéro de ROM en C, vous pouvez donc lancer le RSX tout de suite après un CALL &BCD4 :

```
            LD    HL,RSXNAME
            CALL  &BCD4              ;KL_FIND_COMMAND
            RET   NC                 ;RSX pas ok
            XOR   A
            CALL  &001B              ;KL_FAR_PCHL
            RET
RSXNAME     DB    "PROG","1"+&80
```

Pourquoi ce XOR A avant le CALL &001B ? Eh bien parce que le passage de paramètre à un RSX se fait à travers le registre A et IX. Le registre A contient le nombre de paramètres tandis que IX contient la liste des paramètres (valeurs sur 16 bits), dans l'ordre inverse, le dernier paramètre étant le premier dans la table. Le registre DE doit contenir la valeur du dernier paramètre (première entrée de la table) pour rester compatible avec le BASIC.

Les paramètres sont tous sur 16 bits, pour une chaîne de caractère (nom de fichier par exemple) c'est l'adresse du descripteur de chaine qui est transmis. Le descripteur se compose d'un octet de longueur suivi de l'adresse 16 bits où sont stockés les caractères lettre par lettre. Voici l'exemple de la commande |REN ou ùREN :

```
            LD      HL,RSXNAME
            CALL    &BCD4
            RET     NC
            LD      A,2             ;2 paramètres pour le RSX
            LD      IX,TABLE        ;Table des 2 paramètres
            CALL    &001B
            RET
RSXNAME     DB      "RE","N"+&80
TABLE       DW      PARAM2          ;2ème paramètre
            DW      PARAM1          ;1er paramètre
PARAM1      DB      11              ;Longueur du nom 1
            DW      NOMNOUV         ;Adresse du nom 1
PARAM2      DB      10              ;Longueur du nom 2
            DW      NOMANCI         ;Adresse du nom 2
NOMNOUV     DB      "NOUVEAU.DAT"
NOMANCI     DB      "ANCIEN.BAK"
```

Une autre méthode pour appeler une ROM consiste à appeler le RST &18, qui attend l'adresse et le numéro de ROM en mémoire et non plus dans HL et C.

```
            RST     &18
            DW      TABLE
            RET
TABLE       DW      &C009           ;Adresse routine en ROM
            DB      &04             ;Numéro de ROM
```

Le RST &18 est suivi de l'adresse où sont stockés les 3 octets précédemment cités (adresse sur 16 bits et numéro de ROM sur 8 bits).
Une dernière méthode consiste à appeler le vecteur &0023 qui attend dans HL l'adresse de la même table de 3 octets.

```
        LD    HL,TABLE
        CALL  &0023
        RET
TABLE   DW    &C009          ;Adresse routine en ROM
        DB    &04            ;Numéro de ROM
```

Le RST &18, le CALL &001B et le CALL &0023 permettent d'appeler n'importe quel programme en ROM (dans les 252 numéros de ROMs possibles) et pas seulement des RSXs, vous pouvez par exemple appeler le programme situé à l'adresse &DE53 en ROM 61.

L'avantage du RST &18 réside dans le fait qu'il n'utilise aucun registre (contrairement au CALL &001B et au CALL &0023), les registres restent donc disponibles pour transmettre différents paramètres comme dans l'exemple suivant :

```
        LD    HL,PARAM1
        LD    DE,PARAM2      ;Différents paramètres à envoyer
        LD    A,PARAM3       ;à la routine &DE53 en ROM 61
        LD    IX,PARAM4      ;dans les registres
        RST   &18
        DW    TABLE
        RET                  ;RET ou suite du programme
TABLE   DW    &DE53
        DB    61             ;Adresse + numéro de ROM
```

Le registre HL étant nécessaire pour l'envoi de paramètre, l'utilisation du vecteur &001B ou du vecteur &0023 est impossible.

Ces vecteurs sont très puissants, ils permettent d'appeler n'importe quelle ROM et de revenir à l'ancienne configuration mémoire (ROM ou RAM), ils peuvent donc être utilisés sans danger dans tout programme en ROM. Autre avantage décisif, lorsqu'ils appellent une ROM de second plan, ils transmettent dans IY l'adresse de la zone de donnée réservée lors de l'initialisation. La valeur d'origine de IY est restituée en sortie de ROM.
Le registre IY permet de gérer les données de votre programme en ROM sans se soucier de leurs adresses, il ne pourra donc pas vous servir comme registre pour l'envoi de paramètres. Par contre vous pourrez l'utiliser sans danger dans votre ROM car sa valeur d'origine est toujours restituée au programme appelant.

Le RST &18 et les vecteurs &001B et &0023 considèrent, lorsqu'un numéro de ROM est compris entre 252 et 255, qu'il ne s'agit pas d'une ROM mais d'une combinaison de configuration mémoire. Il est ainsi possible d'appeler n'importe quel programme en RAM ou ROM, sans se soucier des commutations mémoire. Voilà pourquoi le système ne gère que 252 ROMs alors qu'électroniquement 256 ROMs sont possibles.

Le CPC recherche un RSX avec le vecteur &BCD4 de la façon suivante. Un pointeur système lui indique où se trouvent les 4 octets de "KERNAL" en RAM. Ces 4 octets que l'on retrouve aussi lorsque l'on installe des RSXs en RAM ont la signification suivante, les 2 premiers octets forment un pointeur sur les 4 prochains octets réservés "KERNAL". Les 2 derniers octets référencent la table des RSXs en RAM (pour les RSXs installés avec &BCD1) ou le numéro de ROM pour les ROMs de second plan.

Le premier pointeur confère aux RSXs leur niveau de priorité, la recherche de RSX se fait dans l'ordre chronologique inverse de l'installation. Les RSXs dernièrement installés (les RSXs installés en RAM par exemple) sont les premiers testés. Le chaînage des 4 octets permet de reconstituer l'ordre d'installation des RSXs et donc l'ordre de priorité, la recherche s'arretant dès qu'il y a concordance des noms. Si deux RSX du même nom existent celui qui sera exécuté sera celui qui aura été installé le dernier.

Les 2 derniers octets indiquent l'adresse en RAM de la table des RSXs (adresse transmise lors de l'appel du vecteur &BCD1) avec l'octet fort non nul. Si l'octet fort est nul l'octet faible indique le numéro de ROM de second plan où continuer la recherche, l'adresse de la table des RSXs étant toujours en &C004 pour les ROMs. Ici aussi l'ordre des ROMs est déterminé par les 2 premiers octets "KERNAL".

Le vecteur &BCCB initialise les ROMs de 15 à 0, la recherche de RSX s'effectue dans l'ordre chronologique inverse, c'est donc la ROM 0 qui a la plus forte priorité et la ROM 15, la plus faible
Lorsque la recherche d'un RSX a été infructueuse dans les ROMs de second plan initialisées (lorsque les 2 premiers octets du "KERNAL" sont nuls), la recherche de RSX se poursuit dans les ROMs de premiers plans de 0 à 15 dans un premier temps, et de 16 jusqu'à ce que la ROM interne soit détectée dans un deuxième temps.
Lorsque le RSX est trouvé en ROM de premier plan, le CPC est réinitialisé (réinitialisation identique au vecteur &BD16) avant d'être exécuté.

Voila vous savez tout sur l'utilisation des RSXs par le CPC, je n'ai pas précisé mais la recherche s'effectue caractère par caractère avec distinction minuscule/majuscule (il n'y a que le BASIC qui convertit les minuscules en majuscules).
Il ne nous reste plus qu'à passer en revue les vecteurs systèmes responsable des commutations mémoires RAM/ROM pour tout savoir sur la gestion des ROMs sur CPC.

VI) LA COMMUTATION RAM/ROM

Pour la gestion de ROM le système dispose de plusieurs vecteurs situés aux adresses &B900 à &B91E ainsi que quelques RSTs ou vecteurs dans la zone &0000 à &0040. Commençons par le RST &18 qui a déjà été décrit dans le chapitre précédent.

- **RST &18** = Appel d'une ROM. L'adresse qui suit le RST indique où se trouve les 3 octets contenant l'adresse de la routine et le numéro de ROM. Les registres A, BC, DE, HL et IX peuvent servir pour l'envoi ou le retour de paramètres. Le registre IY peut également servir comme paramètre lorsque le no de ROM correspond à une commutation mémoire (252, 253, 254, 255).

Remarque: lors d'un appel d'une ROM de 0 à 251, IY est sauvegardé et est restauré en sorti de ROM. Dans le programme en ROM, IY garde sa valeur d'origine pour une ROM de 16 à 251 ou prend la valeur de la zone de donnée réservée par une ROM de second plan pour une ROM de 0 à 15. Si à ce numéro ne correspond pas une ROM de second plan le registre IY est indéterminé.

- **CALL &001B** = Appel d'une routine en ROM dont l'adresse est contenue dans HL et dont le numéro de ROM est spécifié dans C. Le fonctionnement et la remarque sur le RST &18 reste valable pour ce vecteur.

- **CALL &0023** = Appel d'une routine en ROM dont l'adresse et le numéro de ROM sont stockés en RAM sous forme de 3 octets comme pour le RST &18. HL contient l'adresse de ces 3 octets. Même remarque que pour le vecteur &001B.

Un vecteur bien pratique, le vecteur &B912 (KL_CURR_SELECTION) renvoi dans A le numéro de ROM en cours. Ceci vous sera bien utile si vous avez un programme qui ne tient pas sur 16Ko et qui doit appeler une autre ROM car vous ne savez jamais d'avance quel est le numéro absolu d'une ROM.

Une autre méthode pour appeler jusqu'à 4 ROMs consécutifs c'est le RST &10, l'adresse en ROM - &C000 doit être transmise en mémoire sous forme 16 bits juste après le RST &10. Les bits 14 et 15 désignent sous forme binaire laquelle des 4 ROMs consécutifs doit être activées. Exemple, votre programme de base est en ROM 3, vous voulez appeler une partie du programme en ROM 4 à l'adresse &C32F :

```
RST &10
DW &432F                ;Adresse &32F  de la ROM courante+1
```

Un simple RET dans la nouvelle ROM appelée permet de revenir à la ROM d'origine. Avec le RST &10 vous n'avez absolument pas besoin de connaître le numéro de ROM courant, il vous suffit de savoir que vos modules ont des numéros consécutifs. De plus dans notre exemple, dans la ROM 4 à l'adresse &C32F on peut appeler une autre ROM avec encore un RST &10, la ROM de base étant toujours la ROM 3.

- **RST &10** = Appel d'un programme dans une des 4 ROMs consécutifs à la ROM de base. Les 2 octets qui suit le RST &10 indique l'adresse en ROM haute. Les 2 bits supérieurs contiennent le numéro de ROM (0 à 3) à ajouter à la ROM de base.

- **CALL &0013** = Identique au RST &10 sauf que les 2 octets sont contenus dans HL. Les mêmes remarques que pour les vecteurs &001B et &0023 ou le RST &18 restent valables pour le CALL &0013 ou le RST &10 sauf que les combinaisons de commutation RAM/ROM (252, 253, 254, 255) ne sont pas gérées. Ce sont les seuls vecteurs capable de gérer les 256 numéros de ROM.

- **CALL &B912** = Retourne le numéro de ROM en cours dans A. Pas de registres utilisés.

Les vecteurs &B900 à &B90C permettent de commuter la ROM ou RAM en mémoire basse (&0000 à &3FFF) et haute (&C000 à &FFFF). La ROM basse correspond à la ROM système tandis que la ROM haute correspond à une ROM d'extension ou à la ROM BASIC si la ROM d'extension n'existe pas.

- **CALL &B900** = Commutation ROM haute. Retour dans l'Accu de l'état ROM/RAM avant la commutation (état du Gate Array). Les autres registres ne sont pas utilisés.

- **CALL &B903** = Commutation RAM haute. Retour dans l'Accu de l'état ROM/RAM avant la commutation (état du Gate Array). Les autres registres ne sont pas utilisés.

- **CALL &B906** = Commutation ROM basse. Retour dans l'Accu de l'état ROM/RAM avant la commutation (état du Gate Array). Les autres registres ne sont pas utilisés.

- **CALL &B909** = Commutation RAM basse. Retour dans l'Accu de l'état ROM/RAM avant la commutation (état du Gate Array). Les autres registres ne sont pas utilisés.

- **CALL &B90C** = Remet commutation haute et basse suivant valeur de l'Accu (utilisation en complément de &B900 à &B909). L'Accu contient en retour le nouvel état ROM/RAM (état du Gate Array). Les autres registres ne sont pas utilisés.

Les dernières fonctions de commutation RAM/ROM de la mémoire haute et basse sont contenues sous forme de RST : les vecteurs &0018, &001B ou &0023 que l'on a déjà vu, le RST &08 et le CALL &000B qui permettent l'exécution d'une routine en mémoire basse (&0000 - &3FFF) tout en commutant la RAM ou la ROM, à l'aide des bits 14 et 15 de l'adresse, en mémoire basse ou haute. Le dernier RST, le RST &28 permet quant à lui d'exécuter une routine à n'importe quelle adresse en mémoire tout en commutant la ROM basse.

Combinaisons des numéros réservés pour la commutation RAM/ROM

	Mémoire Basse	Mémoire Haute
252 (00)	ROM	ROM
253 (01)	RAM	ROM
254 (10)	ROM	RAM
255 (11)	RAM	RAM

(Numéro de ROM pour le RST &18, le CALL &001B et le CALL &0023 configuration des bits 14 et 15 pour le RST &08 et le CALL &000B)

- **RST &08** = Appel d'une routine en mémoire basse (&0000 à &3FFF) à l'adresse spécifiée par les 2 octets qui suivent le RST. Les 2 bits supérieurs indiquent la commutation ROM/RAM (voir tableau page précédente). L'état RAM/ROM est restauré en sortie du RST. Tous les registres peuvent servir à l'envoi ou au retour de paramètres. Attention il n'y a pas de retour après le RST, le RST &08 se comporte comme un JUMP.

- **CALL &000B** = Appel d'une routine en mémoire basse dont l'adresse est spécifiée dans HL. Les 2 bits supérieurs contiennent la commutation ROM/RAM à effectuer avant d'exécuter la routine. L'état d'origine est restauré en fin de routine. A part HL tous les autres registres peuvent servir à l'envoi ou à la réception de données avec la routine exécutée.

- **RST &28** = Appel d'une routine en mémoire dont l'adresse est spécifiée par les 2 octets qui suivent le RST. Equivalent à un JUMP à l'adresse indiquée avec commutation au préalable de la ROM basse. Attention, l'état RAM/ROM n'est pas restauré en sortie du RST, la RAM basse est activée en sortie et ce quelque soit l'état d'origine. Tous les registres peuvent servir pour l'envoi ou le retour de paramètres.

Lorsque l'on fait tourner un programme en ROM, on a parfois besoin d'accéder à la RAM sous-jacente (lecture en mémoire écran par exemple), le système offre à cet effet un RST et 2 vecteurs systèmes pour effectuer une lecture ou un transfert de donnée quelque soit l'état ROM du CPC.

- **CALL &B91B** = Effectue un LDIR dans la RAM quelque soit l'état de connexion des ROMs. Les registres sont modifiés de la même façon que pour l'exécution de l'instruction LDIR.

- **CALL &B91E** = Effectue un LDDR dans la RAM quelque soit l'état de connexion des ROMs. Les registres sont modifiés de la même façon que pour l'exécution de l'instruction LDDR.

- **RST &20** = Lit un octet en RAM quelque soit l'état de connexion des ROMs. HL doit contenir l'adresse où lire la donnée, en retour A contient la donnée lue dans la RAM. Les registres ne sont pas modifiés, équivalent à l'exécution de l'instruction LD A,(HL).

Les derniers vecteurs systèmes sont responsable de la commutation d'une ROM haute quelconque.

- **CALL &B90F** = Active la ROM haute et sélectionne la ROM dont le numéro est contenu dans C. En retour A et B contiennent l'état des ROMs (état Gate Array) avant la commutation tandis que C contient le numéro de l'ancienne ROM connectée. Les autres registres ne sont pas modifiés. Ce vecteur permet d'activer les ROMs de 0 à 255.

- **CALL &B915** = Test ROM haute dont le numéro est dans C. En retour A et B contiennent l'octet de type de ROM tandis que HL contient le numéro de version de la ROM (H=Version Number et L=Mark Number). Les autres registres ne sont pas modifiés. Les numéros de ROM de 0 à 255 sont gérés, ce vecteur permet donc d'accéder aux ROMs 252 à 255.

- **CALL &B918** = Restaure ancienne configuration ROM contenue dans BC. Ce vecteur s'utilise associé au vecteur &B90F, la ROM haute de numéro C est sélectionnée et l'état RAM/ROM conservé dans B est réactivé. En retour C contient le numéro de ROM qui était activé, le registre B est modifié et les autres registres sont inchangés.

Pour terminer sur les vecteurs voici le rappel des vecteurs &BCCB, &BCCE responsable de l'initialisation des ROMs de second plan et du vecteur &BCD4 responsable de la recherche des RSXs.

- **CALL &BCCB** = Initialise toutes les ROMs de second plan (de 15 à 0). En entrée ce vecteur attend dans HL l'adresse Himem (mémoire la plus haute en RAM) où les ROMs de second plan peuvent se réserver de la mémoire et en DE l'adresse Lowmem (mémoire la plus basse en RAM). Les valeurs standard sont &ABFF pour HL et &0040 pour DE. Les registres A et BC sont utilisés par ce vecteur, IX et IY ne sont pas modifiés.

- **CALL &BCCE** = Initialise la ROM de second plan de numéro contenu dans C (entre 0 et 15). HL et DE ont la même signification qu'avec le vecteur &BCCB. De même les registres A et BC sont utilisés tandis que IX et IY ne sont pas modifiés.

- **CALL &BCD4** = Recherche le RSX dont le nom est stocké en RAM et qui est pointé par HL. Si le RSX est trouvé le Carry est mis, HL contient l'adresse et C le numéro de ROM de la routine associée au RSX. Dans le cas contraire le Carry est annulé. Si le RSX trouvé est en ROM de premier plan le système est réinitialisé et le RSX est exécuté (pas de retour après le CALL &BCD4). Tous les registres sauf IX et IY sont utilisés, en cas de retour les valeurs de ces registres sont indéterminées.

Pour être réellement complet encore faut-il que je vous donne les adresses des ports d'entrées/sorties responsable de la gestion mémoire ROM/RAM. Utilisez les ports d'entrées/sorties qu'avec parcimonie, utilisez plutôt les routines systèmes.
Tout d'abord voyons le Gate Array, adresse de port &7Fxx, qui gère (entre autres) la commutation ROM/RAM, pour cela les bits 7 et 6 doivent être respectivement à 1 et à 0, les bits 1 et 0 définissent le mode écran (MODE 0, 1 ou 2) tandis que les bits 3 et 2 détermine l'état ROM/RAM pour la mémoire haute et basse respectivement (plusieurs routines systèmes vous renvoient cet octet du Gate Array comme octet d'état).

```
Bit 3 :  0=ROM HAUTE    1=RAM HAUTE
Bit 2 :  0=ROM BASSE    1=RAM BASSE
```

Exemple 1 :

```
LD    BC,&7F8D          ;MODE 1 et
OUT   (C),C             ;RAM HAUTE et RAM BASSE
```

Exemple 2:

```
LD    BC,&7F86          ;MODE 2 et
OUT   (C),C             ;ROM HAUTE ET RAM BASSE
```

Le numéro de ROM est envoyé sur 8 bits sur le port &DFxx, si le numéro de ROM n'existe pas c'est la ROM BASIC qui apparaît aux adresses &C000-&FFFF.

```
LD    BC,&7F84          ;MODE 0 et
OUT   (C),C             ;ROM HAUTE ET RAM BASSE
LD    BC,&DF07          ;Numéro de ROM = 7
OUT   (C),C             ;ROM 7 accessible à partir de &C000
```

N'utilisez la commutation des ROMs avec les ports d'entrées/sorties que si vous êtes sûr de vous (ne commutez pas n'importe quoi, n'importe où, n'importe quand). Faites ces commutations hors interruptions, n'utiliser plus les vecteurs du système (qui appellerons à un moment ou à un autre une routine de commutation mémoire du système).

Ne commutez pas la RAM écran si votre programme de commutation est en ROM (exécutez la RAM écran ne donne rien de bon). Soyez très vigilant avec les ports d'entrées/sorties, une utilisation sans précautions et vous planterez à coup sûr l'ordinateur. A moins que vous n'en ayez l'utilité utilisez plutôt les fonctions systèmes pour la commutation mémoire.

Avec le port &DFxx (qui est sur 8 bits) vous pouvez accéder aux ROMs 252, 253, 254 et 255, par exemple pour accéder à la ROM 253, effectué une écriture sur le port &DFxx avec la valeur &FD (253 en décimal).

Exemple 1:

```
LD    BC,&7F85      ;MODE 1 et
OUT   (C),C         ;ROM HAUTE ET RAM BASSE
LD    BC,&DF
LD    A,&FD         ;Numéro de ROM = 253
OUT   (C),A         ;ROM 253 accessible à partir de &C000
```

Exemple 2:

```
LD    BC,&7F82      ;MODE 2 et
OUT   (C),C         ;ROM HAUTE ET ROM BASSE
LD    BC,&DFFD      ;Numéro de ROM = 253
OUT   (C),C         ;ROM 253 accessible à partir de &C000
```

VII) L'ECRITURE DANS LA RAMCARD

Pour écriture dans la RAMCARD il suffit d'effectuer une écriture en RAM (de &C000 à &FFFF) avec le switch d'écriture validé. L'écriture aura lieu si le numéro de ROM sélectionné sur le port &DFxx correspond bien à une ROM de la RAMCARD et que son switch sur la carte est validé (ROM active).

Peu importe l'état de connexion de la mémoire haute (ROM ou RAM), une écriture atterrira toujours en RAM interne et aussi en RAM externe (RAMCARD). La sélection RAM/ROM haute par le Gate Array n'est valable que pour la lecture et en aucun cas pour l'écriture.

Exemple 1:

```
LD    BC,&7F84      ;MODE 0 et
OUT   (C),C         ;ROM HAUTE ET RAM BASSE
LD    BC,&DF10      ;ROM = 16 accessible à partir de &C000
OUT   (C),C         ;en lecture et écriture (si switch ON).
```

Exemple 2:

```
LD    BC,&7F8E      ;MODE 2 et
OUT   (C),C         ;RAM HAUTE ET RAM BASSE
LD    BC,&DF10      ;ROM = 16 accessible à partir de &C000
OUT   (C),C         ;en écriture seulement (si switch ON)
                    ;en lecture on lit la RAM.
```

Dès qu'un numéro de ROM valide pour la RAMCARD est détecté et que le switch d'écriture est positionné sur ON, l'écriture dans la ROM est possible. Le seul moyen d'éviter une écriture dans la RAMCARD c'est de changer le numéro de ROM haute.

Par exemple si vous avez besoin de faire une écriture en RAM centrale aux adresses &C000 à &FFFF (écran), changez le numéro de ROM avec un OUT &DFxx avec une valeur quelconque non utilisée par la RAMCARD, exemple 0, 7, 255, etc...
Vous pouvez également utiliser les vecteurs systèmes &B90F et &B918 pour sélectionner une ROM et ensuite revenir à la ROM d'origine.

```
LD    C,&10         ;ROM 16 de la RAMCARD
CALL  &B90F         ;Active ROM et sauvegarde
PUSH  BC            ;ancienne configuration
LD    A,&53
LD    (&D8F2),A     ;Ecriture dans la RAMCARD
...                 ;Suite de l'écriture
POP   BC
CALL  &B918         ;Restaure config. ROM
...                 ;Suite du programme (affichage)
```

Le seul inconvénient de cette méthode, réside dans le fait qu'une programmation de la RAMCARD apparaît à l'écran. Ceci peut être bien utile, voir le programme RAMCARD.BAS pour la programmation de ROM, pour visualiser qu'une écriture s'est bien déroulée. Il est par contre très désagréable de voir l'écriture à l'écran lorsque l'on réalise un buffer RAM.

Une méthode pour masquer l'écriture dans la RAM centrale consiste à commuter une bank d'extension mémoire en &C000 à l'aide d'un OUT &7Fxx avec la valeur &C1 pour toutes les phases d'écriture en RAMCARD. Pour l'affichage on reviendra en RAM centrale à l'aide du OUT &7Fxx et de la valeur &C0 sans oublier de changer le numéro de ROM. L'affichage n'est plus perturbé par l'écriture dans la RAMCARD, c'est la bank mémoire qui est perturbée et qui sert de mémoire tampon pour toutes les écritures.

Cette méthode n'est réalisable que si l'on dispose d'un 6128 ou d'un 464, 664 avec extension mémoire. Une autre méthode consiste à déplacer l'écran à une autre adresse, en &4000-&7FFF (en utilisant le vecteur &BC08 par exemple). La zone &C000-&FFFF ainsi libérée peut servir de mémoire tampon et on peut utiliser sans souci les vecteurs systèmes pour l'affichage.
Dans tous les cas on arrive toujours au même point, il faut sacrifier 16Ko en mémoire, que se soit en bank d'extension ou en mémoire centrale, pour recevoir l'écriture dans la RAM CPC qui ne peut pas être évitée. Sous certaines conditions ont peut récupérer ces 16Ko de mémoire si on réalise une application qui utilise la RAMCARD comme un buffer (un copieur disc par exemple).
La méthode consiste à remplir la mémoire de la RAMCARD avec les commutations ROM puis de remplir en dernier la RAM entre &C000-&FFFF (en ayant commuté une ROM qui n'appartient pas à la RAMCARD). Ainsi le contenu de la RAMCARD ne sera pas modifié et les données écrites en RAM haute correspondent au dernier bloc de 16Ko du buffer tant que l'on écrira plus en RAMCARD.
Pour la lecture il suffira de commuter les ROMs pour lire la RAMCARD et de commuter la RAM HAUTE pour lire la RAM entre &C000-&FFFF, le numéro de ROM courant n'ayant aucune importance en lecture.

VIII) LES LOGICIELS DISPONIBLES

La RAMCARD a été conçue pour permettre la réalisation de logiciels en ROM, je vous encourage donc vivement à développer vos propres softs en ROM et à les distribuer, je me ferai un plaisir de vous aider dans cette voie. Je tiens à préciser que certains logiciels commerciaux ne doivent pas être installer sur la RAMCARD sans consentement préalable des ayant droits (certaines ROM sont libre d'utilisation sur émulateur mais ne le sont pas sur CPC).

Je vous propose 2 petits logiciels avec leur source dans cette notice qui vous seront utile pour tester et programmer votre RAMCARD.
Le premier logiciel nommé ROMTEST.BIN permet de tester l'existence de toutes les ROMs (0 à 255) sur CPC. Le numéro des ROMs présentes sur votre CPC est affiché et rafraîchi au fur et à mesure. Une activation (ou désactivation) d'une ROM apparaît (ou disparaît) au pire dans les 5 secondes (temps de rafraîchissement). Si l'écriture est possible dans une ROM (switch écriture de la RAMCARD basculé), le numéro de cette ROM apparaît en rouge.
Vous trouverez dans les pages suivantes le programme ROMTEST.BAS à taper pour générer le programme ROMTEST.BIN (programme exécutable) ainsi que son source assembleur commenté.

Le second logiciel permet la programmation d'une ROM dans la RAMCARD à partir d'un fichier 17Ko binaire. Pour cela il vous suffit de taper le programme RAMCARD.BAS et de le sauvegarder. Vous trouverez également dans les pages qui suivent la routine assembleur commentée contenue dans les lignes DATA qui utilise le switching d'une ROM bidon pour pouvoir écrire en RAM écran tout en ne détruisant pas la ROM programmée.

Source de ROMTEST.BIN :

```
A000  3E 40         LD    A,&40
A002  CD 08 BC      CALL  &BC08       ;Ecran en &4000
A005  3E 01         LD    A,&01
A007  CD 0E BC      CALL  &BC0E       ;Mode 1
A00A  3E 01         LD    A,&01
A00C  CD 63 BB      CALL  &BB63
A00F  01 00 00      LD    BC,&0000
A012  CD 38 BC      CALL  &BC38       ;Border noir
A015  01 00 00      LD    BC,&0000
A018  AF            XOR   A
A019  CD 32 BC      CALL  &BC32       ;Fond noir
A01C  01 18 18      LD    BC,&1818
A01F  3E 01         LD    A,&01
A021  CD 32 BC      CALL  &BC32       ;Ecriture jaune
A024  01 06 06      LD    BC,&0606
A027  3E 03         LD    A,&03
A029  CD 32 BC      CALL  &BC32       ;Couleur rouge
A02C  CD BA BB      CALL  &BBBA       ;GRA INITIALISE
A02F  3E 03         LD    A,&03       ;Superposition
A031  CD 59 BC      CALL  &BC59
A034  11 02 00      LD    DE,&0002    ;Place curseur
A037  21 83 01      LD    HL,&0183
A03A  CD C0 BB      CALL  &BBC0
A03D  3E 03         LD    A,&03
A03F  CD DE BB      CALL  &BBDE       ;Couleur rouge
A042  21 00 A1      LD    HL,&A100
A045  7E            LD    A,(HL)
A046  23            INC   HL
A047  B7            OR    A
A048  28 05         JR    Z,&A04F     ;Message
A04A  CD 5A BB      CALL  &BB5A
A04D  18 F6         JR    &A045
A04F  11 00 00      LD    DE,&0000
A052  21 85 01      LD    HL,&0185    ;Place curseur
A055  CD C0 BB      CALL  &BBC0
A058  3E 01         LD    A,&01
A05A  CD DE BB      CALL  &BBDE       ;Couleur jaune
A05D  21 00 A1      LD    HL,&A100
A060  7E            LD    A,(HL)
A061  23            INC   HL
A062  B7            OR    A
A063  28 05         JR    Z,&A06A     ;En superposée
A065  CD 5A BB      CALL  &BB5A
A068  18 F6         JR    &A060
A06A  AF            XOR   A
A06B  CD 59 BC      CALL  &BC59       ;Mode normal
A06E  11 00 00      LD    DE,&0000
A071  21 3F 01      LD    HL,&013F    ;Coordonnée
```

```
A074  3E 00          LD    A,&00
A076  ED 53 00 A1    LD    (&A100),DE   ;Coordonnée X
A07A  22 02 A1       LD    (&A102),HL   ;Coordonnée Y
A07D  32 04 A1       LD    (&A104),A    ;No ROM
A080  F3             DI                 ;Plus de systeme
A081  06 DF          LD    B,&DF
A083  ED 79          OUT   (C),A        ;Active No ROM
A085  D9             EXX
A086  79             LD    A,C
A087  E6 F7          AND   &F7          ;Gate Array
A089  ED 79          OUT   (C),A        ;ROM haute oui
A08B  3A 00 C0       LD    A,(&C000)    ;Type de ROM
A08E  ED 49          OUT   (C),C        ;ROM haute
A090  D9             EXX
A091  FE 80          CP    &80
A093  28 20          JR    Z,&A0B5      ;Rom BASIC=&80
A095  2F             CPL
A096  32 00 C0       LD    (&C000),A    ;Ecriture en RAM
A099  47             LD    B,A          ;La RAMCARD est
A09A  D9             EXX                ;restée activée
A09B  79             LD    A,C          ;par le OUT &DF
A09C  E6 F7          AND   &F7
A09E  ED 79          OUT   (C),A
A0A0  3A 00 C0       LD    A,(&C000)    ;Relire la ROM
A0A3  ED 49          OUT   (C),C
A0A5  D9             EXX
A0A6  B8             CP    B
A0A7  20 08          JR    NZ,&A0B1     ;Donnée écrite ?
A0A9  2F             CPL
A0AA  32 00 C0       LD    (&C000),A    ;Donnée origine
A0AD  3E 03          LD    A,&03        ;Rouge = RAMCARD
A0AF  18 05          JR    &A0B6
A0B1  3E 01          LD    A,&01        ;Jaune = ROM ok
A0B3  18 01          JR    &A0B6
A0B5  AF             XOR   A            ;Rien = pas ROM
A0B6  FB             EI
A0B7  CD DE BB       CALL  &BBDE
A0BA  ED 5B 00 A1    LD    DE,(&A100)
A0BE  2A 02 A1       LD    HL,(&A102)   ;Positionnement
A0C1  CD C0 BB       CALL  &BBC0
A0C4  3A 04 A1       LD    A,(&A104)
A0C7  F5             PUSH  AF           ;Poids fort
A0C8  0F             RRCA
A0C9  0F             RRCA
A0CA  0F             RRCA
A0CB  0F             RRCA
A0CC  CD F3 A0       CALL  &A0F3
A0CF  F1             POP   AF
A0D0  CD F3 A0       CALL  &A0F3        ;Poids faible
A0D3  2A 00 A1       LD    HL,(&A100)
```

```
A0D6  11 28 00       LD    DE,&0028     ;X=X+40
A0D9  19             ADD   HL,DE
A0DA  EB             EX    DE,HL
A0DB  2A 02 A1       LD    HL,(&A102)
A0DE  3A 04 A1       LD    A,(&A104)    ;No ROM + 1
A0E1  3C             INC   A
A0E2  28 8A          JR    Z,&A06E      ;Recommencer
A0E4  47             LD    B,A
A0E5  E6 0F          AND   &0F
A0E7  78             LD    A,B          ;Fin ligne ?
A0E8  20 8C          JR    NZ,&A076
A0EA  11 00 00       LD    DE,&0000     ;X=0
A0ED  01 F0 FF       LD    BC,&FFF0     ;Y=Y-16
A0F0  09             ADD   HL,BC
A0F1  18 83          JR    &A076        ;Ligne suivante
A0F3  E6 0F          AND   &0F
A0F5  C6 30          ADD   A,&30        ;Affichage hexa
A0F7  FE 3A          CP    &3A
A0F9  38 02          JR    C,&A0FD
A0FB  C6 07          ADD   A,&07
A0FD  C3 5A BB       JP    &BB5A
A100  52 4F 4D 54 45 DEFM  "ROMTEST "
A105  53 54 20 64 65 DEFM  "detection de "
A10A  74 56 63 74 69 DEFM  "ROM sur CPC."
A10F  6F 6E 20 64 65 DEFB 0
A114  20 52 4F 4D 20
A119  73 75 72 20 43
A11E  50 43 2E 00
```

ROMTEST.BAS (générateur de ROMTEST.BIN) :

```
10 MEMORY &9FFF:total=0:ligne=100
20 FOR i=&A000 TO &A121 STEP 8:FOR j=0 TO 7:READ a$:a=V AL
("&"+a$):total=(total*2+a) AND &FFF:POKE i+j,a:NEXT j
30 READ a$:a=VAL("&"+a$):IF total<>a THEN PRINT "Erreur
ligne :",ligne:STOP
40 ligne=ligne+10:NEXT i:SAVE "ROMTEST.BIN",b,&A000,&12
2,&A000:END
100 DATA 3E,40,CD,08,BC,3E,01,CD,0C7
110 DATA 0E,BC,3E,01,CD,63,BB,01,E3B
120 DATA 00,00,CD,38,BC,01,00,00,E04
130 DATA AF,CD,32,BC,01,18,18,3E,196
140 DATA 01,CD,32,BC,01,06,06,3E,C2A
150 DATA 03,CD,32,BC,CD,BA,BB,3E,BC4
160 DATA 03,CD,59,BC,11,02,00,21,051
170 DATA 83,01,CD,C0,BB,3E,03,CD,003
180 DATA DE,BB,21,00,A1,7E,23,B7,CDD
190 DATA 28,05,CD,5A,BB,18,F6,11,9B5
200 DATA 00,00,21,85,01,CD,C0,BB,6E7
210 DATA 3E,01,CD,DE,BB,21,00,A1,4BD
220 DATA 7E,23,B7,28,05,CD,5A,BB,2EB
230 DATA 18,F6,AF,CD,59,BC,11,00,D0A
240 DATA 00,21,3F,01,3E,00,ED,53,E4D
250 DATA 00,A1,22,02,A1,32,04,A1,019
260 DATA F3,06,DF,ED,79,D9,79,E6,7B4
270 DATA F7,ED,79,3A,00,C0,ED,49,2A3
280 DATA D9,FE,80,28,20,2F,32,00,3A0
290 DATA C0,47,D9,79,E6,F7,ED,79,1CF
300 DATA 3A,00,C0,ED,49,D9,B8,20,A0C
310 DATA 08,2F,32,00,C0,3E,03,18,916
320 DATA 05,3E,01,18,01,AF,FB,CD,F27
330 DATA DE,BB,ED,5B,00,A1,2A,02,AEA
340 DATA A1,CD,C0,BB,3A,04,A1,F5,587
350 DATA 0F,0F,0F,0F,CD,F3,A0,F1,175
360 DATA CD,F3,A0,2A,00,A1,11,28,1AE
370 DATA 00,19,EB,2A,02,A1,3A,04,74C
380 DATA A1,3C,28,8A,47,E6,0F,78,F86
390 DATA 20,8C,11,00,00,01,F0,FF,E03
400 DATA 09,18,83,E6,0F,C6,30,FE,12E
410 DATA 3A,38,02,C6,07,C3,5A,BB,A53
420 DATA 52,4F,4D,54,45,53,54,20,2DC
430 DATA 64,65,74,65,63,74,69,6F,239
440 DATA 6E,20,64,65,20,52,4F,4D,E03
450 DATA 20,73,75,72,20,43,50,43,86F
460 DATA 2E,00,00,00,00,00,00,00,600
```

Programme RAMCARD.BAS:

```
10 MODE 1:PRINT CHR$(24)+" RAMCARD "+CHR$(24)+"Programm
ation de la RAMCARD":PRINT
20 MEMORY &5FFF:adr=&A000:FOR i=1 TO 18:READ a$:FOR j=1  TO
LEN(a$) STEP 2:POKE adr,VAL("&"+MID$(a$,j,2)):adr=ad
r+1:NEXT j,i
30 PRINT:INPUT"Nom du fichier :",a$
40 LOAD a$,&6000
50 INPUT"Numero de ROM :&",a$
60 POKE &A004,VAL("&"+a$)
70 PRINT:PRINT"Placer le switch d'ecriture sur ON.":PRI  NT
"Appuyer sur une touche pour programmer."
80 CALL &A000:GOTO 30
100 DATA CD18BB0E00CD0FB9C52100601100C0
110 DATA 010040EDB001B2A02100C01100601A
120 DATA BE200A132C20F82420F5014CA0C53A
130 DATA 04A02F4FCD0FB93E01CD0EBCE17EB7
140 DATA 280623CD5ABB18F6CD18BBC1CD18B9
150 DATA C90D0A50726F6772616D6D6174696F
160 DATA 6E207465726D696E65652061766563
170 DATA 207375636365732E0D0A0A506C6163
180 DATA 6572206C652073776974636820642 7
190 DATA 656372697475726520737572204F46
200 DATA 462E0D0A417070757965722073757 2
210 DATA 20756E6520746F756368652E000D0A
220 DATA 457272657572206461 6E73206C6120
230 DATA 70726F6772616D6D6174696F6E2021
240 DATA 21210D0A0A41707075796572207375
250 DATA 7220756E6520746F75636865206574
260 DATA 207265636F6D6D656E6365722E0D0A
270 DATA 00
```

Routine assembleur de RAMCARD.BAS:

```
A000  CD 18 BB       CALL  &BB18        ;Attend touche
A003  0E 00          LD    C,&00        ;No ROM
A005  CD 0F B9       CALL  &B90F        ;Connecter ROM
A008  C5             PUSH  BC           ;Sauve état ROM
A009  21 00 60       LD    HL,&6000     ;Source
A00C  11 00 C0       LD    DE,&C000     ;Ecran/RAMCARD
A00F  01 00 40       LD    BC,&4000     ;Programmation
A012  ED B0          LDIR
A014  01 B2 A0       LD    BC,&A0B2     ;Message erreur
A017  21 00 C0       LD    HL,&C000     ;Ecran/RAMCARD
A01A  11 00 60       LD    DE,&6000     ;Source
A01D  1A             LD    A,(DE)       ;Vérification
A01E  BE             CP    (HL)
A01F  20 0A          JR    NZ,&A02B     ;Erreur détectée
A021  13             INC   DE
A022  2C             INC   L
A023  20 F8          JR    NZ,&A01D     ;Octet suivant
A025  24             INC   H
A026  20 F5          JR    NZ,&A01D
A028  01 4C A0       LD    BC,&A04C     ;Message ok
A02B  C5             PUSH  BC
A02C  3A 04 A0       LD    A,(&A004)
A02F  2F             CPL
A030  4F             LD    C,A          ;ROM bidon pour
A031  CD 0F B9       CALL  &B90F        ;écrire message
A034  3E 01          LD    A,&01        ;sans détruire
A036  CD 0E BC       CALL  &BC0E        ;la RAMCARD
A039  E1             POP   HL
A03A  7E             LD    A,(HL)
A03B  B7             OR    A
A03C  28 06          JR    Z,&A044
A03E  23             INC   HL
A03F  CD 5A BB       CALL  &BB5A
A042  18 F6          JR    &A03A
A044  CD 18 BB       CALL  &BB18        ;Attend touche
A047  C1             POP   BC
A048  CD 18 B9       CALL  &B918        ;Restaure ROM
A04B  C9             RET
A04C  0D 0A 50 72 6F
A051  67 72 61 6D 6D
A056  61 74 69 6F 6E DEFB  13,10
A05B  20 74 65 72 6D DEFM  "Programmation "
A060  69 6E 65 65 20 DEFM  "terminee avec "
A065  61 76 65 63 20 DEFM  "succes."
A06A  73 75 63 63 65 DEFB  13,10,10
A06F  73 2E 0D 0A 0A
A074  50 6C 61 63 65
A079  72 20 6C 65 20
```

```
A07E  73 77 69 74 63 DEFM  "Placer le switch"
A083  68 20 64 27 65 DEFM  " d'ecriture "
A088  63 72 69 74 75 DEFM  "sur OFF."
A08D  72 65 20 73 75 DEFB  13,10
A092  72 20 4F 46 46
A097  2E 0D 0A 41 70
A09C  70 75 79 65 72 DEFM  "Appuyer sur "
A0A1  20 73 75 72 20 DEFM  "une touche."
A0A6  75 6E 65 20 74 DEFB  0
A0AB  6F 75 63 68 65
A0B0  2E 00
A0B2  0D 0A 45 72 72
A0B7  65 75 72 20 64 DEFB  13,10
A0BC  61 6E 73 20 6C DEFM  "Erreur dans la "
A0C1  61 20 70 72 6F DEFM  "programmation !!!"
A0C6  67 72 61 6D 6D DEFB  13,10,10
A0CB  61 74 69 6F 6E
A0D0  20 21 21 21 0D
A0D5  0A 0A 41 70 70
A0DA  75 79 65 72 20
A0DF  73 75 72 20 75 DEFM  "Appuyer sur "
A0E4  6E 65 20 74 6F DEFM  "une touche et"
A0E9  75 63 68 65 20 DEFM  " recommencer."
A0EE  65 74 20 72 65 DEFB  13,10,0
A0F3  63 6F 6D 6D 65
A0F8  6E 63 65 72 2E
A0FD  0D 0A 00
```

Voilà j'espère avoir été assez clair, je me suis efforcé de traiter tous les aspects des ROMs sur le CPC et j'en oublie sûrement, il y a tant de chose à dire et à développer sur les ROMs d'extension du CPC, donc si vous avez des remarques, suggestions, questions ou si vous désirez de l'aide pour la réalisation d'un logiciel en ROM, n'hésitez pas à me contacter. Ecrivez-moi :

Mr DOS SANTOS Francisco
123 A_3 bd STRASBOURG
94130 NOGENT SUR MARNE

(tél: 01-48-75-34-82)

Cette notice (version **1.0**) a été téléchargé sur le sîte du **COCOON SYSTEM** en Octobre 2001. Je vous donne rendez-vous pour les prochaines réalisations …

Electroniquement votre,
RAM 7

Echelle 1/1
(liste des composants page 2)

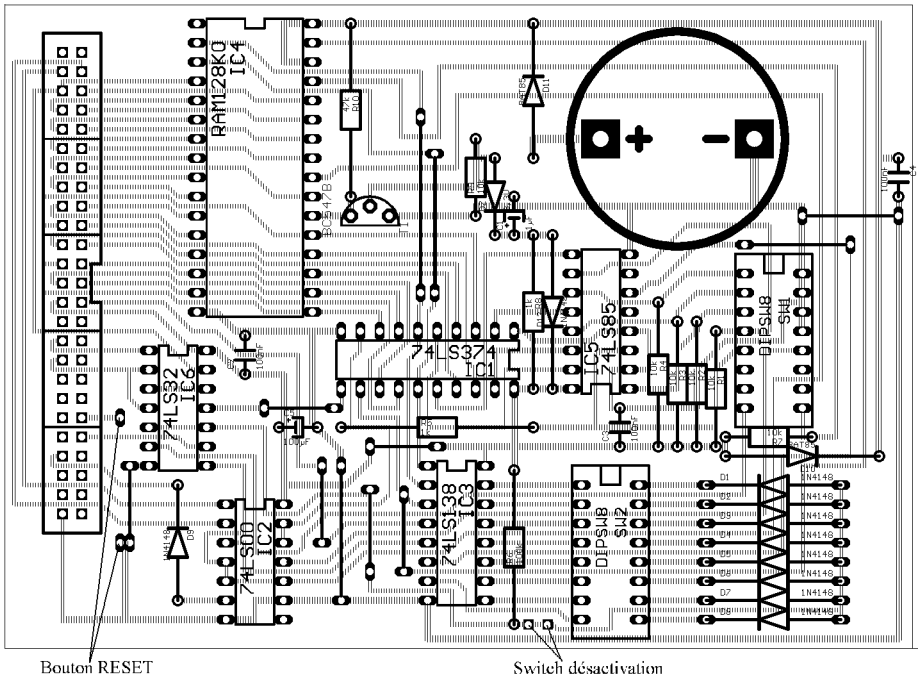

Schéma d'implantation
des composants
(RAMCARD v2.3)

Note : Le bouton RESET et le switch de désactivation de toutes les ROMs sont facultatifs. Attention dans le dernier cas n'oubliez pas de couper la piste entre les 2 pastilles du switch.